

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#)

# Dokumentation zum Apache HTTP Server Version 2.0

Google-Suche

## Hinweise zur Version

[Neue Funktionen in Version 2.0](#)

[Upgrade auf Version 2.0](#)

[Apache-Lizenz](#)

## **Bedienungsanleitung**

[Kompilieren und Installieren](#)

[Starten](#)

[Stoppen und Neustarten](#)

[Konfigurationsanweisungen](#)

[Kurzreferenz der Direktiven](#)

[Module](#)

[Multi-Processing-Module \(MPMs\)](#)

[Filter](#)

[Handler](#)

[Server und Hilfsprogramme](#)

[Glossar](#)

## Benutzerhandbuch

[Bindung](#)

[Konfigurationsdateien](#)

[Konfigurationsabschnitte](#)

[Content Negotiation](#)

[Dynamic Shared Objects \(DSO\)](#)

[Umgebungsvariablen](#)

[Log-Dateien](#)

[URLs auf das Dateisystem abbilden](#)

[Performance-Hinweise](#)

[Tipps zur Sicherheit](#)

[Serverweite Konfiguration](#)

[SSL/TLS-Verschlüsselung](#)

[Suexec für CGI](#)

[Einführung in die URL-Manipulation](#)

[Virtuelle Hosts](#)

## Praxis / Anleitungen

[Authentisierung, Autorisierung und Zugriffskontrolle](#)

[CGI: Dynamische Inhalte](#)

[.htaccess-Dateien](#)

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

[Web-Verzeichnisse für Benutzer \(public\\_html\)](#)

## Plattform-spezifische Anmerkungen

[Microsoft Windows](#)

[Novell NetWare](#)

[EBCDIC-Portierung](#)

## Weitere Themen

[Häufig gestellte Fragen \(FAQ\)](#)

[Seitenindex](#)

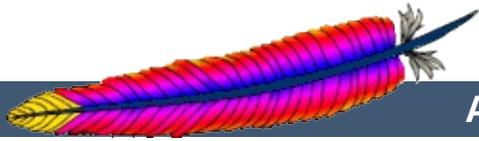
[Dokumentation für Entwickler](#)

[Sonstiges](#)

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

## Upgrade von 1.3 auf 2.0

Dieses Dokument dient der Unterstützung beim Upgrade. Es enthält die entscheidenden Informationen für bisherige Apache-Nutzer. Diese sind als kurze Anmerkungen gedacht. Weitere Informationen finden Sie entweder unter [Neue Funktionen](#) oder in den src/CHANGES-Dateien.

### Siehe auch

[Übersicht der neuen Funktionen in Apache 2.0](#)



## Änderungen der Konfiguration und der Kompilierung

- Der Apache benutzt jetzt ein autoconf- und libtool-System zur [Konfiguration des Erstellungsverfahrens](#). Die Verwendung dieses Systems ist ähnlich, aber nicht identisch mit dem APACI-System des Apache 1.3.
- Zusätzlich zu der üblichen Auswahl von Modulen, die kompiliert werden sollen, wurde der Hauptteil der Request-Verarbeitung im Apache 2.0 in die [Multi-Processing-Module \(MPMs\)](#) verschoben.



- Viele Anweisungen aus dem Serverkern des Apache 1.3 sind jetzt in den MPMs enthalten. Wenn Sie ein Serververhalten wünschen, das demjenigen des Apache 1.3 möglichst ähnlich ist, sollten Sie das [prefork](#)-MPM auswählen. Andere MPMs verwenden abweichende Anweisungen für die Prozess-Erstellung und Request-Verarbeitung.
- Das [Proxy-Modul](#) wurde umgearbeitet, um es auf den Stand von HTTP/1.1 zu bringen. Eine der bedeutendsten Änderungen ist die Platzierung der Proxy-Zugriffskontrolle innerhalb eines [<Proxy>](#)-Blocks, statt innerhalb eines `<Directory proxy:>`-Blocks.
- Die Behandlung von PATH\_INFO (hinter dem tatsächlichen Dateinamen angefügte Pfadangaben) wurde für einige Module geändert. Module, die bisher als Handler implementiert waren, jetzt aber als Filter implementiert sind, akzeptieren möglicherweise keine Requests mit PATH\_INFO mehr. Filter wie [INCLUDES](#) oder [PHP](#) sind gleich oben im Core-Handler implementiert und weisen deshalb Requests mit PATH\_INFO ab. Sie können die [AcceptPathInfo](#)-Direktive verwenden, um den Core-Handler zu zwingen, Requests mit PATH\_INFO zu akzeptieren, und dadurch die Fähigkeit wiederherstellen, PATH\_INFO in Server Side Includes zu benutzen.
- Die [CacheNegotiatedDocs](#)-Direktive hat jetzt das Argument an (on) oder aus (off). Die vorhandenen Anweisungen [CacheNegotiatedDocs](#) sollten durch `CacheNegotiatedDocs on` ersetzt werden.
- Die [ErrorDocument](#)-Direktive verwendet kein Anführungszeichen mehr am Anfang des Arguments, um eine Textnachricht anzuzeigen. Stattdessen sollten Sie die Nachricht in doppelte Anführungszeichen einschließen. Zum Beispiel sollten existierende Angaben wie

```
ErrorDocument 403 "Eine Nachricht"
```

durch

```
ErrorDocument 403 "Eine Nachricht"
```

ersetzt werden. Solange das zweite Argument kein gültiger URL oder Pfadname ist, wird es als Textnachricht behandelt.

- Die Direktiven `AccessConfig` und `ResourceConfig` sind entfallen. Diese Direktiven können durch die `Include`-Direktive ersetzt werden, die eine äquivalente Funktionalität besitzt. Wenn Sie die Defaultwerte dieser Direktiven verwendet haben, ohne sie in die Konfigurationsdateien einzufügen, müssen Sie möglicherweise `Include conf/access.conf` und `Include conf/srm.conf` zu Ihrer `httpd.conf` hinzufügen. Um sicherzustellen, daß der Apache die Konfigurationsdateien in der gleichen Reihenfolge liest, wie sie von den älteren Direktiven impliziert wurde, sollten die `Include`-Direktiven ans Ende der `httpd.conf` gestellt werden, wobei die Direktive für `srm.conf` derjenigen für `access.conf` vorangeht.
- Die Direktiven `BindAddress` und `Port` sind entfallen. Eine äquivalente Funktionalität wird von der flexibleren Direktive `Listen` bereitgestellt.
- Im Apache 1.3 wurde die `Port`-Direktive außerdem dazu verwendet, die Portnummer für selbstreferenzierende URLs festzulegen. Die neue `ServerName`-Syntax stellt das Apache-2.0-Äquivalent dar: sie wurde dahingehend verändert, sowohl den Hostnamen *als auch* die Portnummer für selbstreferenzierende URLs in einer Direktive angeben zu können.
- Die `ServerType`-Direktive entfällt. Die Methode zum Bedienen der Requests wird nun durch die Auswahl des MPM

ermittelt. Derzeit ist kein MPM dafür bestimmt, von inetd gestartet zu werden.

- Die Module `mod_log_agent` und `mod_log_referer`, welche die Direktiven `AgentLog`, `RefererLog` und `RefererIgnore` bereitgestellt hatten, wurden entfernt. Durch Verwendung der Direktive `CustomLog` aus `mod_log_config` sind die Agent- und Refererlogs auch weiterhin verfügbar.
- Die Direktiven `AddModule` und `ClearModuleList` sind entfallen. Diese Direktiven wurden benutzt, um sicherzustellen, daß die Module in der richtigen Reihenfolge aktiviert werden können. Die neue Apache 2.0 API erlaubt es Modulen, ihre Reihenfolge explizit anzugeben, und macht diese Direktiven damit überflüssig.
- Die Direktive `FancyIndexing` wurde entfernt. Die gleiche Funktionalität ist nun mit der Option `FancyIndexing` der Direktive `IndexOptions` verfügbar.
- Die von `mod_negotiation` bereitgestellte Content-Negotiation-Technik `MultiViews` führt nun eine strengere Dateierkennung durch. Es wird ausschließlich unter den *aushandelbaren* Dateien gewählt. Das bisherige Verhalten kann jedoch mit der Direktive `MultiViewsMatch` wiederhergestellt werden.
- (*Ab Version 2.0.51*) `ErrorHeader` war eine Fehlbenennung, weshalb die Funktionalität dieser Direktive mit der `Header`-Anweisung zusammengelegt wurde. Verwenden Sie stattdessen

```
Header always set foo bar
```

um den gleichen Effekt zu erzielen.



- Das Modul `mod_auth_digest`, das im Apache 1.3 experimentellen Status hatte, ist nun ein Standardmodul.
- Das Modul `mod_mmap_static`, das im Apache 1.3 experimentellen Status hatte, wurde durch das Modul `mod_file_cache` ersetzt.
- Die Distribution wurde komplett reorganisiert und enthält kein unabhängiges `src`-Verzeichnis mehr. Stattdessen wurden die Quellcodes logisch unterhalb des Hauptverzeichnisses der Distribution angeordnet. Installationen des kompilierten Servers sollten in ein separates Verzeichnis erfolgen.

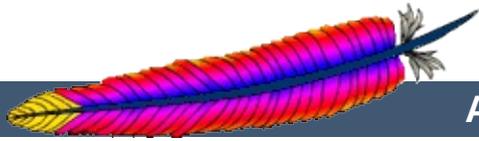


An der API des Apache 2.0 wurden umfassende Änderungen vorgenommen. Bestehende Module, die für die Apache 1.3 API entwickelt wurden, werden **nicht** ohne Modifikationen mit der Version 2.0 des Apache zusammenarbeiten. Details sind in der [Dokumentation für Entwickler](#) beschrieben.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

# Übersicht der neuen Funktionen in Apache 2.0

Dieses Dokument beschreibt einige der wichtigsten Änderungen des Apache HTTP Servers 2.0 gegenüber der Version 1.3.

## Siehe auch

[Upgrade von 1.3 auf 2.0](#)



## Unix-Threading

Auf Unix-Systemen mit Unterstützung für POSIX-Threads, kann Apache jetzt in einem Multi-Process, Multi-Threaded Hybrid-Mode gestartet werden. Dies verbessert die Skalierfähigkeit für viele, jedoch nicht unbedingt alle Konfigurationen.

## Neues Build-System

Das Build-System wurde komplett auf der Basis von `autoconf` und `libtool` neu geschrieben. Dadurch wird das Apache-Konfigurationssystem dem vieler anderer Packages ähnlicher.

## Multi-Protokoll-Unterstützung

Apache stellt jetzt die notwendigen Grundfunktionalitäten bereit, um mehrere Protokolle unterstützen und verarbeiten zu können. `mod_echo` wurde hierfür als Beispiel geschrieben.

## Bessere Unterstützung von Nicht-Unix-Plattformen

Apache 2.0 ist schneller und stabiler auf Nicht-Unix-Plattformen wie BeOS, OS/2 und Windows. Mit der Einführung von Plattform-spezifischen [Multi-Processing Modulen](#) (MPMs) und der Apache Portable Runtime (APR), sind diese Plattformen jetzt in ihrem nativen API implementiert, wodurch die Verwendung der häufig fehlerbehafteten und schlecht funktionierenden POSIX-Emulation-Layer vermieden wird.

## Neues Apache API

Das API für Module hat sich in 2.0 stark verändert. Die meisten der Sortierungs-/Prioritätsprobleme von Modulen bei 1.3 sollten nun verschwunden sein. In 2.0 wird hiervon vieles automatisch durchgeführt. Die Modulsortierung wird jetzt über einen pre-hook vorgenommen, um mehr Flexibilität zu bieten. Außerdem wurden neue API-Calls hinzugefügt, die

zusätzliche Modulfähigkeiten zur Verfügung stellen, ohne den Apache-Kern anpassen zu müssen.

### **IPv6-Unterstützung**

Auf Systemen, bei denen die zugrundeliegende Apache Portable Runtime-Bibliothek IPv6 unterstützt, bekommt Apache standarmäßig IPv6 Listening Sockets. Zusätzlich unterstützen die Konfigurationsanweisungen [Listen](#), [NameVirtualHost](#) und [VirtualHost](#) numerische IPv6-Adressangaben (z.B., "Listen [2001:db8::1]:8080").

### **Filterung**

Apache-Module können jetzt als Filter entwickelt und zur Filterung des rein- und rausgehenden Datenstroms des Servers eingesetzt werden. Hierdurch kann beispielsweise die Ausgabe von CGI-Skripten durch den INCLUDE-Filter von [mod\\_include](#) bearbeitet werden und so Server-Side Include-Anweisungen ausgeführt werden. Das Modul [mod\\_ext\\_filter](#) erlaubt externen Programmen als Filter zu agieren, in der gleichen Weise wie CGI-Programme als Eingabe dienen können.

### **Mehrsprachige Fehlermeldungen**

Fehlermeldungen die an den Browser rausgehen, stehen jetzt als SSI-Dokumente in verschiedenen Sprachen zur Verfügung. Sie können bei Bedarf durch den Administrator angepasst werden, um ein einheitliches Design zu erreichen.

### **Vereinfachte Konfiguration**

Viele der verwirrenden Konfigurationsanweisungen wurden vereinfacht. Die oft für Verwirrung sorgenden Port- und BindAddress-Anweisungen wurden entfernt. Ausschließlich die [Listen](#)-Anweisung wird nun zum Setzen von IP-Adressen und Portnummern benutzt. Der Servername und die Portnummer, die für Weiterleitungen und zur Erkennung virtueller Server verwendet werden, werden über die

[ServerName](#)-Anweisung konfiguriert.

### **Native Windows NT Unicode-Unterstützung**

Apache 2.0 auf Windows NT benutzt jetzt utf-8 für alle Dateinamen-Kodierungen. Diese werden direkt auf das zugrundeliegende Unicode-Dateisystem abgebildet, wodurch Mehrsprach-Unterstützung für alle Windows NT-basierten Installationen, inklusive Windows 2000 und Windows XP, zur Verfügung gestellt wird. *Diese Unterstützung ist nicht auf Windows 95, 98 oder ME verfügbar. Hier wird weiterhin die jeweils lokale Codepage des Rechners für den Zugriff auf das Dateisystem verwendet.*

### **Bibliothek für reguläre Ausdrücke aktualisiert**

Apache 2.0 enthält die ["Perl Compatible Regular Expression Library"](#) (PCRE). Bei der Auswertung aller regulären Ausdrücke wird nun die leistungsfähigere Syntax von Perl 5 verwendet.



## mod\_ssl

Neues Modul in Apache 2.0. Dieses Modul ist ein Interface zu den von OpenSSL bereitgestellten SSL/TLS Verschlüsselungs-Protokollen.

## mod\_dav

Neues Modul in Apache 2.0. Dieses Modul implementiert die HTTP Distributed Authoring and Versioning (DAV) Spezifikation zur Erzeugung und Pflege von Web-Inhalten.

## mod\_deflate

Neues Modul in Apache 2.0. Dieses Modul erlaubt es Browsern, die dies unterstützen, eine Komprimierung des Inhaltes vor der Auslieferung anzufordern, um so Netzwerk-Bandbreite zu sparen.

## mod\_auth\_ldap

Neues Modul in Apache 2.0.41. Diese Modul ermöglicht die Verwendung einer LDAP-Datenbank zur Speicherung von Berechtigungsdaten für die HTTP-Basic-Authentication. Ein Begleitmodul, mod\_ldap, stellt einen Verbindungs-Pool und die Pufferung von Abfrageergebnissen zur Verfügung.

## mod\_auth\_digest

Zusätzliche Unterstützung für prozessübergreifendes Session-Caching mittels Shared-Memory.

## mod\_charset\_lite

Neues Modul in Apache 2.0. Dieses experimentelle Modul erlaubt Zeichensatz-Übersetzungen oder -Umschlüsselung.

## mod\_file\_cache

Neues Modul in Apache 2.0. Dieses Modul beinhaltet die Funktionalität von `mod_mmap_static` aus Apache 1.3, plus einige weitere Caching-Funktionen.

## mod\_headers

Dieses Modul ist in Apache 2.0 deutlich flexibler geworden. Es kann jetzt die von [mod\\_proxy](#) genutzten Request-Header manipulieren und es ist möglich Response-Header auf Basis von definierten Bedingungen zu verändern.

### mod\_proxy

Das Proxy Modul wurde komplett neu geschrieben um die Möglichkeiten der neuen Filter-Funktionalität auszuschöpfen und um einen zuverlässigen Proxy zu haben, der den HTTP/1.1-Spezifikationen entspricht. Neue [<Proxy>](#) - Konfigurationsabschnitte bieten eine besser lesbare (und intern schnellere) Kontrolle der vermittelten Seiten. Die überladenen `<Directory "proxy:...">`-Konfigurationen werden nicht mehr unterstützt. Das Modul ist nun in mehrere Module unterteilt, die jeweils ein bestimmtes Übertragungsprotokoll unterstützen, wie `proxy_connect`, `proxy_ftp` und `proxy_http`.

### mod\_negotiation

Die neue Konfigurationsanweisung [ForceLanguagePriority](#) kann benutzt werden, um sicherzustellen, dass ein Client auf jeden Fall ein einzelnes Dokument, anstatt einer NOT ACCEPTABLE- oder MULTIPLE CHOICES-Antwort, bekommt. Zusätzlich wurden die Negotiation- und Multiview-Algorithmen angepasst um einheitlichere Ergebnisse zu liefern. Außerdem wird ein neues Type-Map-Format bereitgestellt, das Dokumenteninhalte direkt enthalten kann.

### mod\_autoindex

Automatisch erzeugte Verzeichnisindizes können zur besseren Übersichtlichkeit durch HTML-Tabellen dargestellt werden. Genauere Sortierungen, wie Sortierung nach Versionsnummer und Wildcard-Filterung des Verzeichnisindizes werden unterstützt.

## **mod\_include**

Neue Anweisungen erlauben es, die Standard Start- und Endtags von SSI-Elementen zu ändern. Zudem können die Default-Formate für Fehlermeldungen und Zeitangaben nun ebenfalls in der Serverkonfiguration vorgenommen werden. Auf die Ergebnisse der Auswertung und Gruppierung von regulären Ausdrücken (jetzt auf Basis der Perl-Syntax für reguläre Ausdrücke) kann über die **mod\_include** Variablen \$0 bis \$9 zugegriffen werden.

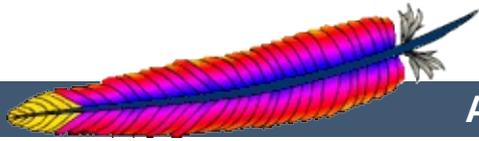
## **mod\_auth\_dbm**

DBM-ähnliche Datenbanken werden jetzt durch die Konfigurationsanweisung **AuthDBMType** unterstützt.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



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



## Apache HTTP Server Version 2.0

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

# The Apache License, Version 2.0

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

## TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

### 1. Definitions

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

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

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

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

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

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not

limited to compiled object code, generated documentation, and conversions to other media types.

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

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

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

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by

Licensor and subsequently incorporated within the Work.

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

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

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

5. **Submission of Contributions.** Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall

supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

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

or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

## END OF TERMS AND CONDITIONS

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

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

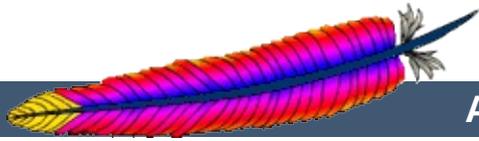
```
Copyright [yyyy] [name of copyright owner]
```

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

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

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

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



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

# Kompilieren und Installieren

Dieses Dokument umfaßt nur die Kompilierung und Installation des Apache auf Unix und Unix-ähnlichen Systemen. Für die Kompilierung und Installation unter Windows lesen Sie bitte [Den Apache unter Microsoft Windows betreiben](#). Für andere Plattformen lesen Sie bitte die Dokumentation [Plattformen](#).

Die Konfigurations- und Installationsumgebung des Apache 2.0 hat sich seit dem Apache 1.3 komplett verändert. Der Apache 1.3 benutzt einen speziellen Satz von Skripten, um eine einfache Installation zu ermöglichen. Der Apache 2.0 dagegen verwendet nun `libtool` und `autoconf`, um eine Umgebung zu schaffen, die der vieler anderer Open Source Projekte ähnlich sieht.

Wenn Sie von einer Unterversion auf die nächste aktualisieren (z.B. von 2.0.50 auf 2.0.51), springen Sie bitte zum Abschnitt [Upgrade](#).

## Siehe auch

[Den Quellcode konfigurieren](#)

[Apache starten](#)

[Beenden und Neustarten](#)





**Anforderungen**

Folgende Anforderungen gelten für die Erstellung des Apache:

### **Plattenplatz**

Stellen Sie sicher, dass Sie kurzzeitig wenigstens 50 MB freien Festplattenplatz zur Verfügung haben. Nach der Installation belegt der Apache ungefähr 10 MB Plattenplatz. Der tatsächliche Platzbedarf variiert in Abhängigkeit von den gewählten Konfigurationseinstellungen und Modulen von Drittanbietern.

### **ANSI-C-Compiler und Generierungswerkzeuge**

Stellen Sie sicher, dass Sie einen ANSI-C Compiler installiert haben. Der [GNU C Compiler \(GCC\)](#) der [Free Software Foundation \(FSF\)](#) ist empfehlenswert (Version 2.7.2 ist gut). Wenn Sie den GCC nicht besitzen, stellen Sie zumindest sicher, dass der Compiler Ihres Anbieters ANSI-kompatibel ist. Außerdem muss Ihr PATH wesentliche Generierungswerkzeuge wie make enthalten.

### **Zeitgenauigkeit bewahren**

Elemente des HTTP-Protokolls werden in Form einer Tageszeit ausgedrückt. Darum sollten Sie jetzt prüfen, ob Ihr System die Fähigkeit zur Zeitsynchronisation besitzt, und diese gegebenenfalls installieren. Üblicherweise werden hierfür die Programme `ntpd` oder `xntpd` verwendet, die auf dem Network Time Protocol (NTP) basieren. Nähere Informationen über NTP Software und öffentliche Zeitserver finden Sie in der Usenet Newsgroup [comp.protocols.time.ntp](#) und auf der [NTP Homepage](#).

### **Perl 5 [OPTIONAL]**

Für einige Hilfsskripte wie [apxs](#) oder [dbmmanage](#) (die in Perl geschrieben sind) wird der Perl 5 Interpreter benötigt (die Versionen ab 5.003 sind ausreichend). Wenn Sie mehrere Perl Interpreter haben (beispielsweise eine systemweite

Installation von Perl 4 und Ihre eigene Perl 5-Installation), dann sollten Sie die `--with-perl`-Option (siehe unten) verwenden, um sicherzustellen, dass der richtige Interpreter von `configure` ausgewählt wird. Wenn kein Perl 5-Interpreter vom `configure`-Skript gefunden werden kann, können Sie die betroffenen Hilfsskripte nicht verwenden, sind jedoch selbstverständlich nach wie vor in der Lage, den Apache 2.0 zu bauen und zu installieren.



Der Apache kann von der [Apache HTTP Server Downloadseite](#) heruntergeladen werden, auf der verschiedene Spiegelserver angegeben sind. Für die meisten Benutzer des Apache ist es auf Unix-ähnlichen Systemen am Besten, die Quellcodeversion herunterzuladen und zu kompilieren. Der Erstellungsprozess (weiter unten beschrieben) ist einfach und erlaubt es Ihnen, den Server Ihren Bedürfnissen anzupassen. Dazu kommt, dass Binärdistributionen gegenüber der aktuellen Quellcodeversion oft veraltet sind. Wenn Sie tatsächlich ein Binärpaket herunterladen, folgen Sie bitte den Anweisungen in der Datei `INSTALL.bindist`, die der Distribution beiliegt.

Es ist wichtig, dass Sie nach dem Herunterladen überprüfen, dass es sich um eine vollständige und unveränderte Version des Apache HTTP Servers handelt. Das können Sie erreichen, indem Sie das heruntergeladene Paket gegen die PGP-Signatur prüfen. Einzelheiten dazu erfahren Sie auf der [Download-Seite](#). Es ist auch ein erweitertes Beispiel verfügbar, das die [Anwendung von PGP](#) beschreibt.



## Auspacken

Das Auspacken des Quellcodes aus dem Apache HTTPD Tarball besteht aus einem simplen Dekomprimieren und danach "Enttarren":

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

Dies erstellt unterhalb des aktuellen Verzeichnisses ein neues Verzeichnis, das den Quellcode für die Distribution enthält. Sie sollten mit `cd` in dieses Verzeichnis wechseln, bevor Sie mit der Kompilierung des Servers weitermachen.



Der nächste Schritt ist die Konfiguration des Apache-Codebaumes für Ihre spezielle Plattform und Ihre persönlichen Bedürfnisse. Dies wird mit dem Skript [configure](#) durchgeführt, das im Wurzelverzeichnis der Distribution enthalten ist. (Entwickler, welche die CVS Version des Apache-Codebaumes herunterladen, müssen `autoconf` und `libtool` installiert haben und müssen `buildconf` ausführen, bevor sie mit den nächsten Schritten fortfahren können. Dies wird bei offiziellen Releases nicht notwendig sein.)

Um den Codebaum mit den Standardeinstellungen zu konfigurieren, geben Sie einfach `./configure` ein. Zur Änderung dieser Voreinstellungen akzeptiert [configure](#) eine Reihe von Variablen und Kommandozeilenoptionen.

Die wichtigste Option ist `--prefix`, der Ablageort, an dem der Apache später installiert wird, da er für diesen Ort konfiguriert werden muss, um korrekt zu arbeiten. Eine feinere Einstellung der Dateiablagen ist mit weiteren [configure-Optionen](#) möglich.

Weiterhin können Sie zu diesem Zeitpunkt festlegen, welche [Funktionalität](#) Sie in den Apache aufnehmen möchten, indem Sie [Module](#) aktivieren oder deaktivieren. Der Apache bindet standardmäßig einen Satz von [Basismodulen](#) ein. Andere Module werden mit Hilfe der Option `--enable-module` aktiviert, wobei *module* den Namen des Moduls ohne das Präfix `mod_` darstellt. Ausserdem sind alle Unterstriche durch Bindestriche zu ersetzen. Sie können sich auch entscheiden, Module als "[Shared Objects \(DSOs\)](#)" zu kompilieren, welche zur Laufzeit ge- und entladen werden können. Dazu verwenden Sie die Option `--enable-module=shared`. Entsprechend können Sie Basismodule mit der Option `--disable-module` deaktivieren. Lassen Sie Vorsicht walten, wenn Sie diese Optionen verwenden, da [configure](#) Sie

nicht warnen kann, wenn die von Ihnen angegebenen Module nicht existieren; die Option wird dann einfach ignoriert.

Zusätzlich ist es zuweilen notwendig, das [configure](#)-Skript mit Extrainformationen zum Ablageort Ihres Compilers, Ihrer Bibliotheken oder Header-Dateien zu versorgen. Das tun Sie, indem Sie entweder Umgebungsvariablen oder Kommandozeilenoptionen an [configure](#) übergeben. Für mehr Informationen lesen Sie bitte die Hilfeseite zu [configure](#).

Um einen kurzen Eindruck zu gewinnen, welche Möglichkeiten Sie haben, folgt hier ein typisches Beispiel, das den Apache mit einem speziellen Compiler und Compilerflags für das Installationsverzeichnis `/sk/pkg/apache` kompiliert, sowie die beiden zusätzlichen Module `mod_rewrite` und `mod_speling` für späteres Laden durch den DSO-Mechanismus:

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

Wenn [configure](#) startet, benötigt es mehrere Minuten, um die Verfügbarkeit von Features auf Ihrem System zu prüfen und ein Makefile zu generieren, das später zur Kompilierung des Servers verwendet wird.

Einzelheiten zu den vielen verschiedenen [configure](#)-Optionen finden Sie auf der Hilfeseite zu [configure](#).



Nun können Sie die verschiedenen Teile, die das Apache-Paket bilden, einfach durch Ausführen des folgenden Befehls erstellen:

```
$ make
```

Seien Sie hierbei bitte geduldig, denn eine Basiskonfiguration benötigt ungefähr 3 Minuten auf einem Pentium III/Linux 2.2. System. Dies kann aber abhängig von Ihrer Hardware und der Anzahl der Module, die Sie aktiviert haben, sehr stark variieren.



**INSTALLATION**

Nun endlich installieren Sie das Package unter dem konfigurierten Installations-*PREFIX* (siehe oben: Option `--prefix` durch Aufrufen von:

```
$ make install
```

Wenn Sie upgraden, wird die Installation Ihre Konfigurationsdateien oder Dokumente nicht überschrieben.



Als nächstes können Sie Ihren Apache HTTP Server anpassen, indem Sie die [Konfigurationsdateien](#) unterhalb von `PREFIX/conf/` editieren.

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

Werfen Sie auch einen Blick in das Apache-Handbuch unter [docs/manual/](#). Die aktuellste Version dieses Handbuchs sowie eine komplette Referenz der verfügbaren [Konfigurationsanweisungen](#) finden Sie unter <http://httpd.apache.org/docs/2.0/>.



Sie können nun Ihren Apache HTTP Server [starten](#), indem Sie einfach

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

ausführen.

Danach sollten Sie Ihr erstes Dokument unter dem URL `http://localhost/` anfordern können. Die Webseite, die Sie sehen, ist im [DocumentRoot](#) abgelegt, welches üblicherweise `PREFIX/htdocs/` ist. Den Server [stoppen](#) Sie wieder durch Ausführen von:

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



Der erste Schritt beim Aktualisieren besteht darin, die Versionsankündigung sowie die CHANGES-Datei in der Quelltextdistribution zu lesen, um Änderungen zu finden, die Ihr System möglicherweise betreffen. Wenn Sie einen größeren Versionssprung durchführen (z.B. vom 1.3 auf 2.0 oder von 2.0 auf 2.2), wird es wahrscheinlich auch größere Unterschiede in der Kompilier- und Laufzeitkonfiguration geben, die manuelle Nacharbeiten erfordern. Außerdem müssen alle Module aktualisiert werden, um den Änderungen der Modul-API gerecht zu werden.

Die Aktualisierung einer Unterversion auf eine andere (z.B. von 2.0.55 auf 2.0.57) ist einfacher. `make install` überschreibt keine der bereits existierenden Dokumente, Log- und Konfigurationsdateien. Ausserdem bemühen sich die Entwickler, inkompatible Änderungen der `configure`-Optionen, der Laufzeitkonfiguration sowie der Modul-API zu vermeiden. In den meisten Fällen sollten Sie in der Lage sein, den gleichen `configure`-Befehl, die gleiche Konfiguration und die gleichen Module wieder zu verwenden. (Das gilt erst seit Version 2.0.41 -- frühere Versionen enthielten noch inkompatible Änderungen).

Um auf eine neue Unterversion zu aktualisieren, suchen Sie zunächst die Datei `config.nice` im `build`-Verzeichnis Ihrer Serverinstallation oder im Wurzelverzeichnis des Quelltextbaums der alten Installation. Die Datei enthält den genauen `configure`-Befehl, der verwendet wurde, um den Quellcode zu konfigurieren. Um jetzt von einer Version auf die nächste zu aktualisieren, kopieren Sie einfach die `config.nice` in das Verzeichnis der neuen Version, passen sie bei Bedarf an, und führen Sie sie aus:

```
$ ./config.nice
$ make
$ make install
```

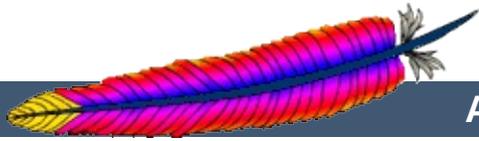
```
$ PREFIX/bin/apachectl stop  
$ PREFIX/bin/apachectl start
```

Sie sollten jede neue Version immer in Ihrer Umgebung testen, bevor Sie sie produktiv schalten. Beispielsweise können Sie die neue Version neben der alten installieren, indem Sie ein anderes `--prefix` und einen anderen Port wählen (durch Anpassen der `Listen`-Direktive). So können Sie auf eventuelle Inkompatibilitäten testen, bevor Sie endgültig die neue Version verwenden.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

## Apache starten

Unter Windows läuft der Apache üblicherweise als Dienst (Windows NT, 2000 und XP) oder als Konsolenanwendung (Windows 9x und ME). Für Einzelheiten lesen Sie bitte [Apache als Dienst betreiben](#) und [Apache als Konsolenanwendung betreiben](#).

Unter Unix wird das [httpd](#)-Programm als Daemon ausgeführt, der im Hintergrund fortlaufend aktiv ist, um Anfragen zu bearbeiten. Dieses Dokument beschreibt, wie [httpd](#) aufgerufen wird.

### Siehe auch

[Beenden und Neustarten](#)

[httpd](#)

[apachectl](#)



## How to start Apache

Wenn die in der Konfigurationsdatei angegebene [Listen](#)-Anweisung auf die Voreinstellung von 80 gesetzt ist (oder einen anderen Port unterhalb von 1024), dann müssen Sie root-Berechtigung besitzen, um den Apache starten zu können. Nur dann kann er sich an diesen privilegierten Port binden. Sobald der Server gestartet ist und einige vorbereitende Aktionen wie das Öffnen seiner Log-Dateien ausgeführt hat, startet er mehrere *Kind*-Prozesse, welche die Arbeit erledigen: das Lauschen auf und Beantworten von Anfragen von Clients. Der Haupt-`httpd`-Prozess läuft unter dem Benutzer `root` weiter, die *Kind*-Prozesse jedoch werden unter weniger privilegierten Benutzerkennungen ausgeführt. Dies wird von dem ausgewählten [Multi-Processing-Modul](#) gesteuert.

Die Verwendung des Steuerskripts [apachectl](#) ist die empfohlene Methode, das `httpd`-Programm zu starten. Dieses Skript setzt verschiedene Umgebungsvariablen, die für die korrekte Funktion von `httpd` unter einigen Betriebssystemen notwendig sind, und startet dann das `httpd`-Programm. [apachectl](#) reicht alle Kommandozeilenargumente durch, so dass alle `httpd`-Optionen auch mit [apachectl](#) verwendet werden können. Um den korrekten Ablageort des `httpd`-Programms sowie einige Kommandozeilenargumente anzugeben, die Sie *immer* verwenden möchten, können Sie auch das Skript [apachectl](#) direkt editieren und die Variable `HTTPD` am Anfang ändern.

Das Erste was `httpd` macht, wenn es startet, ist das Suchen und Einlesen der [Konfigurationsdatei](#) `httpd.conf`. Der Ablageort dieser Datei wird zur Kompilierungszeit festgelegt. Es ist aber möglich, den Ablageort zur Laufzeit anzugeben, indem die Kommandozeilenoption `-f` wie folgt verwendet wird:

```
/usr/local/apache2/bin/apachectl -f
```

```
/usr/local/apache2/conf/httpd.conf
```

Wenn während des Starts alles gutgeht, trennt sich der Server vom Terminal ab und die Eingabeaufforderung erscheint gleich darauf wieder. Dies zeigt an, dass der Server hochgefahren ist und läuft. Sie können nun Ihren Browser benutzen, um Verbindung zum Server aufzunehmen und sich die Testseite im [DocumentRoot](#)-Verzeichnis anzusehen wie auch die lokale Kopie der Dokumentation, die von dieser Seite aus verlinkt ist.



Wenn der Apache während des Hochfahrens einen schweren Fehler feststellt, schreibt er entweder eine Nachricht, die das Problem näher schildert, auf die Konsole oder ins [ErrorLog](#), bevor er sich selbst beendet. Eine der häufigsten Fehlermeldungen ist "Unable to bind to Port ..."  
(Anm.d.Ü.: "Kann nicht an Port ... binden"). Diese Meldung wird üblicherweise verursacht:

- entweder durch den Versuch, den Server an einem privilegierten Port zu starten, während man nicht als Benutzer root angemeldet ist,
- oder durch den Versuch, den Server zu starten, wenn bereits eine andere Instanz des Apache oder ein anderer Webserver an den gleichen Port gebunden ist.

Für weitere Anleitungen zur Fehlerbehebung lesen Sie bitte die Apache-[FAQ](#).



## Beim Bootvorgang starten

Wenn Sie möchten, dass Ihr Server direkt nach einem System-Neustart weiterläuft, sollten Sie einen Aufruf von [apachectl](#) zu den Startdateien Ihres Systems hinzufügen (üblicherweise `rc.local` oder eine Datei in einem `rc.N`-Verzeichnis). Dies startet den Apache als `root`. Stellen Sie zuvor jedoch sicher, dass Ihr Server hinsichtlich Sicherheit und Zugriffsbeschränkungen richtig konfiguriert ist.

Das [apachectl](#)-Skript ist dafür ausgelegt, wie ein Standard-SysV-init-Skript zu arbeiten. Es akzeptiert die Argumente `start`, `restart` und `stop` und übersetzt sie in die entsprechenden Signale für [httpd](#). Daher können Sie oftmals einfach [apachectl](#) in das entsprechende init-Verzeichnis linken. Überprüfen Sie bitte auf jeden Fall die genauen Anforderungen Ihres Systems.

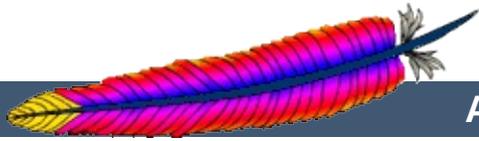


Weitere Informationen über Kommandozeilenoptionen von [httpd](#) und [apachectl](#) sowie anderen Hilfsprogrammen, die dem Server beigefügt sind, sind auf der Seite [Server und Hilfsprogramme](#) verfügbar. Es existiert außerdem eine Dokumentation aller in der Apache-Distribution enthaltenen [Module](#) und der von ihnen bereitgestellten [Direktiven](#).

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

## Beenden und Neustarten

Diese Übersetzung ist möglicherweise nicht mehr aktuell. Bitte prüfen Sie die englische Version auf die neuesten Änderungen.

Dieses Dokument umfasst das Beenden und Neustarten des Apache auf Unix-ähnlichen Systemen. Anwender von Windows NT, 2000 und XP sollten [Betreiben des Apache als Dienst](#) lesen, während hingegen Anwender von Windows 9x sowie ME [Betreiben des Apache als Konsolenanwendung](#) lesen sollten, um mehr Informationen zur Handhabung des Apache auf diesen Systemen zu erhalten.

### Siehe auch

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



Um den Apache zu stoppen oder neu zu starten, müssen Sie ein Signal an den laufenden [httpd](#)-Prozess senden. Es gibt zwei Möglichkeiten, diese Signale zu senden. Zum einen können Sie den Unix-Befehl `kill` verwenden, um den Prozessen direkt Signale zu senden. Sie werden feststellen, dass auf Ihrem System mehrere [httpd](#)-Programme laufen. Sie sollten jedoch nicht jedem dieser Prozesse ein Signal senden, sondern nur dem Elternprozess, dessen PID im [PidFile](#) steht. Das heißt, Sie sollten es niemals nötig haben, einem anderen Prozess, als dem Elternprozess, ein Signal zu senden. Es gibt drei Signale, die Sie an den Elternprozess senden können: [TERM](#), [HUP](#) und [USR1](#), die nachfolgend beschrieben werden.

Um dem Elternprozess ein Signal zu senden, verwenden Sie einen Befehl wie z.B.:

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

Die zweite Methode, dem [httpd](#)-Prozess zu signalisieren, ist die Verwendung der `-k`-Befehlszeilenoptionen `stop`, `restart` und `graceful`, wie unten beschrieben. Dies sind Argumente des [httpd](#)-Programms, es wird jedoch empfohlen, sie unter Verwendung des Steuerskripts [apachectl](#) zu senden, welches diese an [httpd](#) durchreicht.

Nachdem Sie [httpd](#) signalisiert haben, können Sie dessen Fortschritt beobachten, indem Sie eingeben:

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

Passen Sie diese Beispiele entsprechend Ihren [ServerRoot](#)- und [PidFile](#)-Einstellungen an.



---

## Signal: TERM

```
apachectl -k stop
```

Das Senden des TERM- oder stop-Signals an den Elternprozess veranlasst diesen, sofort zu versuchen, alle seine Kindprozesse zu beenden. Es kann einige Sekunden dauern, bis alle Kindprozesse komplett beendet sind. Danach beendet sich der Elternprozess selbst. Alle gerade bearbeiteten Anfragen werden abgebrochen. Es werden keine weiteren Anfragen mehr bedient.



## Signal: USR1

```
apachectl -k graceful
```

Das USR1- oder graceful-Signal veranlasst den Elternprozess, die Kinder *anzuweisen*, sich nach Abschluß ihrer momentanen bearbeiteten Anfrage zu beenden (oder sich sofort zu beenden, wenn sie gerade keine Anfrage bedienen). Der Elternprozess liest seine Konfigurationsdateien erneut ein und öffnet seine Logdateien neu. Wenn ein Kindprozess stirbt, ersetzt der Elternprozess ihn durch ein Kind der neuen Konfigurations-*Generation*. Dieses beginnt sofort damit, neue Anfragen zu bedienen.

Auf bestimmten Plattformen, welche kein USR1 für einen unterbrechungsfreien Neustart erlauben, kann ein alternatives Signal verwendet werden (wie z.B. WINCH). Der Befehl `apachectl graceful` sendet das jeweils richtige Signal für Ihre Plattform.

Der Code ist dafür ausgelegt, stets die MPM-Direktiven zur Prozesssteuerung zu beachten, so dass die Anzahl der Prozesse und Threads, die zur Bedienung der Clients bereitstehen, während des Neustarts auf die entsprechenden Werte gesetzt werden. Weiterhin wird [StartServers](#) auf folgende Art und Weise interpretiert: Wenn nach einer Sekunde nicht mindestens [StartServers](#) neue Kindprozesse erstellt wurden, dann werden, um den Durchsatz zu beschleunigen, entsprechend weitere erstellt. Auf diese Weise versucht der Code sowohl die Anzahl der Kinder entsprechend der Serverlast anzupassen als auch Ihre Wünsche hinsichtlich des Parameters [StartServers](#) zu berücksichtigen.

Benutzer von [mod\\_status](#) werden feststellen, dass die

Serverstatistiken **nicht** auf Null zurückgesetzt werden, wenn ein USR1 gesendet wurde. Der Code wurde so geschrieben, dass sowohl die Zeit minimiert wird, in der der Server nicht in der Lage ist, neue Anfragen zu bedienen (diese werden vom Betriebssystem in eine Warteschlange gestellt, so dass sie auf keinen Fall verloren gehen) als auch Ihre Parameter zur Feinabstimmung berücksichtigt werden. Um dies zu erreichen, muss die *Statustabelle* (Scoreboard), die dazu verwendet wird, alle Kinder über mehrere Generationen zu verfolgen, erhalten bleiben.

Das Statusmodul benutzt außerdem ein G, um diejenigen Kinder zu kennzeichnen, die noch immer Anfragen bedienen, welche gestartet wurden, bevor ein unterbrechungsfreier Neustart veranlaßt wurde.

Derzeit gibt es keine Möglichkeit für ein Log-Rotationsskript, das USR1 verwendet, sicher festzustellen, dass alle Kinder, die in ein vor dem Neustart geöffnetes Log schreiben, beendet sind. Wir schlagen vor, dass Sie nach dem Senden des Signals USR1 eine angemessene Zeitspanne warten, bevor Sie das alte Log anfassen. Wenn beispielsweise die meisten Ihrer Zugriffe bei Benutzern mit niedriger Bandbreite weniger als 10 Minuten für eine vollständige Antwort benötigen, dann könnten Sie 15 Minuten warten, bevor Sie auf das alte Log zugreifen.

Wenn Ihre Konfigurationsdatei Fehler enthält, während Sie einen Neustart anweisen, dann wird Ihr Elternprozess nicht neu starten, sondern sich mit einem Fehler beenden. Im Falle eines unterbrechungsfreien Neustarts läßt er die Kinder weiterlaufen, wenn er sich beendet. (Dies sind die Kinder, die sich "sanft beenden", indem sie ihre letzte Anfrage erledigen.) Das verursacht Probleme, wenn Sie versuchen, den Server neu zu starten -- er ist nicht in der Lage, sich an die Ports zu binden, an denen er lauschen soll. Bevor Sie einen Neustart durchführen,

können Sie die Syntax der Konfigurationsdateien mit dem Befehlszeilenargument `-t` überprüfen (siehe auch [httpd](#)). Das garantiert allerdings nicht, dass der Server korrekt starten wird. Um sowohl die Syntax als auch die Semantik der Konfigurationsdateien zu prüfen, können Sie versuchen, [httpd](#) als nicht-root-Benutzer zu starten. Wenn dabei keine Fehler auftreten, wird er versuchen, seine Sockets und Logdateien zu öffnen und fehlschlagen, da er nicht root ist (oder weil sich der gegenwärtig laufende [httpd](#) bereits diese Ports gebunden hat). Wenn er aus einem anderen Grund fehlschlägt, dann liegt wahrscheinlich ein Konfigurationsfehler vor. Der Fehler sollte behoben werden, bevor der unterbrechungsfreie Neustart angewiesen wird.



## Signal: HUP

```
apachectl -k restart
```

Das Senden des Signals HUP oder `restart` veranlaßt den Elternprozess, wie bei TERM alle seine Kinder zu beenden. Der Elternprozess beendet sich jedoch nicht. Er liest seine Konfigurationsdateien neu ein und öffnet alle Logdateien erneut. Dann erzeugt er einen neuen Satz Kindprozesse und setzt die Bedienung von Zugriffen fort.

Benutzer von `mod_status` werden feststellen, dass die Serverstatistiken auf Null gesetzt werden, wenn ein HUP gesendet wurde.

Wenn Ihre Konfigurationsdatei einen Fehler enthält, während Sie einen Neustart anweisen, dann wird Ihr Elternprozess nicht neu starten, sondern sich mit einem Fehler beenden. Lesen Sie oben, wie Sie das vermeiden können.



Vor der Version 1.2b9 des Apache existierten verschiedene *Wettkampfsituationen* (race conditions), die den Neustart und die Signale beeinflusst haben. (Einfach erklärt ist eine Wettkampfsituation ein zeitabhängiges Problem - wenn etwas zum falschen Zeitpunkt erfolgt oder Dinge in der falschen Reihenfolge passieren, ist unerwartetes Verhalten die Folge. Wenn die gleichen Dinge zur richtigen Zeit geschehen, funktioniert alles korrekt.) Bei Architekturen mit dem "richtigen" Funktionsumfang haben wir so viele eliminiert wie wir nur konnten. Dennoch sollte beachtet werden, dass noch immer Wettkampfsituationen auf bestimmten Architekturen existieren.

Bei Architekturen, die ein [ScoreBoardFile](#) auf Platte verwenden, besteht die Gefahr, dass die Statustabelle beschädigt wird. Das kann zu "bind: Address already in use" ("bind: Adresse wird bereits verwendet", nach einem HUP) oder "long lost child came home!" ("Der verlorene Sohn ist heimgekehrt", nach einem USR1) führen. Ersteres ist ein schwerer Fehler, während letzteres lediglich bewirkt, dass der Server einen Eintrag in der Statustabelle verliert. So kann es ratsam sein, unterbrechungsfreie Neustarts zusammen mit einem gelegentlichen harten Neustart zu verwenden. Diese Probleme lassen sich nur sehr schwer umgehen, aber glücklicherweise benötigen die meisten Architekturen keine Statustabelle in Form einer Datei. Bitte lesen Sie für Architekturen, die sie benötigen, die Dokumentation zu [ScoreBoardFile](#).

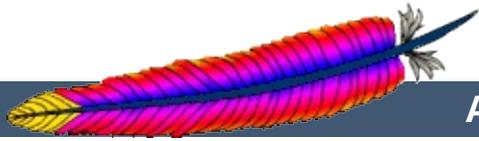
Alle Architekturen haben in jedem Kindprozess eine kleine Wettkampfsituation, welche die zweite und nachfolgende Anfragen einer persistenten HTTP-Verbindung (KeepAlive) umfaßt. Der Prozess kann nach dem Lesen der Anfragezeile aber vor dem Lesen der Anfrage-Header enden. Es existiert eine Korrektur, die für 1.2 zu spät kam. Theoretisch sollte das kein Problem

darstellen, da der KeepAlive-Client derartige Ereignisse aufgrund von Netzwerk-Latenzzeiten und Auszeiten des Servers erwarten sollte. In der Praxis scheint keiner von beiden beeinflußt zu werden -- in einem Testfall wurde der Server zwanzig mal pro Sekunde neu gestartet, während Clients das Angebot abgegrast haben, ohne kaputte Bilder oder leere Dokumente zu erhalten.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

# Konfigurationsdateien

Diese Übersetzung ist möglicherweise nicht mehr aktuell. Bitte prüfen Sie die englische Version auf die neuesten Änderungen.

Dieses Dokument beschreibt die Dateien, die zur Konfiguration des Apache HTTP Servers verwendet werden.



## Hauptkonfigurationsdateien

Referenzierte Module	Referenzierte Direktiven
<u>mod_mime</u>	<u>&lt;IfDefine&gt;</u> <u>Include</u> <u>TypesConfig</u>

Der Apache wird konfiguriert, indem [Direktiven](#) in einfache Textdateien eingetragen werden. Die Hauptkonfigurationsdatei heißt üblicherweise `httpd.conf`. Der Ablageort dieser Datei wird bei der Kompilierung festgelegt, kann jedoch mit der Befehlszeilenoption `-f` überschrieben werden. Durch Verwendung der Direktive [Include](#) können außerdem weitere Konfigurationsdateien hinzugefügt werden. Zum Einfügen von mehreren Konfigurationsdateien können Platzhalter verwendet werden. Jede Direktive darf in jeder dieser Konfigurationsdateien angegeben werden. Änderungen in den Hauptkonfigurationsdateien werden vom Apache nur beim Start oder Neustart erkannt.

Der Server liest auch eine Datei mit MIME-Dokumenttypen ein. Der Name dieser Datei wird durch die Direktive [TypesConfig](#) bestimmt. Die Voreinstellung ist `mime.types`.



## Syntax der Konfigurationsdateien

Die Konfigurationsdateien des Apache enthalten eine Direktive pro Zeile. Der Backslash "\ " läßt sich als letztes Zeichen in einer Zeile dazu verwenden, die Fortsetzung der Direktive in der nächsten Zeile anzuzeigen. Es darf kein weiteres Zeichen oder Whitespace zwischen dem Backslash und dem Zeilenende folgen.

In den Konfigurationsdateien wird bei den Direktiven nicht zwischen Groß- und Kleinschreibung unterschieden. Bei den Argumenten der Direktiven wird dagegen oftmals zwischen Groß- und Kleinschreibung differenziert. Zeilen, die mit dem Doppelkreuz "# " beginnen, werden als Kommentare betrachtet und ignoriert. Kommentare dürfen **nicht** am Ende einer Zeile nach der Direktive eingefügt werden. Leerzeilen und Whitespaces vor einer Direktive werden ignoriert. Dadurch lassen sich Direktiven zur besseren Lesbarkeit einrücken.

Sie können die Syntax Ihrer Konfigurationsdateien auf Fehler prüfen, ohne den Server zu starten, indem Sie `apachectl configtest` oder die Befehlszeilenoption `-t` verwenden.



## Module

Referenzierte Module	Referenzierte Direktiven
<code>mod_so</code>	<code>&lt;IfModule&gt;</code> <code>LoadModule</code>

Der Apache ist ein modularer Server. Das bedeutet, dass nur die absolute Grundfunktionalität im Kernserver enthalten ist.

Weitergehende Fähigkeiten sind mittels [Modulen](#) verfügbar, die in den Apache geladen werden können. Standardmäßig wird bei der Kompilierung ein Satz von Basismodulen (*Anm.d.Ü.:* die sogenannten [Base-Module](#)) in den Server eingebunden. Wenn der Server für die Verwendung von [dynamisch ladbaren](#) Modulen kompiliert wurde, dann können Module separat kompiliert und jederzeit mittels der Direktive `LoadModule` hinzugefügt werden. Andernfalls muss der Apache neu kompiliert werden, um Module hinzuzufügen oder zu entfernen. Konfigurationsanweisungen können abhängig vom Vorhandensein eines bestimmten Moduls eingesetzt werden, indem sie in einen `<IfModule>`-Block eingeschlossen werden.

Um zu sehen, welche Module momentan in den Server einkompiliert sind, kann die Befehlszeilenoption `-l` verwendet werden.



## 2.1.1.1 Konfiguration der Direktiven

Referenzierte Module	Referenzierte Direktiven
	<a href="#"><u>&lt;Directory&gt;</u></a>
	<a href="#"><u>&lt;DirectoryMatch&gt;</u></a>
	<a href="#"><u>&lt;Files&gt;</u></a>
	<a href="#"><u>&lt;FilesMatch&gt;</u></a>
	<a href="#"><u>&lt;Location&gt;</u></a>
	<a href="#"><u>&lt;LocationMatch&gt;</u></a>
	<a href="#"><u>&lt;VirtualHost&gt;</u></a>

Direktiven in den Hauptkonfigurationsdateien gelten für den gesamten Server. Wenn Sie die Konfiguration nur für einen Teil des Servers verändern möchten, können Sie den Gültigkeitsbereich der Direktiven beschränken, indem Sie diese in [<Directory>](#)-, [<DirectoryMatch>](#)-, [<Files>](#)-, [<FilesMatch>](#)-, [<Location>](#)- oder [<LocationMatch>](#)-Abschnitte eingefügen. Diese Abschnitte begrenzen die Anwendung der umschlossenen Direktiven auf bestimmte Pfade des Dateisystems oder auf bestimmte URLs. Sie können für eine fein abgestimmte Konfiguration auch ineinander verschachtelt werden.

Der Apache besitzt die Fähigkeit, mehrere verschiedene Websites gleichzeitig zu bedienen. Dies wird [virtuelles Hosten](#) genannt. Direktiven können auch in ihrem Gültigkeitsbereich eingeschränkt werden, indem sie innerhalb eines [<VirtualHost>](#)-Abschnittes angegeben werden. Sie werden dann nur auf Anfragen für eine bestimmte Website angewendet.

Obwohl die meisten Direktiven in jedem dieser Abschnitte platziert werden können, ergeben einige Direktiven in manchen Kontexten keinen Sinn. Direktiven zur Prozesssteuerung beispielsweise dürfen nur im Kontext des Hauptservers angegeben werden.

Prüfen Sie den [Kontext](#) der Direktive, um herauszufinden, welche Direktiven in welche Abschnitte eingefügt werden können. Weitere Informationen finden Sie unter "[Wie Directory-, Location- und Files-Abschnitte arbeiten](#)".

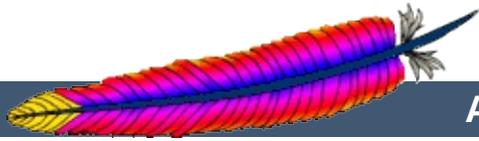


Referenzierte Module	Referenzierte Direktiven
	<a href="#">AccessFileName</a>
	<a href="#">AllowOverride</a>

Der Apache ermöglicht die dezentrale Verwaltung der Konfiguration mittels spezieller Dateien innerhalb des Web-Verzeichnisbaums. Diese speziellen Dateien heißen gewöhnlich `.htaccess`, mit der Direktive [AccessFileName](#) kann jedoch auch ein anderer Name festgelegt werden. In `.htaccess`-Dateien angegebene Direktiven werden auf das Verzeichnis und dessen Unterverzeichnisse angewendet, in dem die Datei abgelegt ist. `.htaccess`-Dateien folgen der gleichen Syntax wie die Hauptkonfigurationsdateien. Da `.htaccess`-Dateien bei jeder Anfrage eingelesen werden, werden Änderungen in diesen Dateien sofort wirksam.

Prüfen Sie den [Kontext](#) der Direktive, um herauszufinden, welche Direktiven in `.htaccess`-Dateien angegeben werden können. Darüber hinaus steuert der Serveradministrator mit der Einstellung der Direktive [AllowOverride](#) in den Hauptkonfigurationsdateien welche Direktiven in `.htaccess`-Dateien verwendet werden dürfen.

Weitere Informationen über `.htaccess`-Dateien finden Sie in der [.htaccess-Einführung](#).



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



## Apache HTTP Server Version 2.0

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

## Configuration Sections

Directives in the [configuration files](#) may apply to the entire server, or they may be restricted to apply only to particular directories, files, hosts, or URLs. This document describes how to use configuration section containers or `.htaccess` files to change the scope of other configuration directives.



## Types of Container and Match Containers

Related Modules	Related Directives
<a href="#">core</a>	<a href="#">&lt;Directory&gt;</a>
<a href="#">mod_proxy</a>	<a href="#">&lt;DirectoryMatch&gt;</a>
	<a href="#">&lt;Files&gt;</a>
	<a href="#">&lt;FilesMatch&gt;</a>
	<a href="#">&lt;IfDefine&gt;</a>
	<a href="#">&lt;IfModule&gt;</a>
	<a href="#">&lt;Location&gt;</a>
	<a href="#">&lt;LocationMatch&gt;</a>
	<a href="#">&lt;Proxy&gt;</a>
	<a href="#">&lt;ProxyMatch&gt;</a>
	<a href="#">&lt;VirtualHost&gt;</a>

There are two basic types of containers. Most containers are evaluated for each request. The enclosed directives are applied only for those requests that match the containers. The [<IfDefine>](#) and [<IfModule>](#) containers, on the other hand, are evaluated only at server startup and restart. If their conditions are true at startup, then the enclosed directives will apply to all requests. If the conditions are not true, the enclosed directives will be ignored.

The [<IfDefine>](#) directive encloses directives that will only be applied if an appropriate parameter is defined on the [httpd](#) command line. For example, with the following configuration, all requests will be redirected to another site only if the server is started using `httpd -DClosedForNow`:

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

The `<IfModule>` directive is very similar, except it encloses directives that will only be applied if a particular module is available in the server. The module must either be statically compiled in the server, or it must be dynamically compiled and its `LoadModule` line must be earlier in the configuration file. This directive should only be used if you need your configuration file to work whether or not certain modules are installed. It should not be used to enclose directives that you want to work all the time, because it can suppress useful error messages about missing modules.

In the following example, the `MimeMagicFiles` directive will be applied only if `mod_mime_magic` is available.

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

Both `<IfDefine>` and `<IfModule>` can apply negative conditions by preceding their test with "!". Also, these sections can be nested to achieve more complex restrictions.



The most commonly used configuration section containers are the ones that change the configuration of particular places in the filesystem or web space. First, it is important to understand the difference between the two. The filesystem is the view of your disks as seen by your operating system. For example, in a default install, Apache resides at `/usr/local/apache2` in the Unix filesystem or `"c:/Program Files/Apache Group/Apache2"` in the Windows filesystem. (Note that forward slashes should always be used as the path separator in Apache, even for Windows.) In contrast, the web space is the view of your site as delivered by the web server and seen by the client. So the path `/dir/` in the web space corresponds to the path `/usr/local/apache2/htdocs/dir/` in the filesystem of a default Apache install on Unix. The web space need not map directly to the filesystem, since webpages may be generated dynamically from databases or other locations.

## Filesystem Containers

The `<Directory>` and `<Files>` directives, along with their regex counterparts, apply directives to parts of the filesystem. Directives enclosed in a `<Directory>` section apply to the named filesystem directory and all subdirectories of that directory. The same effect can be obtained using [.htaccess files](#). For example, in the following configuration, directory indexes will be enabled for the `/var/web/dir1` directory and all subdirectories.

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

Directives enclosed in a `<Files>` section apply to any file with the specified name, regardless of what directory it lies in. So for example, the following configuration directives will, when placed in

the main section of the configuration file, deny access to any file named `private.html` regardless of where it is found.

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

To address files found in a particular part of the filesystem, the `<Files>` and `<Directory>` sections can be combined. For example, the following configuration will deny access to `/var/web/dir1/private.html`, `/var/web/dir1/subdir2/private.html`, `/var/web/dir1/subdir3/private.html`, and any other instance of `private.html` found under the `/var/web/dir1/` directory.

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

## Webpace Containers

The `<Location>` directive and its regex counterpart, on the other hand, change the configuration for content in the webpace. For example, the following configuration prevents access to any URL-path that begins in `/private`. In particular, it will apply to requests for `http://yoursite.example.com/private`, `http://yoursite.example.com/private123`, and `http://yoursite.example.com/private/dir/file.html` as well as any other requests starting with the `/private` string.

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

```
Deny from all
</Location>
```

The `<Location>` directive need not have anything to do with the filesystem. For example, the following example shows how to map a particular URL to an internal Apache handler provided by `mod_status`. No file called `server-status` needs to exist in the filesystem.

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

## Wildcards and Regular Expressions

The `<Directory>`, `<Files>`, and `<Location>` directives can each use shell-style wildcard characters as in `fnmatch` from the C standard library. The character `"*` matches any sequence of characters, `"?"` matches any single character, and `"[seq]"` matches any character in `seq`. The `"/"` character will not be matched by any wildcard; it must be specified explicitly.

If even more flexible matching is required, each container has a regular-expression (regex) counterpart `<DirectoryMatch>`, `<FilesMatch>`, and `<LocationMatch>` that allow perl-compatible [regular expressions](#) to be used in choosing the matches. But see the section below on configuration merging to find out how using regex sections will change how directives are applied.

A non-regex wildcard section that changes the configuration of all user directories could look as follows:

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

Using regex sections, we can deny access to many types of image files at once:

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

## What to use When

Choosing between filesystem containers and web space containers is actually quite easy. When applying directives to objects that reside in the filesystem always use [<Directory>](#) or [<Files>](#). When applying directives to objects that do not reside in the filesystem (such as a webpage generated from a database), use [<Location>](#).

It is important to never use [<Location>](#) when trying to restrict access to objects in the filesystem. This is because many different web space locations (URLs) could map to the same filesystem location, allowing your restrictions to be circumvented. For example, consider the following configuration:

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

This works fine if the request is for `http://yoursite.example.com/dir/`. But what if you are on a case-insensitive filesystem? Then your restriction could be easily circumvented by requesting `http://yoursite.example.com/DIR/`. The [<Directory>](#) directive, in contrast, will apply to any content served from that location, regardless of how it is called. (An exception is filesystem links. The same directory can be placed in more than one part of

the filesystem using symbolic links. The `<Directory>` directive will follow the symbolic link without resetting the pathname. Therefore, for the highest level of security, symbolic links should be disabled with the appropriate `Options` directive.)

If you are, perhaps, thinking that none of this applies to you because you use a case-sensitive filesystem, remember that there are many other ways to map multiple webspace locations to the same filesystem location. Therefore you should always use the filesystem containers when you can. There is, however, one exception to this rule. Putting configuration restrictions in a `<Location />` section is perfectly safe because this section will apply to all requests regardless of the specific URL.



**VIRTUAL HOSTS**

The `<VirtualHost>` container encloses directives that apply to specific hosts. This is useful when serving multiple hosts from the same machine with a different configuration for each. For more information, see the [Virtual Host Documentation](#).



The `<Proxy>` and `<ProxyMatch>` containers apply enclosed configuration directives only to sites accessed through `mod_proxy`'s proxy server that match the specified URL. For example, the following configuration will prevent the proxy server from being used to access the `cnn.com` website.

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



---

To find out what directives are allowed in what types of configuration sections, check the [Context](#) of the directive. Everything that is allowed in [<Directory>](#) sections is also syntactically allowed in [<DirectoryMatch>](#), [<Files>](#), [<FilesMatch>](#), [<Location>](#), [<LocationMatch>](#), [<Proxy>](#), and [<ProxyMatch>](#) sections. There are some exceptions, however:

- The [AllowOverride](#) directive works only in [<Directory>](#) sections.
- The [FollowSymLinks](#) and [SymLinksIfOwnerMatch](#) [Options](#) work only in [<Directory>](#) sections or `.htaccess` files.
- The [Options](#) directive cannot be used in [<Files>](#) and [<FilesMatch>](#) sections.



The configuration sections are applied in a very particular order. Since this can have important effects on how configuration directives are interpreted, it is important to understand how this works.

The order of merging is:

1. `<Directory>` (except regular expressions) and `.htaccess` done simultaneously (with `.htaccess`, if allowed, overriding `<Directory>`)
2. `<DirectoryMatch>` (and `<Directory ~>`)
3. `<Files>` and `<FilesMatch>` done simultaneously
4. `<Location>` and `<LocationMatch>` done simultaneously

Apart from `<Directory>`, each group is processed in the order that they appear in the configuration files. `<Directory>` (group 1 above) is processed in the order shortest directory component to longest. So for example, `<Directory /var/web/dir>` will be processed before `<Directory /var/web/dir/subdir>`. If multiple `<Directory>` sections apply to the same directory they are processed in the configuration file order. Configurations included via the `Include` directive will be treated as if they were inside the including file at the location of the `Include` directive.

Sections inside `<VirtualHost>` sections are applied *after* the corresponding sections outside the virtual host definition. This allows virtual hosts to override the main server configuration.

Later sections override earlier ones.

#### Technical Note

There is actually a `<Location>/<LocationMatch>` sequence

performed just before the name translation phase (where `Aliases` and `DocumentRoots` are used to map URLs to filenames). The results of this sequence are completely thrown away after the translation has completed.

## Some Examples

Below is an artificial example to show the order of merging. Assuming they all apply to the request, the directives in this example will be applied in the order `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>
```

For a more concrete example, consider the following. Regardless of any access restrictions placed in `<Directory>` sections, the `<Location>` section will be evaluated last and will allow unrestricted access to the server. In other words, order of merging is important, so be careful!

```
<Location />
```

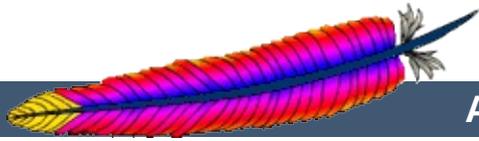
```
Order deny,allow
Allow from all
</Location>

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

## Server-Wide Configuration

This document explains some of the directives provided by the [core](#) server which are used to configure the basic operations of the server.



## SERVER IDENTIFICATION

Related Modules	Related Directives
	<a href="#">ServerName</a>
	<a href="#">ServerAdmin</a>
	<a href="#">ServerSignature</a>
	<a href="#">ServerTokens</a>
	<a href="#">UseCanonicalName</a>

The [ServerAdmin](#) and [ServerTokens](#) directives control what information about the server will be presented in server-generated documents such as error messages. The [ServerTokens](#) directive sets the value of the Server HTTP response header field.

The [ServerName](#) and [UseCanonicalName](#) directives are used by the server to determine how to construct self-referential URLs. For example, when a client requests a directory, but does not include the trailing slash in the directory name, Apache must redirect the client to the full name including the trailing slash so that the client will correctly resolve relative references in the document.



## File Locations

Related Modules	Related Directives
	<a href="#"><u>CoreDumpDirectory</u></a>
	<a href="#"><u>DocumentRoot</u></a>
	<a href="#"><u>ErrorLog</u></a>
	<a href="#"><u>LockFile</u></a>
	<a href="#"><u>PidFile</u></a>
	<a href="#"><u>ScoreBoardFile</u></a>
	<a href="#"><u>ServerRoot</u></a>

These directives control the locations of the various files that Apache needs for proper operation. When the pathname used does not begin with a slash (/), the files are located relative to the [ServerRoot](#). Be careful about locating files in paths which are writable by non-root users. See the [security tips](#) documentation for more details.

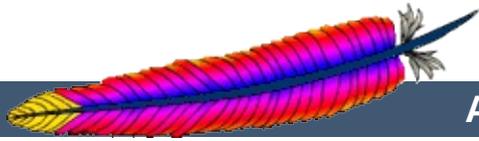


Related Modules	Related Directives
	<a href="#">LimitRequestBody</a>
	<a href="#">LimitRequestFields</a>
	<a href="#">LimitRequestFieldsize</a>
	<a href="#">LimitRequestLine</a>
	<a href="#">RLimitCPU</a>
	<a href="#">RLimitMEM</a>
	<a href="#">RLimitNPROC</a>
	<a href="#">ThreadStackSize</a>

The [LimitRequest](#)\* directives are used to place limits on the amount of resources Apache will use in reading requests from clients. By limiting these values, some kinds of denial of service attacks can be mitigated.

The [RLimit](#)\* directives are used to limit the amount of resources which can be used by processes forked off from the Apache children. In particular, this will control resources used by CGI scripts and SSI exec commands.

The [ThreadStackSize](#) directive is used only on Netware to control the stack size.



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



## Apache HTTP Server Version 2.0

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

## Log Files

In order to effectively manage a web server, it is necessary to get feedback about the activity and performance of the server as well as any problems that may be occurring. The Apache HTTP Server provides very comprehensive and flexible logging capabilities. This document describes how to configure its logging capabilities, and how to understand what the logs contain.



Anyone who can write to the directory where Apache is writing a log file can almost certainly gain access to the uid that the server is started as, which is normally root. Do *NOT* give people write access to the directory the logs are stored in without being aware of the consequences; see the [security tips](#) document for details.

In addition, log files may contain information supplied directly by the client, without escaping. Therefore, it is possible for malicious clients to insert control-characters in the log files, so care must be taken in dealing with raw logs.



Related Modules	Related Directives
	<a href="#">ErrorLog</a>
	<a href="#">LogLevel</a>

The server error log, whose name and location is set by the [ErrorLog](#) directive, is the most important log file. This is the place where Apache httpd will send diagnostic information and record any errors that it encounters in processing requests. It is the first place to look when a problem occurs with starting the server or with the operation of the server, since it will often contain details of what went wrong and how to fix it.

The error log is usually written to a file (typically `error_log` on Unix systems and `error.log` on Windows and OS/2). On Unix systems it is also possible to have the server send errors to `syslog` or [pipe them to a program](#).

The format of the error log is relatively free-form and descriptive. But there is certain information that is contained in most error log entries. For example, here is a typical message.

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

The first item in the log entry is the date and time of the message. The second item lists the severity of the error being reported. The [LogLevel](#) directive is used to control the types of errors that are sent to the error log by restricting the severity level. The third item gives the IP address of the client that generated the error. Beyond that is the message itself, which in this case indicates that the server has been configured to deny the client access. The server reports the file-system path (as opposed to the web path) of the

requested document.

A very wide variety of different messages can appear in the error log. Most look similar to the example above. The error log will also contain debugging output from CGI scripts. Any information written to `stderr` by a CGI script will be copied directly to the error log.

It is not possible to customize the error log by adding or removing information. However, error log entries dealing with particular requests have corresponding entries in the [access log](#). For example, the above example entry corresponds to an access log entry with status code 403. Since it is possible to customize the access log, you can obtain more information about error conditions using that log file.

During testing, it is often useful to continuously monitor the error log for any problems. On Unix systems, you can accomplish this using:

```
tail -f error_log
```



## Access Log

Related Modules	Related Directives
<a href="#">mod_log_config</a>	<a href="#">CustomLog</a>
<a href="#">mod_setenvif</a>	<a href="#">LogFormat</a>
	<a href="#">SetEnvIf</a>

The server access log records all requests processed by the server. The location and content of the access log are controlled by the [CustomLog](#) directive. The [LogFormat](#) directive can be used to simplify the selection of the contents of the logs. This section describes how to configure the server to record information in the access log.

Of course, storing the information in the access log is only the start of log management. The next step is to analyze this information to produce useful statistics. Log analysis in general is beyond the scope of this document, and not really part of the job of the web server itself. For more information about this topic, and for applications which perform log analysis, check the [Open Directory](#) or [Yahoo](#).

Various versions of Apache httpd have used other modules and directives to control access logging, including `mod_log_referer`, `mod_log_agent`, and the `TransferLog` directive. The [CustomLog](#) directive now subsumes the functionality of all the older directives.

The format of the access log is highly configurable. The format is specified using a format string that looks much like a C-style `printf(1)` format string. Some examples are presented in the next sections. For a complete list of the possible contents of the format string, see the [mod\\_log\\_config format strings](#).

## Common Log Format

A typical configuration for the access log might look as follows.

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

This defines the *nickname* common and associates it with a particular log format string. The format string consists of percent directives, each of which tell the server to log a particular piece of information. Literal characters may also be placed in the format string and will be copied directly into the log output. The quote character (") must be escaped by placing a backslash before it to prevent it from being interpreted as the end of the format string. The format string may also contain the special control characters "\n" for new-line and "\t" for tab.

The [CustomLog](#) directive sets up a new log file using the defined *nickname*. The filename for the access log is relative to the [ServerRoot](#) unless it begins with a slash.

The above configuration will write log entries in a format known as the Common Log Format (CLF). This standard format can be produced by many different web servers and read by many log analysis programs. The log file entries produced in CLF will look something like this:

```
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET
/apache_pb.gif HTTP/1.0" 200 2326
```

Each part of this log entry is described below.

### **127.0.0.1 (%h)**

This is the IP address of the client (remote host) which made the request to the server. If [HostnameLookups](#) is set to On, then the server will try to determine the hostname and log it in place of the IP address. However, this configuration is not

recommended since it can significantly slow the server. Instead, it is best to use a log post-processor such as [logresolve](#) to determine the hostnames. The IP address reported here is not necessarily the address of the machine at which the user is sitting. If a proxy server exists between the user and the server, this address will be the address of the proxy, rather than the originating machine.

#### - (%1)

The "hyphen" in the output indicates that the requested piece of information is not available. In this case, the information that is not available is the RFC 1413 identity of the client determined by `identd` on the clients machine. This information is highly unreliable and should almost never be used except on tightly controlled internal networks. Apache `httpd` will not even attempt to determine this information unless `IdentityCheck` is set to `On`.

#### frank (%u)

This is the userid of the person requesting the document as determined by HTTP authentication. The same value is typically provided to CGI scripts in the `REMOTE_USER` environment variable. If the status code for the request (see below) is 401, then this value should not be trusted because the user is not yet authenticated. If the document is not password protected, this part will be "-" just like the previous one.

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

The time that the request was received. The format is:

```
[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
```

It is possible to have the time displayed in another format by specifying `%{format}t` in the log format string, where `format` is as in `strftime(3)` from the C standard library.

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

The request line from the client is given in double quotes. The request line contains a great deal of useful information. First, the method used by the client is GET. Second, the client requested the resource `/apache_pb.gif`, and third, the client used the protocol `HTTP/1.0`. It is also possible to log one or more parts of the request line independently. For example, the format string `"%m %U%q %H"` will log the method, path, query-string, and protocol, resulting in exactly the same output as `"%r"`.

### **200 (%>s)**

This is the status code that the server sends back to the client. This information is very valuable, because it reveals whether the request resulted in a successful response (codes beginning in 2), a redirection (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5). The full list of possible status codes can be found in the [HTTP specification](#) (RFC2616 section 10).

### **2326 (%b)**

The last part indicates the size of the object returned to the client, not including the response headers. If no content was returned to the client, this value will be `"-"`. To log `"0"` for no content, use `%B` instead.

## Combined Log Format

Another commonly used format string is called the Combined Log Format. It can be used as follows.

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

This format is exactly the same as the Common Log Format, with the addition of two more fields. Each of the additional fields uses the percent-directive `%{header}i`, where *header* can be any HTTP request header. The access log under this format will look like:

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

The additional fields are:

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

The "Referer" (sic) HTTP request header. This gives the site that the client reports having been referred from. (This should be the page that links to or includes `/apache_pb.gif`).

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

The User-Agent HTTP request header. This is the identifying information that the client browser reports about itself.

## Multiple Access Logs

Multiple access logs can be created simply by specifying multiple [CustomLog](#) directives in the configuration file. For example, the

following directives will create three access logs. The first contains the basic CLF information, while the second and third contain referer and browser information. The last two [CustomLog](#) lines show how to mimic the effects of the ReferLog and AgentLog directives.

```
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"
```

This example also shows that it is not necessary to define a nickname with the [LogFormat](#) directive. Instead, the log format can be specified directly in the [CustomLog](#) directive.

## Conditional Logs

There are times when it is convenient to exclude certain entries from the access logs based on characteristics of the client request. This is easily accomplished with the help of [environment variables](#). First, an environment variable must be set to indicate that the request meets certain conditions. This is usually accomplished with [SetEnvIf](#). Then the env= clause of the [CustomLog](#) directive is used to include or exclude requests where the environment variable is set. Some examples:

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

As another example, consider logging requests from english-speakers to one log file, and non-english speakers to a different log file.

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

Although we have just shown that conditional logging is very powerful and flexible, it is not the only way to control the contents of the logs. Log files are more useful when they contain a complete record of server activity. It is often easier to simply post-process the log files to remove requests that you do not want to consider.



## Log Rotation

On even a moderately busy server, the quantity of information stored in the log files is very large. The access log file typically grows 1 MB or more per 10,000 requests. It will consequently be necessary to periodically rotate the log files by moving or deleting the existing logs. This cannot be done while the server is running, because Apache will continue writing to the old log file as long as it holds the file open. Instead, the server must be [restarted](#) after the log files are moved or deleted so that it will open new log files.

By using a *graceful* restart, the server can be instructed to open new log files without losing any existing or pending connections from clients. However, in order to accomplish this, the server must continue to write to the old log files while it finishes serving old requests. It is therefore necessary to wait for some time after the restart before doing any processing on the log files. A typical scenario that simply rotates the logs and compresses the old logs to save space is:

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

Another way to perform log rotation is using [piped logs](#) as discussed in the next section.



## Piped Logs

Apache httpd is capable of writing error and access log files through a pipe to another process, rather than directly to a file. This capability dramatically increases the flexibility of logging, without adding code to the main server. In order to write logs to a pipe, simply replace the filename with the pipe character "|", followed by the name of the executable which should accept log entries on its standard input. Apache will start the piped-log process when the server starts, and will restart it if it crashes while the server is running. (This last feature is why we can refer to this technique as "reliable piped logging".)

Piped log processes are spawned by the parent Apache httpd process, and inherit the userid of that process. This means that piped log programs usually run as root. It is therefore very important to keep the programs simple and secure.

One important use of piped logs is to allow log rotation without having to restart the server. The Apache HTTP Server includes a simple program called [rotatelogs](#) for this purpose. For example, to rotate the logs every 24 hours, you can use:

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

Notice that quotes are used to enclose the entire command that will be called for the pipe. Although these examples are for the access log, the same technique can be used for the error log.

A similar but much more flexible log rotation program called [cronolog](#) is available at an external site.

As with conditional logging, piped logs are a very powerful tool, but they should not be used where a simpler solution like off-line post-processing is available.



When running a server with many [virtual hosts](#), there are several options for dealing with log files. First, it is possible to use logs exactly as in a single-host server. Simply by placing the logging directives outside the `<VirtualHost>` sections in the main server context, it is possible to log all requests in the same access log and error log. This technique does not allow for easy collection of statistics on individual virtual hosts.

If `CustomLog` or `ErrorLog` directives are placed inside a `<VirtualHost>` section, all requests or errors for that virtual host will be logged only to the specified file. Any virtual host which does not have logging directives will still have its requests sent to the main server logs. This technique is very useful for a small number of virtual hosts, but if the number of hosts is very large, it can be complicated to manage. In addition, it can often create problems with [insufficient file descriptors](#).

For the access log, there is a very good compromise. By adding information on the virtual host to the log format string, it is possible to log all hosts to the same log, and later split the log into individual files. For example, consider the following directives.

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

The `%v` is used to log the name of the virtual host that is serving the request. Then a program like [split-logfile](#) can be used to post-process the access log in order to split it into one file per virtual host.



## ScriptLogFiles

Related Modules	Related Directives
<a href="#">mod_cgi</a>	<a href="#">PidFile</a>
<a href="#">mod_rewrite</a>	<a href="#">RewriteLog</a>
	<a href="#">RewriteLogLevel</a>
	<a href="#">ScriptLog</a>
	<a href="#">ScriptLogBuffer</a>
	<a href="#">ScriptLogLength</a>

### PID File

On startup, Apache httpd saves the process id of the parent httpd process to the file `logs/httpd.pid`. This filename can be changed with the [PidFile](#) directive. The process-id is for use by the administrator in restarting and terminating the daemon by sending signals to the parent process; on Windows, use the `-k` command line option instead. For more information see the [Stopping and Restarting](#) page.

### Script Log

In order to aid in debugging, the [ScriptLog](#) directive allows you to record the input to and output from CGI scripts. This should only be used in testing - not for live servers. More information is available in the [mod\\_cgi](#) documentation.

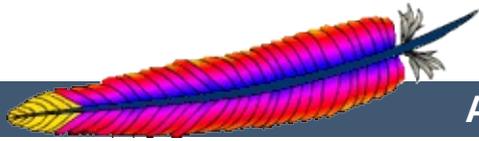
### Rewrite Log

When using the powerful and complex features of [mod\\_rewrite](#), it is almost always necessary to use the [RewriteLog](#) to help in debugging. This log file produces a detailed analysis of how the rewriting engine transforms requests. The level of detail is controlled by the [RewriteLogLevel](#) directive.

---

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

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



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



## Apache HTTP Server Version 2.0

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

## Mapping URLs to Filesystem Locations

This document explains how Apache uses the URL of a request to determine the filesystem location from which to serve a file.



Related Modules	Related Directives
<a href="#">mod_alias</a>	<a href="#">Alias</a>
<a href="#">mod_proxy</a>	<a href="#">AliasMatch</a>
<a href="#">mod_rewrite</a>	<a href="#">CheckSpelling</a>
<a href="#">mod_userdir</a>	<a href="#">DocumentRoot</a>
<a href="#">mod_speling</a>	<a href="#">ErrorDocument</a>
<a href="#">mod_vhost_alias</a>	<a href="#">Options</a>
	<a href="#">ProxyPass</a>
	<a href="#">ProxyPassReverse</a>
	<a href="#">Redirect</a>
	<a href="#">RedirectMatch</a>
	<a href="#">RewriteCond</a>
	<a href="#">RewriteMatch</a>
	<a href="#">ScriptAlias</a>
	<a href="#">ScriptAliasMatch</a>
	<a href="#">UserDir</a>



## DocumentRoot

In deciding what file to serve for a given request, Apache's default behavior is to take the URL-Path for the request (the part of the URL following the hostname and port) and add it to the end of the [DocumentRoot](#) specified in your configuration files. Therefore, the files and directories underneath the [DocumentRoot](#) make up the basic document tree which will be visible from the web.

Apache is also capable of [Virtual Hosting](#), where the server receives requests for more than one host. In this case, a different [DocumentRoot](#) can be specified for each virtual host, or alternatively, the directives provided by the module [mod\\_vhost\\_alias](#) can be used to dynamically determine the appropriate place from which to serve content based on the requested IP address or hostname.



There are frequently circumstances where it is necessary to allow web access to parts of the filesystem that are not strictly underneath the `DocumentRoot`. Apache offers several different ways to accomplish this. On Unix systems, symbolic links can bring other parts of the filesystem under the `DocumentRoot`. For security reasons, Apache will follow symbolic links only if the `Options` setting for the relevant directory includes `FollowSymLinks` or `SymLinksIfOwnerMatch`.

Alternatively, the `Alias` directive will map any part of the filesystem into the web space. For example, with

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

the URL `http://www.example.com/docs/dir/file.html` will be served from `/var/web/dir/file.html`. The `ScriptAlias` directive works the same way, with the additional effect that all content located at the target path is treated as CGI scripts.

For situations where you require additional flexibility, you can use the `AliasMatch` and `ScriptAliasMatch` directives to do powerful regular-expression based matching and substitution. For example,

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

will map a request to `http://example.com/~user/cgi-bin/script.cgi` to the path `/home/user/cgi-bin/script.cgi` and will treat the resulting file as a CGI script.



Traditionally on Unix systems, the home directory of a particular *user* can be referred to as `~user/`. The module `mod_userdir` extends this idea to the web by allowing files under each user's home directory to be accessed using URLs such as the following.

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

For security reasons, it is inappropriate to give direct access to a user's home directory from the web. Therefore, the `UserDir` directive specifies a directory underneath the user's home directory where web files are located. Using the default setting of `Userdir public_html`, the above URL maps to a file at a directory like `/home/user/public_html/file.html` where `/home/user/` is the user's home directory as specified in `/etc/passwd`.

There are also several other forms of the `Userdir` directive which you can use on systems where `/etc/passwd` does not contain the location of the home directory.

Some people find the "~" symbol (which is often encoded on the web as `%7e`) to be awkward and prefer to use an alternate string to represent user directories. This functionality is not supported by `mod_userdir`. However, if users' home directories are structured in a regular way, then it is possible to use the `AliasMatch` directive to achieve the desired effect. For example, to make `http://www.example.com/upages/user/file.html` map to `/home/user/public_html/file.html`, use the following `AliasMatch` directive:

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



The configuration directives discussed in the above sections tell Apache to get content from a specific place in the filesystem and return it to the client. Sometimes, it is desirable instead to inform the client that the requested content is located at a different URL, and instruct the client to make a new request with the new URL. This is called *redirection* and is implemented by the `Redirect` directive. For example, if the contents of the directory `/foo/` under the `DocumentRoot` are moved to the new directory `/bar/`, you can instruct clients to request the content at the new location as follows:

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

This will redirect any URL-Path starting in `/foo/` to the same URL path on the `www.example.com` server with `/bar/` substituted for `/foo/`. You can redirect clients to any server, not only the origin server.

Apache also provides a `RedirectMatch` directive for more complicated rewriting problems. For example, to redirect requests for the site home page to a different site, but leave all other requests alone, use the following configuration:

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

Alternatively, to temporarily redirect all pages on one site to a particular page on another site, use the following:

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



## Reverse Proxy

Apache also allows you to bring remote documents into the URL space of the local server. This technique is called *reverse proxying* because the web server acts like a proxy server by fetching the documents from a remote server and returning them to the client. It is different from normal proxying because, to the client, it appears the documents originate at the reverse proxy server.

In the following example, when clients request documents under the `/foo/` directory, the server fetches those documents from the `/bar/` directory on `internal.example.com` and returns them to the client as if they were from the local server.

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

The `ProxyPass` configures the server to fetch the appropriate documents, while the `ProxyPassReverse` directive rewrites redirects originating at `internal.example.com` so that they target the appropriate directory on the local server. It is important to note, however, that links inside the documents will not be rewritten. So any absolute links on `internal.example.com` will result in the client breaking out of the proxy server and requesting directly from `internal.example.com`.



## URL Rewriting Engine

When even more powerful substitution is required, the rewriting engine provided by `mod_rewrite` can be useful. The directives provided by this module use characteristics of the request such as browser type or source IP address in deciding from where to serve content. In addition, `mod_rewrite` can use external database files or programs to determine how to handle a request. The rewriting engine is capable of performing all three types of mappings discussed above: internal redirects (aliases), external redirects, and proxying. Many practical examples employing `mod_rewrite` are discussed in the [URL Rewriting Guide](#).



## FILE NOT FOUND

Inevitably, URLs will be requested for which no matching file can be found in the filesystem. This can happen for several reasons. In some cases, it can be a result of moving documents from one location to another. In this case, it is best to use [URL redirection](#) to inform clients of the new location of the resource. In this way, you can assure that old bookmarks and links will continue to work, even though the resource is at a new location.

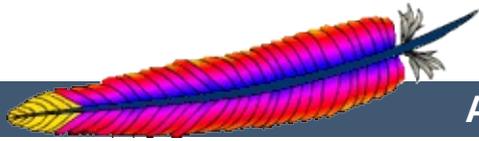
Another common cause of "File Not Found" errors is accidental mistyping of URLs, either directly in the browser, or in HTML links. Apache provides the module [mod\\_speling](#) (sic) to help with this problem. When this module is activated, it will intercept "File Not Found" errors and look for a resource with a similar filename. If one such file is found, mod\_speling will send an HTTP redirect to the client informing it of the correct location. If several "close" files are found, a list of available alternatives will be presented to the client.

An especially useful feature of mod\_speling, is that it will compare filenames without respect to case. This can help systems where users are unaware of the case-sensitive nature of URLs and the unix filesystem. But using mod\_speling for anything more than the occasional URL correction can place additional load on the server, since each "incorrect" request is followed by a URL redirection and a new request from the client.

If all attempts to locate the content fail, Apache returns an error page with HTTP status code 404 (file not found). The appearance of this page is controlled with the [ErrorDocument](#) directive and can be customized in a flexible manner as discussed in the [Custom error responses](#) and [International Server Error Responses](#) documents.

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

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



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



## Apache HTTP Server Version 2.0

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

## Security Tips

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



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

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



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

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

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

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

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

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

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



---

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

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

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

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

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

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

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



---

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

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



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

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



## Script Aliased CGI

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

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



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



## Preventing .htaccess Overrides

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

In the server configuration file, put

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

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



## Prevent Default Access by Default

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

For instance, consider the following example:

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

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

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

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

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

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

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

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

```
UserDir disabled root
```



## Monitoring Your Logs

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

A couple of examples:

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

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

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

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

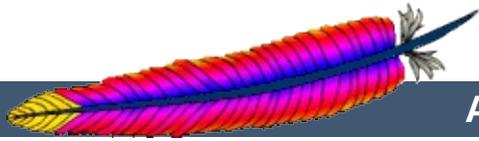
```
foo.bar.com - - [12/Jul/2002:01:59:13 +0200] "GET /.htpasswd  
HTTP/1.1"
```

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

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

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

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



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



## Apache HTTP Server Version 2.0

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

## Dynamic Shared Object (DSO) Support

The Apache HTTP Server is a modular program where the administrator can choose the functionality to include in the server by selecting a set of modules. The modules can be statically compiled into the [httpd](#) binary when the server is built. Alternatively, modules can be compiled as Dynamic Shared Objects (DSOs) that exist separately from the main [httpd](#) binary file. DSO modules may be compiled at the time the server is built, or they may be compiled and added at a later time using the Apache Extension Tool ([apxs](#)).

This document describes how to use DSO modules as well as the theory behind their use.



Implementation

Related Directives

[mod\\_so](#)

[LoadModule](#)

The DSO support for loading individual Apache modules is based on a module named [mod\\_so](#) which must be statically compiled into the Apache core. It is the only module besides [core](#) which cannot be put into a DSO itself. Practically all other distributed Apache modules can then be placed into a DSO by individually enabling the DSO build for them via [configure](#)'s `--enable-module=shared` option as discussed in the [install documentation](#). After a module is compiled into a DSO named `mod_foo.so` you can use [mod\\_so](#)'s [LoadModule](#) command in your `httpd.conf` file to load this module at server startup or restart.

To simplify this creation of DSO files for Apache modules (especially for third-party modules) a new support program named [apxs](#) (*APache eXtenSion*) is available. It can be used to build DSO based modules *outside of* the Apache source tree. The idea is simple: When installing Apache the [configure](#)'s `make install` procedure installs the Apache C header files and puts the platform-dependent compiler and linker flags for building DSO files into the [apxs](#) program. This way the user can use [apxs](#) to compile his Apache module sources without the Apache distribution source tree and without having to fiddle with the platform-dependent compiler and linker flags for DSO support.



To give you an overview of the DSO features of Apache 2.x, here is a short and concise summary:

1. Build and install a *distributed* Apache module, say `mod_foo.c`, into its own DSO `mod_foo.so`:

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

2. Build and install a *third-party* Apache module, say `mod_foo.c`, into its own DSO `mod_foo.so`:

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

3. Configure Apache for *later installation* of shared modules:

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

4. Build and install a *third-party* Apache module, say `mod_foo.c`, into its own DSO `mod_foo.so` *outside of* the Apache source tree using [apxs](#):

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

In all cases, once the shared module is compiled, you must use a [LoadModule](#) directive in `httpd.conf` to tell Apache to activate the module.



On modern Unix derivatives there exists a nifty mechanism usually called dynamic linking/loading of *Dynamic Shared Objects* (DSO) which provides a way to build a piece of program code in a special format for loading it at run-time into the address space of an executable program.

This loading can usually be done in two ways: Automatically by a system program called `ld.so` when an executable program is started or manually from within the executing program via a programmatic system interface to the Unix loader through the system calls `dlopen()`/`dlsym()`.

In the first way the DSO's are usually called *shared libraries* or *DSO libraries* and named `libfoo.so` or `libfoo.so.1.2`. They reside in a system directory (usually `/usr/lib`) and the link to the executable program is established at build-time by specifying `-lfoo` to the linker command. This hard-codes library references into the executable program file so that at start-time the Unix loader is able to locate `libfoo.so` in `/usr/lib`, in paths hard-coded via linker-options like `-R` or in paths configured via the environment variable `LD_LIBRARY_PATH`. It then resolves any (yet unresolved) symbols in the executable program which are available in the DSO.

Symbols in the executable program are usually not referenced by the DSO (because it's a reusable library of general code) and hence no further resolving has to be done. The executable program has no need to do anything on its own to use the symbols from the DSO because the complete resolving is done by the Unix loader. (In fact, the code to invoke `ld.so` is part of the run-time startup code which is linked into every executable program which has been bound non-static). The advantage of dynamic loading of common library code is obvious: the library code needs to be

stored only once, in a system library like `libc.so`, saving disk space for every program.

In the second way the DSO's are usually called *shared objects* or *DSO files* and can be named with an arbitrary extension (although the canonical name is `foo.so`). These files usually stay inside a program-specific directory and there is no automatically established link to the executable program where they are used. Instead the executable program manually loads the DSO at run-time into its address space via `dlopen()`. At this time no resolving of symbols from the DSO for the executable program is done. But instead the Unix loader automatically resolves any (yet unresolved) symbols in the DSO from the set of symbols exported by the executable program and its already loaded DSO libraries (especially all symbols from the ubiquitous `libc.so`). This way the DSO gets knowledge of the executable program's symbol set as if it had been statically linked with it in the first place.

Finally, to take advantage of the DSO's API the executable program has to resolve particular symbols from the DSO via `dlsym()` for later use inside dispatch tables *etc.* In other words: The executable program has to manually resolve every symbol it needs to be able to use it. The advantage of such a mechanism is that optional program parts need not be loaded (and thus do not spend memory) until they are needed by the program in question. When required, these program parts can be loaded dynamically to extend the base program's functionality.

Although this DSO mechanism sounds straightforward there is at least one difficult step here: The resolving of symbols from the executable program for the DSO when using a DSO to extend a program (the second way). Why? Because "reverse resolving" DSO symbols from the executable program's symbol set is against the library design (where the library has no knowledge about the

programs it is used by) and is neither available under all platforms nor standardized. In practice the executable program's global symbols are often not re-exported and thus not available for use in a DSO. Finding a way to force the linker to export all global symbols is the main problem one has to solve when using DSO for extending a program at run-time.

The shared library approach is the typical one, because it is what the DSO mechanism was designed for, hence it is used for nearly all types of libraries the operating system provides. On the other hand using shared objects for extending a program is not used by a lot of programs.

As of 1998 there are only a few software packages available which use the DSO mechanism to actually extend their functionality at run-time: Perl 5 (via its XS mechanism and the DynaLoader module), Netscape Server, *etc.* Starting with version 1.3, Apache joined the crew, because Apache already uses a module concept to extend its functionality and internally uses a dispatch-list-based approach to link external modules into the Apache core functionality. So, Apache is really predestined for using DSO to load its modules at run-time.



The above DSO based features have the following advantages:

- The server package is more flexible at run-time because the actual server process can be assembled at run-time via `LoadModule` `httpd.conf` configuration commands instead of `configure` options at build-time. For instance this way one is able to run different server instances (standard & SSL version, minimalistic & powered up version [`mod_perl`, `PHP3`], *etc.*) with only one Apache installation.
- The server package can be easily extended with third-party modules even after installation. This is at least a great benefit for vendor package maintainers who can create a Apache core package and additional packages containing extensions like `PHP3`, `mod_perl`, `mod_fastcgi`, *etc.*
- Easier Apache module prototyping because with the DSO/`apxs` pair you can both work outside the Apache source tree and only need an `apxs -i` command followed by an `apachectl restart` to bring a new version of your currently developed module into the running Apache server.

DSO has the following disadvantages:

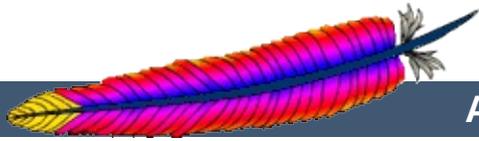
- The DSO mechanism cannot be used on every platform because not all operating systems support dynamic loading of code into the address space of a program.
- The server is approximately 20% slower at startup time because of the symbol resolving overhead the Unix loader now has to do.
- The server is approximately 5% slower at execution time under some platforms because position independent code (PIC) sometimes needs complicated assembler tricks for relative addressing which are not necessarily as fast as absolute addressing.
- Because DSO modules cannot be linked against other DSO-

based libraries (`ld -lfoo`) on all platforms (for instance a.out-based platforms usually don't provide this functionality while ELF-based platforms do) you cannot use the DSO mechanism for all types of modules. Or in other words, modules compiled as DSO files are restricted to only use symbols from the Apache core, from the C library (`libc`) and all other dynamic or static libraries used by the Apache core, or from static library archives (`libfoo.a`) containing position independent code. The only chances to use other code is to either make sure the Apache core itself already contains a reference to it or loading the code yourself via `dlopen()`.

---

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

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



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



## Apache HTTP Server Version 2.0

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

## Content Negotiation

Apache supports content negotiation as described in the HTTP/1.1 specification. It can choose the best representation of a resource based on the browser-supplied preferences for media type, languages, character set and encoding. It also implements a couple of features to give more intelligent handling of requests from browsers that send incomplete negotiation information.

Content negotiation is provided by the [mod\\_negotiation](#) module, which is compiled in by default.



## Content Negotiation

A resource may be available in several different representations. For example, it might be available in different languages or different media types, or a combination. One way of selecting the most appropriate choice is to give the user an index page, and let them select. However it is often possible for the server to choose automatically. This works because browsers can send, as part of each request, information about what representations they prefer. For example, a browser could indicate that it would like to see information in French, if possible, else English will do. Browsers indicate their preferences by headers in the request. To request only French representations, the browser would send

```
Accept-Language: fr
```

Note that this preference will only be applied when there is a choice of representations and they vary by language.

As an example of a more complex request, this browser has been configured to accept French and English, but prefer French, and to accept various media types, preferring HTML over plain text or other text types, and preferring GIF or JPEG over other media types, but also allowing any other media type as a last resort:

```
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 supports 'server driven' content negotiation, as defined in the HTTP/1.1 specification. It fully supports the Accept, Accept-Language, Accept-Charset and Accept-Encoding request headers. Apache also supports 'transparent' content negotiation, which is an experimental negotiation protocol defined in RFC 2295 and RFC 2296. It does not offer support for 'feature negotiation' as defined in these RFCs.

A **resource** is a conceptual entity identified by a URI (RFC 2396). An HTTP server like Apache provides access to **representations** of the resource(s) within its namespace, with each representation in the form of a sequence of bytes with a defined media type, character set, encoding, etc. Each resource may be associated with zero, one, or more than one representation at any given time. If multiple representations are available, the resource is referred to as **negotiable** and each of its representations is termed a **variant**. The ways in which the variants for a negotiable resource vary are called the **dimensions** of negotiation.



## Negotiation in Apache

In order to negotiate a resource, the server needs to be given information about each of the variants. This is done in one of two ways:

- Using a type map (*i.e.*, a \*.var file) which names the files containing the variants explicitly, or
- Using a 'MultiViews' search, where the server does an implicit filename pattern match and chooses from among the results.

### Using a type-map file

A type map is a document which is associated with the handler named type-map (or, for backwards-compatibility with older Apache configurations, the MIME type application/x-type-map). Note that to use this feature, you must have a handler set in the configuration that defines a file suffix as type-map; this is best done with

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

in the server configuration file.

Type map files should have the same name as the resource which they are describing, and have an entry for each available variant; these entries consist of contiguous HTTP-format header lines. Entries for different variants are separated by blank lines. Blank lines are illegal within an entry. It is conventional to begin a map file with an entry for the combined entity as a whole (although this is not required, and if present will be ignored). An example map file is shown below. This file would be named foo.var, as it describes a resource named foo.

```
URI: foo
```

```
URI: foo.en.html
Content-type: text/html
Content-language: en

URI: foo.fr.de.html
Content-type: text/html;charset=iso-8859-2
Content-language: fr, de
```

Note also that a typemap file will take precedence over the filename's extension, even when Multiviews is on. If the variants have different source qualities, that may be indicated by the "qs" parameter to the media type, as in this picture (available as JPEG, GIF, or ASCII-art):

```
URI: foo

URI: foo.jpeg
Content-type: image/jpeg; qs=0.8

URI: foo.gif
Content-type: image/gif; qs=0.5

URI: foo.txt
Content-type: text/plain; qs=0.01
```

qs values can vary in the range 0.000 to 1.000. Note that any variant with a qs value of 0.000 will never be chosen. Variants with no 'qs' parameter value are given a qs factor of 1.0. The qs parameter indicates the relative 'quality' of this variant compared to the other available variants, independent of the client's capabilities. For example, a JPEG file is usually of higher source quality than an ASCII file if it is attempting to represent a photograph. However, if the resource being represented is an original ASCII art, then an ASCII representation would have a higher source quality than a JPEG representation. A qs value is therefore specific to a given variant depending on the nature of the resource it represents.

The full list of headers recognized is available in the

[mod\\_negotiation typemap](#) documentation.

## Multiviews

MultiViews is a per-directory option, meaning it can be set with an [Options](#) directive within a [<Directory>](#), [<Location>](#) or [<Files>](#) section in `httpd.conf`, or (if [AllowOverride](#) is properly set) in `.htaccess` files. Note that `Options All` does not set MultiViews; you have to ask for it by name.

The effect of MultiViews is as follows: if the server receives a request for `/some/dir/foo`, if `/some/dir` has MultiViews enabled, and `/some/dir/foo` does *not* exist, then the server reads the directory looking for files named `foo.*`, and effectively fakes up a type map which names all those files, assigning them the same media types and content-encodings it would have if the client had asked for one of them by name. It then chooses the best match to the client's requirements.

MultiViews may also apply to searches for the file named by the [DirectoryIndex](#) directive, if the server is trying to index a directory. If the configuration files specify

```
DirectoryIndex index
```

then the server will arbitrate between `index.html` and `index.html3` if both are present. If neither are present, and `index.cgi` is there, the server will run it.

If one of the files found when reading the directory does not have an extension recognized by `mod_mime` to designate its Charset, Content-Type, Language, or Encoding, then the result depends on the setting of the [MultiViewsMatch](#) directive. This directive determines whether handlers, filters, and other extension types

can participate in MultiViews negotiation.



## The Negotiation Methods

After Apache has obtained a list of the variants for a given resource, either from a type-map file or from the filenames in the directory, it invokes one of two methods to decide on the 'best' variant to return, if any. It is not necessary to know any of the details of how negotiation actually takes place in order to use Apache's content negotiation features. However the rest of this document explains the methods used for those interested.

There are two negotiation methods:

1. **Server driven negotiation with the Apache algorithm** is used in the normal case. The Apache algorithm is explained in more detail below. When this algorithm is used, Apache can sometimes 'fiddle' the quality factor of a particular dimension to achieve a better result. The ways Apache can fiddle quality factors is explained in more detail below.
2. **Transparent content negotiation** is used when the browser specifically requests this through the mechanism defined in RFC 2295. This negotiation method gives the browser full control over deciding on the 'best' variant, the result is therefore dependent on the specific algorithms used by the browser. As part of the transparent negotiation process, the browser can ask Apache to run the 'remote variant selection algorithm' defined in RFC 2296.

## Dimensions of Negotiation

### Dimension Notes

Media Type	Browser indicates preferences with the Accept header field. Each item can have an associated quality factor. Variant description can also have a quality factor (the "qs" parameter).
Language	Browser indicates preferences with the Accept -

	Language header field. Each item can have a quality factor. Variants can be associated with none, one or more than one language.
Encoding	Browser indicates preference with the Accept - Encoding header field. Each item can have a quality factor.
Charset	Browser indicates preference with the Accept - Charset header field. Each item can have a quality factor. Variants can indicate a charset as a parameter of the media type.

## Apache Negotiation Algorithm

Apache can use the following algorithm to select the 'best' variant (if any) to return to the browser. This algorithm is not further configurable. It operates as follows:

1. First, for each dimension of the negotiation, check the appropriate *Accept\** header field and assign a quality to each variant. If the *Accept\** header for any dimension implies that this variant is not acceptable, eliminate it. If no variants remain, go to step 4.
2. Select the 'best' variant by a process of elimination. Each of the following tests is applied in order. Any variants not selected at each test are eliminated. After each test, if only one variant remains, select it as the best match and proceed to step 3. If more than one variant remains, move on to the next test.
  1. Multiply the quality factor from the Accept header with the quality-of-source factor for this variants media type, and select the variants with the highest value.
  2. Select the variants with the highest language quality factor.

3. Select the variants with the best language match, using either the order of languages in the Accept - Language header (if present), or else the order of languages in the LanguagePriority directive (if present).
  4. Select the variants with the highest 'level' media parameter (used to give the version of text/html media types).
  5. Select variants with the best charset media parameters, as given on the Accept - Charset header line. Charset ISO-8859-1 is acceptable unless explicitly excluded. Variants with a text/\* media type but not explicitly associated with a particular charset are assumed to be in ISO-8859-1.
  6. Select those variants which have associated charset media parameters that are *not* ISO-8859-1. If there are no such variants, select all variants instead.
  7. Select the variants with the best encoding. If there are variants with an encoding that is acceptable to the user-agent, select only these variants. Otherwise if there is a mix of encoded and non-encoded variants, select only the unencoded variants. If either all variants are encoded or all variants are not encoded, select all variants.
  8. Select the variants with the smallest content length.
  9. Select the first variant of those remaining. This will be either the first listed in the type-map file, or when variants are read from the directory, the one whose file name comes first when sorted using ASCII code order.
3. The algorithm has now selected one 'best' variant, so return it as the response. The HTTP response header Vary is set to indicate the dimensions of negotiation (browsers and caches can use this information when caching the resource). End.

4. To get here means no variant was selected (because none are acceptable to the browser). Return a 406 status (meaning "No acceptable representation") with a response body consisting of an HTML document listing the available variants. Also set the HTTP Vary header to indicate the dimensions of variance.



Apache sometimes changes the quality values from what would be expected by a strict interpretation of the Apache negotiation algorithm above. This is to get a better result from the algorithm for browsers which do not send full or accurate information. Some of the most popular browsers send Accept header information which would otherwise result in the selection of the wrong variant in many cases. If a browser sends full and correct information these fiddles will not be applied.

## Media Types and Wildcards

The Accept : request header indicates preferences for media types. It can also include 'wildcard' media types, such as "image/\*" or "\*/\*" where the \* matches any string. So a request including:

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

would indicate that any type starting "image/" is acceptable, as is any other type. Some browsers routinely send wildcards in addition to explicit types they can handle. For example:

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

The intention of this is to indicate that the explicitly listed types are preferred, but if a different representation is available, that is ok too. Using explicit quality values, what the browser really wants is something like:

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

The explicit types have no quality factor, so they default to a preference of 1.0 (the highest). The wildcard \*/\* is given a low preference of 0.01, so other types will only be returned if no

variant matches an explicitly listed type.

If the `Accept :` header contains *no* q factors at all, Apache sets the q value of `"/*`, if present, to 0.01 to emulate the desired behavior. It also sets the q value of wildcards of the format `"type/*"` to 0.02 (so these are preferred over matches against `"/*`). If any media type on the `Accept :` header contains a q factor, these special values are *not* applied, so requests from browsers which send the explicit information to start with work as expected.

## Language Negotiation Exceptions

New in Apache 2.0, some exceptions have been added to the negotiation algorithm to allow graceful fallback when language negotiation fails to find a match.

When a client requests a page on your server, but the server cannot find a single page that matches the `Accept - language` sent by the browser, the server will return either a "No Acceptable Variant" or "Multiple Choices" response to the client. To avoid these error messages, it is possible to configure Apache to ignore the `Accept - language` in these cases and provide a document that does not explicitly match the client's request. The [ForceLanguagePriority](#) directive can be used to override one or both of these error messages and substitute the server's judgement in the form of the [LanguagePriority](#) directive.

The server will also attempt to match language-subsets when no other match can be found. For example, if a client requests documents with the language `en - GB` for British English, the server is not normally allowed by the HTTP/1.1 standard to match that against a document that is marked as simply `en`. (Note that it is almost surely a configuration error to include `en - GB` and not `en` in the `Accept - Language` header, since it is very unlikely that a reader understands British English, but doesn't understand

English in general. Unfortunately, many current clients have default configurations that resemble this.) However, if no other language match is possible and the server is about to return a "No Acceptable Variants" error or fallback to the [LanguagePriority](#), the server will ignore the subset specification and match en-GB against en documents. Implicitly, Apache will add the parent language to the client's acceptable language list with a very low quality value. But note that if the client requests "en-GB; q=0.9, fr; q=0.8", and the server has documents designated "en" and "fr", then the "fr" document will be returned. This is necessary to maintain compliance with the HTTP/1.1 specification and to work effectively with properly configured clients.

In order to support advanced techniques (such as cookies or special URL-paths) to determine the user's preferred language, since Apache 2.0.47 [mod\\_negotiation](#) recognizes the [environment variable](#) `prefer-language`. If it exists and contains an appropriate language tag, [mod\\_negotiation](#) will try to select a matching variant. If there's no such variant, the normal negotiation process applies.

### Example

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



## Extensions to Transparent Content Negotiation

Apache extends the transparent content negotiation protocol (RFC 2295) as follows. A new `{encoding . . }` element is used in variant lists to label variants which are available with a specific content-encoding only. The implementation of the RVSA/1.0 algorithm (RFC 2296) is extended to recognize encoded variants in the list, and to use them as candidate variants whenever their encodings are acceptable according to the `Accept-Encoding` request header. The RVSA/1.0 implementation does not round computed quality factors to 5 decimal places before choosing the best variant.



## File naming, permissions and naming conventions

If you are using language negotiation you can choose between different naming conventions, because files can have more than one extension, and the order of the extensions is normally irrelevant (see the [mod\\_mime](#) documentation for details).

A typical file has a MIME-type extension (e.g., `html`), maybe an encoding extension (e.g., `gz`), and of course a language extension (e.g., `en`) when we have different language variants of this file.

Examples:

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

Here some more examples of filenames together with valid and invalid hyperlinks:

<b>Filename</b>	<b>Valid hyperlink</b>	<b>Invalid hyperlink</b>
<i>foo.html.en</i>	foo foo.html	-
<i>foo.en.html</i>	foo	foo.html
<i>foo.html.en.gz</i>	foo foo.html	foo.gz foo.html.gz
<i>foo.en.html.gz</i>	foo	foo.html foo.html.gz foo.gz
<i>foo.gz.html.en</i>	foo foo.gz foo.gz.html	foo.html
<i>foo.html.gz.en</i>	foo foo.html foo.html.gz	foo.gz

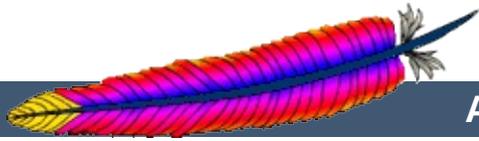
Looking at the table above, you will notice that it is always possible to use the name without any extensions in a hyperlink (e.g., `foo`). The advantage is that you can hide the actual type of a document resp. file and can change it later, e.g., from `html` to `shtml` or `cgi` without changing any hyperlink references.

If you want to continue to use a MIME-type in your hyperlinks (e.g. `foo.html`) the language extension (including an encoding extension if there is one) must be on the right hand side of the MIME-type extension (e.g., `foo.html.en`).



When a cache stores a representation, it associates it with the request URL. The next time that URL is requested, the cache can use the stored representation. But, if the resource is negotiable at the server, this might result in only the first requested variant being cached and subsequent cache hits might return the wrong response. To prevent this, Apache normally marks all responses that are returned after content negotiation as non-cacheable by HTTP/1.0 clients. Apache also supports the HTTP/1.1 protocol features to allow caching of negotiated responses.

For requests which come from a HTTP/1.0 compliant client (either a browser or a cache), the directive [CacheNegotiatedDocs](#) can be used to allow caching of responses which were subject to negotiation. This directive can be given in the server config or virtual host, and takes no arguments. It has no effect on requests from HTTP/1.1 clients.



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



## Apache HTTP Server Version 2.0

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

## Custom Error Responses

Additional functionality allows webmasters to configure the response of Apache to some error or problem.

Customizable responses can be defined to be activated in the event of a server detected error or problem.

If a script crashes and produces a "500 Server Error" response, then this response can be replaced with either some friendlier text or by a redirection to another URL (local or external).



## Old Behavior

NCSA httpd 1.3 would return some boring old error/problem message which would often be meaningless to the user, and would provide no means of logging the symptoms which caused it.

## New Behavior

The server can be asked to:

1. Display some other text, instead of the NCSA hard coded messages, or
2. redirect to a local URL, or
3. redirect to an external URL.

Redirecting to another URL can be useful, but only if some information can be passed which can then be used to explain and/or log the error/problem more clearly.

To achieve this, Apache will define new CGI-like environment variables:

```
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
```

Note the REDIRECT\_ prefix.

At least REDIRECT\_URL and REDIRECT\_QUERY\_STRING will be passed to the new URL (assuming it's a cgi-script or a cgi-include). The other variables will exist only if they existed prior to the error/problem. **None** of these will be set if your [ErrorDocument](#) is an *external* redirect (anything starting with a scheme name like `http:`, even if it refers to the same host as the server).



Use of [ErrorDocument](#) is enabled for .htaccess files when the [AllowOverride](#) is set accordingly.

Here are some examples...

```
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
```

The syntax is,

```
ErrorDocument <3-digit-code> <action>
```

where the action can be,

1. Text to be displayed. Prefix the text with a quote ("). Whatever follows the quote is displayed. *Note: the (") prefix isn't displayed.*
2. An external URL to redirect to.
3. A local URL to redirect to.



## Custom Error Responses and Redirects

Apache's behavior to redirected URLs has been modified so that additional environment variables are available to a script/server-include.

### Old behavior

Standard CGI vars were made available to a script which has been redirected to. No indication of where the redirection came from was provided.

### New behavior

A new batch of environment variables will be initialized for use by a script which has been redirected to. Each new variable will have the prefix REDIRECT\_. REDIRECT\_ environment variables are created from the CGI environment variables which existed prior to the redirect, they are renamed with a REDIRECT\_ prefix, *i.e.*, HTTP\_USER\_AGENT becomes REDIRECT\_HTTP\_USER\_AGENT. In addition to these new variables, Apache will define REDIRECT\_URL and REDIRECT\_STATUS to help the script trace its origin. Both the original URL and the URL being redirected to can be logged in the access log.

If the ErrorDocument specifies a local redirect to a CGI script, the script should include a "Status:" header field in its output in order to ensure the propagation all the way back to the client of the error condition that caused it to be invoked. For instance, a Perl ErrorDocument script might include the following:

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

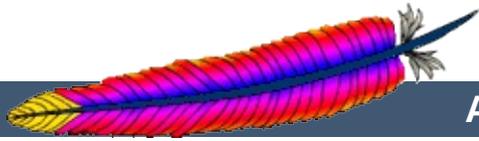
If the script is dedicated to handling a particular error condition, such as 404 Not Found, it can use the specific code and error text instead.

Note that the script *must* emit an appropriate `Status:` header (such as 302 Found), if the response contains a `Location:` header (in order to issue a client side redirect). Otherwise the `Location:` header may have no effect.

---

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

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



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



## Apache HTTP Server Version 2.0

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

# Binding

Configuring Apache to listen on specific addresses and ports.

## See also

[Virtual Hosts](#)

[DNS Issues](#)



Related Modules	Related Directives
<a href="#">core</a>	<a href="#">&lt;VirtualHost&gt;</a>
<a href="#">mpm_common</a>	<a href="#">Listen</a>

When Apache starts, it binds to some port and address on the local machine and waits for incoming requests. By default, it listens to all addresses on the machine. However, it needs to be told to listen on specific ports, or to listen on only selected addresses, or a combination. This is often combined with the Virtual Host feature which determines how Apache responds to different IP addresses, hostnames and ports.

The [Listen](#) directive tells the server to accept incoming requests only on the specified port or address-and-port combinations. If only a port number is specified in the [Listen](#) directive, the server listens to the given port on all interfaces. If an IP address is given as well as a port, the server will listen on the given port and interface. Multiple [Listen](#) directives may be used to specify a number of addresses and ports to listen on. The server will respond to requests from any of the listed addresses and ports.

For example, to make the server accept connections on both port 80 and port 8000, use:

```
Listen 80
Listen 8000
```

To make the server accept connections on two specified interfaces and port numbers, use

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

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

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```



## Special IPv6 Considerations

A growing number of platforms implement IPv6, and APR supports IPv6 on most of these platforms, allowing Apache to allocate IPv6 sockets and handle requests which were sent over IPv6.

One complicating factor for Apache administrators is whether or not an IPv6 socket can handle both IPv4 connections and IPv6 connections. Handling IPv4 connections with an IPv6 socket uses IPv4-mapped IPv6 addresses, which are allowed by default on most platforms but are disallowed by default on FreeBSD, NetBSD, and OpenBSD in order to match the system-wide policy on those platforms. But even on systems where it is disallowed by default, a special `configure` parameter can change this behavior for Apache.

If you want Apache to handle IPv4 and IPv6 connections with a minimum of sockets, which requires using IPv4-mapped IPv6 addresses, specify the `--enable-v4-mapped-configure` option and use generic `Listen` directives like the following:

```
Listen 80
```

With `--enable-v4-mapped`, the `Listen` directives in the default configuration file created by Apache will use this form. `--enable-v4-mapped` is the default on all platforms but FreeBSD, NetBSD, and OpenBSD, so this is probably how your Apache was built.

If you want Apache to handle IPv4 connections only, regardless of what your platform and APR will support, specify an IPv4 address on all `Listen` directives, as in the following examples:

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

If you want Apache to handle IPv4 and IPv6 connections on

separate sockets (i.e., to disable IPv4-mapped addresses), specify the `--disable-v4-mapped` [configure](#) option and use specific Listen directives like the following:

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

With `--disable-v4-mapped`, the Listen directives in the default configuration file created by Apache will use this form. `--disable-v4-mapped` is the default on FreeBSD, NetBSD, and OpenBSD.



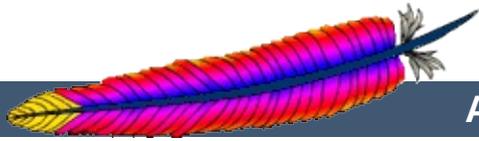
**HOW THE `Listen` DIRECTIVE WORKS**

`Listen` does not implement Virtual Hosts. It only tells the main server what addresses and ports to listen to. If no `<VirtualHost>` directives are used, the server will behave the same for all accepted requests. However, `<VirtualHost>` can be used to specify a different behavior for one or more of the addresses and ports. To implement a VirtualHost, the server must first be told to listen to the address and port to be used. Then a `<VirtualHost>` section should be created for a specified address and port to set the behavior of this virtual host. Note that if the `<VirtualHost>` is set for an address and port that the server is not listening to, it cannot be accessed.

---

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

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



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

## **Multi-Processing-Module (MPMs)**

Das Dokument beschreibt, was ein Multi-Processing-Modul ist und wie solche Module beim Apache HTTP Server verwendet werden.



Der Apache HTTP Server wurde als leistungsfähiger und flexibler Webserver konzipiert, der auf einer Vielzahl von Plattformen in einer Reihe unterschiedlicher Umgebungen arbeiten kann. Unterschiedliche Plattformen und unterschiedliche Umgebungen verlangen oftmals verschiedene Fähigkeiten oder kennen verschiedene Wege, die gleiche Funktionalität sehr effizient zu implementieren. Der Apache hat durch seinen modularen Aufbau schon immer eine breite Auswahl von Umgebungen unterstützt. Dieses Design erlaubt es dem Webmaster, durch Auswahl der Module, die zur Kompilierungszeit oder zur Laufzeit geladen werden, die Features auszuwählen, die in den Server integriert werden.

Der Apache 2.0 erweitert dieses modulare Design auf die grundlegenden Funktionen eines Webserver. Der Server wird mit einer Auswahl von Multi-Processing-Modulen (MPMs) ausgeliefert, die für die Bindung an Netzwerkports der Maschine, die Annahme von Anfragen und die Abfertigung von Kindprozessen zur Behandlung der Anfragen zuständig sind.

Die Erweiterung des modularen Aufbaus auf diese Ebene des Servers bringt zwei wesentliche Vorteile:

- Der Apache kann nun eine Vielfalt von Betriebssystemen sauberer und effizienter unterstützen. Insbesondere die Windows-Version des Apache ist jetzt deutlich effizienter, da mpm\_winnt native Netzwerkfähigkeiten anstelle der im Apache 1.3 verwendeten POSIX-Schicht benutzen kann. Dieser Vorteil gilt auch für andere Betriebssysteme, für die spezielle MPMs implementiert sind.
- Der Server läßt sich besser auf die Bedürfnisse der jeweiligen Website anpassen. Sites beispielsweise, die eine hohe Skalierbarkeit benötigen, können ein Threaded-MPM wie worker wählen, während Sites, die Stabilität oder

Kompatibilität mit älterer Software erfordern, [prefork](#) wählen können. Darüber hinaus können Spezialfähigkeiten wie die Bedienung verschiedener Hosts unter unterschiedlichen User-IDs ([perchild](#)) angeboten werden.

Auf Anwenderebene erscheinen MPMs fast wie andere Apache-Module. Der Hauptunterschied ist, dass jeweils nur ein einziges MPM in den Server geladen werden kann. Die Liste der verfügbaren MPMs finden Sie im [Modul-Index](#).



MPMs müssen während der (*Anm.d.Ü.: Quelltext-*)Konfiguration ausgewählt und in den Server einkompiliert werden. Compiler sind in der Lage eine Reihe von Funktionen zu optimieren, wenn Threads verwendet werden. Sie können dies allerdings nur, wenn sie wissen, dass Threads benutzt werden.

Um das gewünschte MPM tatsächlich auszuwählen, verwenden Sie beim `configure`-Skript das Argument `--with-mpm=NAME`. *NAME* ist der Name des gewünschten MPMs.

Ist der Server kompiliert, so ist es mittels `./httpd -l` möglich, das ausgewählte MPM zu ermitteln. Dieser Befehl listet alle in den Server einkompilierten Module auf, einschließlich des MPM.



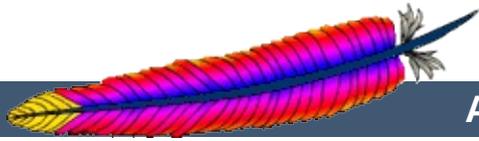
Die folgende Tabelle gibt die voreingestellten MPMs für verschiedene Betriebssysteme an. Wenn Sie während der Kompilierung keine andere Auswahl treffen, wird dieses MPM gewählt.

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

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



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



## Apache HTTP Server Version 2.0

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

## Environment Variables in Apache

The Apache HTTP Server provides a mechanism for storing information in named variables that are called *environment variables*. This information can be used to control various operations such as logging or access control. The variables are also used as a mechanism to communicate with external programs such as CGI scripts. This document discusses different ways to manipulate and use these variables.

Although these variables are referred to as *environment variables*, they are not the same as the environment variables controlled by the underlying operating system. Instead, these variables are stored and manipulated in an internal Apache structure. They only become actual operating system environment variables when they are provided to CGI scripts and Server Side Include scripts. If you wish to manipulate the operating system environment under which the server itself runs, you must use the standard environment manipulation mechanisms provided by your operating system shell.



Related Modules	Related Directives
<a href="#">mod_env</a>	<a href="#">BrowserMatch</a>
<a href="#">mod_rewrite</a>	<a href="#">BrowserMatchNoCase</a>
<a href="#">mod_setenvif</a>	<a href="#">PassEnv</a>
<a href="#">mod_unique_id</a>	<a href="#">RewriteRule</a>
	<a href="#">SetEnv</a>
	<a href="#">SetEnvIf</a>
	<a href="#">SetEnvIfNoCase</a>
	<a href="#">UnsetEnv</a>

## Basic Environment Manipulation

The most basic way to set an environment variable in Apache is using the unconditional [SetEnv](#) directive. Variables may also be passed from the environment of the shell which started the server using the [PassEnv](#) directive.

## Conditional Per-Request Settings

For additional flexibility, the directives provided by [mod\\_setenvif](#) allow environment variables to be set on a per-request basis, conditional on characteristics of particular requests. For example, a variable could be set only when a specific browser (User-Agent) is making a request, or only when a specific Referer [sic] header is found. Even more flexibility is available through the [mod\\_rewrite](#)'s [RewriteRule](#) which uses the [E=...] option to set environment variables.

## Unique Identifiers

Finally, [mod\\_unique\\_id](#) sets the environment variable UNIQUE\_ID for each request to a value which is guaranteed to be unique across "all" requests under very specific conditions.

## Standard CGI Variables

In addition to all environment variables set within the Apache configuration and passed from the shell, CGI scripts and SSI pages are provided with a set of environment variables containing meta-information about the request as required by the [CGI specification](#).

## Some Caveats

- It is not possible to override or change the standard CGI variables using the environment manipulation directives.
- When [suexec](#) is used to launch CGI scripts, the environment will be cleaned down to a set of *safe* variables before CGI scripts are launched. The list of *safe* variables is defined at compile-time in `suexec.c`.
- For portability reasons, the names of environment variables may contain only letters, numbers, and the underscore character. In addition, the first character may not be a number. Characters which do not match this restriction will be replaced by an underscore when passed to CGI scripts and SSI pages.
- The [SetEnv](#) directive runs late during request processing meaning that directives such as [SetEnvIf](#) and [RewriteCond](#) will not see the variables set with it.



Related Modules	Related Directives
<a href="#">mod_access</a>	<a href="#">Allow</a>
<a href="#">mod_cgi</a>	<a href="#">CustomLog</a>
<a href="#">mod_ext_filter</a>	<a href="#">Deny</a>
<a href="#">mod_headers</a>	<a href="#">ExtFilterDefine</a>
<a href="#">mod_include</a>	<a href="#">Header</a>
<a href="#">mod_log_config</a>	<a href="#">LogFormat</a>
<a href="#">mod_rewrite</a>	<a href="#">RewriteCond</a> <a href="#">RewriteRule</a>

## CGI Scripts

One of the primary uses of environment variables is to communicate information to CGI scripts. As discussed above, the environment passed to CGI scripts includes standard meta-information about the request in addition to any variables set within the Apache configuration. For more details, see the [CGI tutorial](#).

## SSI Pages

Server-parsed (SSI) documents processed by [mod\\_include](#)'s INCLUDES filter can print environment variables using the echo element, and can use environment variables in flow control elements to make parts of a page conditional on characteristics of a request. Apache also provides SSI pages with the standard CGI environment variables as discussed above. For more details, see the [SSI tutorial](#).

## Access Control

Access to the server can be controlled based on the value of environment variables using the `allow from env=` and `deny`

from `env=` directives. In combination with [SetEnvIf](#), this allows for flexible control of access to the server based on characteristics of the client. For example, you can use these directives to deny access to a particular browser (User-Agent).

## Conditional Logging

Environment variables can be logged in the access log using the [LogFormat](#) option `%e`. In addition, the decision on whether or not to log requests can be made based on the status of environment variables using the conditional form of the [CustomLog](#) directive. In combination with [SetEnvIf](#) this allows for flexible control of which requests are logged. For example, you can choose not to log requests for filenames ending in `gif`, or you can choose to only log requests from clients which are outside your subnet.

## Conditional Response Headers

The [Header](#) directive can use the presence or absence of an environment variable to determine whether or not a certain HTTP header will be placed in the response to the client. This allows, for example, a certain response header to be sent only if a corresponding header is received in the request from the client.

## External Filter Activation

External filters configured by [mod\\_ext\\_filter](#) using the [ExtFilterDefine](#) directive can be activated conditional on an environment variable using the `disableenv=` and `enableenv=` options.

## URL Rewriting

The `%{ENV:variable}` form of *TestString* in the [RewriteCond](#) allows [mod\\_rewrite](#)'s rewrite engine to make decisions

conditional on environment variables. Note that the variables accessible in mod\_rewrite without the ENV: prefix are not actually environment variables. Rather, they are variables special to mod\_rewrite which cannot be accessed from other modules.



Interoperability problems have led to the introduction of mechanisms to modify the way Apache behaves when talking to particular clients. To make these mechanisms as flexible as possible, they are invoked by defining environment variables, typically with [BrowserMatch](#), though [SetEnv](#) and [PassEnv](#) could also be used, for example.

### **downgrade-1.0**

This forces the request to be treated as a HTTP/1.0 request even if it was in a later dialect.

### **force-no-vary**

This causes any Vary fields to be removed from the response header before it is sent back to the client. Some clients don't interpret this field correctly (see the [known client problems](#) page); setting this variable can work around this problem. Setting this variable also implies **force-response-1.0**.

### **force-response-1.0**

This forces an HTTP/1.0 response to clients making an HTTP/1.0 request. It was originally implemented as a result of a problem with AOL's proxies. Some HTTP/1.0 clients may not behave correctly when given an HTTP/1.1 response, and this can be used to interoperate with them.

### **gzip-only-text/html**

When set to a value of "1", this variable disables the DEFLATE output filter provided by [mod\\_deflate](#) for content-types other than text/html.

### **no-gzip**

When set, the DEFLATE filter of [mod\\_deflate](#) will be turned off.

## **nokeepalive**

This disables [KeepAlive](#) when set.

## **prefer-language**

This influences [mod\\_negotiation](#)'s behaviour. If it contains a language tag (such as en, ja or x-klïngon), [mod\\_negotiation](#) tries to deliver a variant with that language. If there's no such variant, the normal [negotiation](#) process applies.

## **redirect-carefully**

This forces the server to be more careful when sending a redirect to the client. This is typically used when a client has a known problem handling redirects. This was originally implemented as a result of a problem with Microsoft's WebFolders software which has a problem handling redirects on directory resources via DAV methods.

## **suppress-error-charset**

*Available in versions after 2.0.54*

When Apache issues a redirect in response to a client request, the response includes some actual text to be displayed in case the client can't (or doesn't) automatically follow the redirection. Apache ordinarily labels this text according to the character set which it uses, which is ISO-8859-1.

However, if the redirection is to a page that uses a different character set, some broken browser versions will try to use the character set from the redirection text rather than the actual page. This can result in Greek, for instance, being incorrectly rendered.

Setting this environment variable causes Apache to omit the character set for the redirection text, and these broken browsers will then correctly use that of the destination page.



## Changing protocol behavior with misbehaving clients

We recommend that the following lines be included in `httpd.conf` to deal with known client problems.

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

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

## Do not log requests for images in the access log

This example keeps requests for images from appearing in the access log. It can be easily modified to prevent logging of particular directories, or to prevent logging of requests coming from particular hosts.

```
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
```

## Prevent "Image Theft"

This example shows how to keep people not on your server from

using images on your server as inline-images on their pages. This is not a recommended configuration, but it can work in limited circumstances. We assume that all your images are in a directory called /web/images.

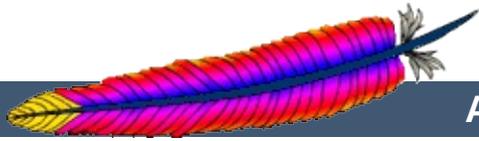
```
SetEnvIf Referer "^http://www\.example\.com/" local_referal
# Allow browsers that do not send Referer info
SetEnvIf Referer "^$" local_referal
<Directory /web/images>
    Order Deny,Allow
    Deny from all
    Allow from env=local_referal
</Directory>
```

For more information about this technique, see the "[Keeping Your Images from Adorning Other Sites](#)" tutorial on ServerWatch.

---

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

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



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



## Apache HTTP Server Version 2.0

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

# Apache's Handler Use

This document describes the use of Apache's Handlers.



Related Modules	Related Directives
<a href="#">mod_actions</a>	<a href="#">Action</a>
<a href="#">mod_asis</a>	<a href="#">AddHandler</a>
<a href="#">mod_cgi</a>	<a href="#">RemoveHandler</a>
<a href="#">mod_imap</a>	<a href="#">SetHandler</a>
<a href="#">mod_info</a>	
<a href="#">mod_mime</a>	
<a href="#">mod_negotiation</a>	
<a href="#">mod_status</a>	

A "handler" is an internal Apache representation of the action to be performed when a file is called. Generally, files have implicit handlers, based on the file type. Normally, all files are simply served by the server, but certain file types are "handled" separately.

Apache 1.1 adds the ability to use handlers explicitly. Based on either filename extensions or on location, handlers can be specified without relation to file type. This is advantageous both because it is a more elegant solution, and because it also allows for both a type **and** a handler to be associated with a file. (See also [Files with Multiple Extensions](#).)

Handlers can either be built into the server or included in a module, or they can be added with the [Action](#) directive. The built-in handlers in the standard distribution are as follows:

- **default-handler:** Send the file using the `default_handler()`, which is the handler used by default to handle static content. (core)
- **send-as-is:** Send file with HTTP headers as is. ([mod\\_asis](#))
- **cgi-script:** Treat the file as a CGI script. ([mod\\_cgi](#))

- **imap-file**: Parse as an imagemap rule file. ([mod\\_imap](#))
- **server-info**: Get the server's configuration information. ([mod\\_info](#))
- **server-status**: Get the server's status report. ([mod\\_status](#))
- **type-map**: Parse as a type map file for content negotiation. ([mod\\_negotiation](#))



## Modifying static content using a CGI script

The following directives will cause requests for files with the `html` extension to trigger the launch of the `footer.pl` CGI script.

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

Then the CGI script is responsible for sending the originally requested document (pointed to by the `PATH_TRANSLATED` environment variable) and making whatever modifications or additions are desired.

## Files with HTTP headers

The following directives will enable the `send-as-is` handler, which is used for files which contain their own HTTP headers. All files in the `/web/htdocs/asis/` directory will be processed by the `send-as-is` handler, regardless of their filename extensions.

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



In order to implement the handler features, an addition has been made to the [Apache API](#) that you may wish to make use of. Specifically, a new record has been added to the `request_rec` structure:

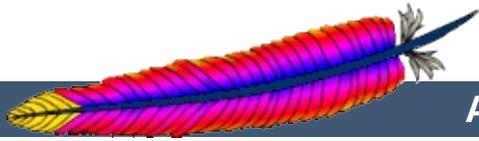
```
char *handler
```

If you wish to have your module engage a handler, you need only to set `r->handler` to the name of the handler at any time prior to the `invoke_handler` stage of the request. Handlers are implemented as they were before, albeit using the handler name instead of a content type. While it is not necessary, the naming convention for handlers is to use a dash-separated word, with no slashes, so as to not invade the media type name-space.

---

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

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



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



## Apache HTTP Server Version 2.0

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

# Filters

This document describes the use of filters in Apache.



Related Modules	Related Directives
<a href="#">mod_deflate</a>	<a href="#">AddInputFilter</a>
<a href="#">mod_ext_filter</a>	<a href="#">AddOutputFilter</a>
<a href="#">mod_include</a>	<a href="#">RemoveInputFilter</a>
	<a href="#">RemoveOutputFilter</a>
	<a href="#">ExtFilterDefine</a>
	<a href="#">ExtFilterOptions</a>
	<a href="#">SetInputFilter</a>
	<a href="#">SetOutputFilter</a>

A *filter* is a process that is applied to data that is sent or received by the server. Data sent by clients to the server is processed by *input filters* while data sent by the server to the client is processed by *output filters*. Multiple filters can be applied to the data, and the order of the filters can be explicitly specified.

Filters are used internally by Apache to perform functions such as chunking and byte-range request handling. In addition, modules can provide filters that are selectable using run-time configuration directives. The set of filters that apply to data can be manipulated with the [SetInputFilter](#), [SetOutputFilter](#), [AddInputFilter](#), [AddOutputFilter](#), [RemoveInputFilter](#), and [RemoveOutputFilter](#) directives.

The following user-selectable filters are currently provided with the Apache HTTP Server distribution.

## INCLUDES

Server-Side Includes processing by [mod\\_include](#)

## DEFLATE

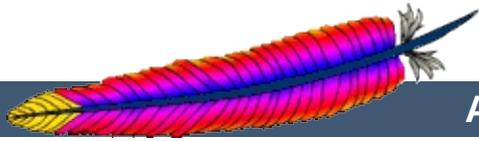
Compress output before sending it to the client using [mod\\_deflate](#)

In addition, the module [mod\\_ext\\_filter](#) allows for external programs to be defined as filters.

---

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

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



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



## Apache HTTP Server Version 2.0

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

## suEXEC Support

The **suEXEC** feature provides Apache users the ability to run **CGI** and **SSI** programs under user IDs different from the user ID of the calling web server. Normally, when a CGI or SSI program executes, it runs as the same user who is running the web server.

Used properly, this feature can reduce considerably the security risks involved with allowing users to develop and run private CGI or SSI programs. However, if suEXEC is improperly configured, it can cause any number of problems and possibly create new holes in your computer's security. If you aren't familiar with managing *setuid root* programs and the security issues they present, we highly recommend that you not consider using suEXEC.



## Before We Begin

Before jumping head-first into this document, you should be aware of the assumptions made on the part of the Apache Group and this document.

First, it is assumed that you are using a UNIX derivative operating system that is capable of **setuid** and **setgid** operations. All command examples are given in this regard. Other platforms, if they are capable of supporting suEXEC, may differ in their configuration.

Second, it is assumed you are familiar with some basic concepts of your computer's security and its administration. This involves an understanding of **setuid/setgid** operations and the various effects they may have on your system and its level of security.

Third, it is assumed that you are using an **unmodified** version of suEXEC code. All code for suEXEC has been carefully scrutinized and tested by the developers as well as numerous beta testers. Every precaution has been taken to ensure a simple yet solidly safe base of code. Altering this code can cause unexpected problems and new security risks. It is **highly** recommended you not alter the suEXEC code unless you are well versed in the particulars of security programming and are willing to share your work with the Apache Group for consideration.

Fourth, and last, it has been the decision of the Apache Group to **NOT** make suEXEC part of the default installation of Apache. To this end, suEXEC configuration requires of the administrator careful attention to details. After due consideration has been given to the various settings for suEXEC, the administrator may install suEXEC through normal installation methods. The values for these settings need to be carefully determined and specified by the administrator to properly maintain system security during the use of suEXEC functionality. It is through this detailed process that the

Apache Group hopes to limit suEXEC installation only to those who are careful and determined enough to use it.

Still with us? Yes? Good. Let's move on!



Before we begin configuring and installing suEXEC, we will first discuss the security model you are about to implement. By doing so, you may better understand what exactly is going on inside suEXEC and what precautions are taken to ensure your system's security.

**suEXEC** is based on a setuid "wrapper" program that is called by the main Apache web server. This wrapper is called when an HTTP request is made for a CGI or SSI program that the administrator has designated to run as a userid other than that of the main server. When such a request is made, Apache provides the suEXEC wrapper with the program's name and the user and group IDs under which the program is to execute.

The wrapper then employs the following process to determine success or failure -- if any one of these conditions fail, the program logs the failure and exits with an error, otherwise it will continue:

- 1. Is the user executing this wrapper a valid user of this system?**

This is to ensure that the user executing the wrapper is truly a user of the system.

- 2. Was the wrapper called with the proper number of arguments?**

The wrapper will only execute if it is given the proper number of arguments. The proper argument format is known to the Apache web server. If the wrapper is not receiving the proper number of arguments, it is either being hacked, or there is something wrong with the suEXEC portion of your Apache binary.

- 3. Is this valid user allowed to run the wrapper?**

Is this user the user allowed to run this wrapper? Only one user (the Apache user) is allowed to execute this program.

**4. Does the target CGI or SSI program have an unsafe hierarchical reference?**

Does the target CGI or SSI program's path contain a leading '/' or have a '..' backreference? These are not allowed; the target CGI/SSI program must reside within suEXEC's document root (see `--with-suexec-docroot=DIR` below).

**5. Is the target user name valid?**

Does the target user exist?

**6. Is the target group name valid?**

Does the target group exist?

**7. Is the target user *NOT* superuser?**

Presently, suEXEC does not allow *root* to execute CGI/SSI programs.

**8. Is the target userid *ABOVE* the minimum ID number?**

The minimum user ID number is specified during configuration. This allows you to set the lowest possible userid that will be allowed to execute CGI/SSI programs. This is useful to block out "system" accounts.

**9. Is the target group *NOT* the superuser group?**

Presently, suEXEC does not allow the *root* group to execute CGI/SSI programs.

**10. Is the target groupid *ABOVE* the minimum ID number?**

The minimum group ID number is specified during configuration. This allows you to set the lowest possible groupid that will be allowed to execute CGI/SSI programs. This is useful to block out "system" groups.

**11. Can the wrapper successfully become the target user and group?**

Here is where the program becomes the target user and group via setuid and setgid calls. The group access list is also initialized with all of the groups of which the user is a member.

**12. Can we change directory to the one in which the target CGI/SSI program resides?**

If it doesn't exist, it can't very well contain files. If we can't change directory to it, it might aswell not exist.

**13. Is the directory within the Apache webospace?**

If the request is for a regular portion of the server, is the requested directory within suEXEC's document root? If the request is for a [UserDir](#), is the requested directory within the directory configured as suEXEC's userdir (see [suEXEC's configuration options](#))?

**14. Is the directory *NOT* writable by anyone else?**

We don't want to open up the directory to others; only the owner user may be able to alter this directories contents.

**15. Does the target CGI/SSI program exist?**

If it doesn't exists, it can't very well be executed.

**16. Is the target CGI/SSI program *NOT* writable by anyone else?**

We don't want to give anyone other than the owner the ability to change the CGI/SSI program.

**17. Is the target CGI/SSI program *NOT* setuid or setgid?**

We do not want to execute programs that will then change our UID/GID again.

**18. Is the target user/group the same as the program's user/group?**

Is the user the owner of the file?

**19. Can we successfully clean the process environment to ensure safe operations?**

suEXEC cleans the process' environment by establishing a safe execution PATH (defined during configuration), as well as only passing through those variables whose names are listed in the safe environment list (also created during configuration).

**20. Can we successfully become the target CGI/SSI program and execute?**

Here is where suEXEC ends and the target CGI/SSI program begins.

This is the standard operation of the suEXEC wrapper's security model. It is somewhat stringent and can impose new limitations and guidelines for CGI/SSI design, but it was developed carefully step-by-step with security in mind.

For more information as to how this security model can limit your

possibilities in regards to server configuration, as well as what security risks can be avoided with a proper suEXEC setup, see the ["Beware the Jabberwock"](#) section of this document.



Here's where we begin the fun.

## suEXEC configuration options

### **--enable-suexec**

This option enables the suEXEC feature which is never installed or activated by default. At least one `--with-suexec-xxxxx` option has to be provided together with the `--enable-suexec` option to let APACI accept your request for using the suEXEC feature.

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

The path to the suexec binary must be hard-coded in the server for security reasons. Use this option to override the default path. e.g. `--with-suexec-bin=/usr/bin/suexec`

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

The [username](#) under which Apache normally runs. This is the only user allowed to execute this program.

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

Define to be the subdirectory under users' home directories where suEXEC access should be allowed. All executables under this directory will be executable by suEXEC as the user so they should be "safe" programs. If you are using a "simple" [UserDir](#) directive (ie. one without a "\*" in it) this should be set to the same value. suEXEC will not work properly in cases where the [UserDir](#) directive points to a location that is not the same as the user's home directory as referenced in the passwd file. Default value is "public\_html".

If you have virtual hosts with a different [UserDir](#) for each, you will need to define them to all reside in one parent directory; then name that parent directory here. **If this is not defined properly, "~userdir" cgi requests will not work!**

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

Define as the DocumentRoot set for Apache. This will be the only hierarchy (aside from [UserDirs](#)) that can be used for suEXEC behavior. The default directory is the --datadir value with the suffix "/htdocs", e.g. if you configure with "--datadir=/home/apache" the directory "/home/apache/htdocs" is used as document root for the suEXEC wrapper.

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

Define this as the lowest UID allowed to be a target user for suEXEC. For most systems, 500 or 100 is common. Default value is 100.

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

Define this as the lowest GID allowed to be a target group for suEXEC. For most systems, 100 is common and therefore used as default value.

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

This defines the filename to which all suEXEC transactions and errors are logged (useful for auditing and debugging purposes). By default the logfile is named "suexec\_log" and located in your standard logfile directory (--logfiledir).

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

Define a safe PATH environment to pass to CGI executables. Default value is "/usr/local/bin:/usr/bin:/bin".

## Compiling and installing the suEXEC wrapper

If you have enabled the suEXEC feature with the --enable-suexec option the suexec binary (together with Apache itself) is automatically built if you execute the make command.

After all components have been built you can execute the

command `make install` to install them. The binary image `suexec` is installed in the directory defined by the `--sbindir` option. The default location is `"/usr/local/apache2/bin/suexec"`.

Please note that you need **root privileges** for the installation step. In order for the wrapper to set the user ID, it must be installed as owner `root` and must have the `setuserid` execution bit set for file modes.

## Setting paranoid permissions

Although the `suEXEC` wrapper will check to ensure that its caller is the correct user as specified with the `--with-suexec-caller` [configure](#) option, there is always the possibility that a system or library call `suEXEC` uses before this check may be exploitable on your system. To counter this, and because it is best-practise in general, you should use filesystem permissions to ensure that only the group Apache runs as may execute `suEXEC`.

If for example, your web server is configured to run as:

```
User www
Group webgroup
```

and [suexec](#) is installed at `"/usr/local/apache2/bin/suexec"`, you should run:

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

This will ensure that only the group Apache runs as can even execute the `suEXEC` wrapper.



## Enabling and Disabling suEXEC

Upon startup of Apache, it looks for the file `suexec` in the directory defined by the `--sbindir` option (default is `"/usr/local/apache/bin/suexec"`). If Apache finds a properly configured suEXEC wrapper, it will print the following message to the error log:

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

If you don't see this message at server startup, the server is most likely not finding the wrapper program where it expects it, or the executable is not installed *setuid root*.

If you want to enable the suEXEC mechanism for the first time and an Apache server is already running you must kill and restart Apache. Restarting it with a simple HUP or USR1 signal will not be enough.

If you want to disable suEXEC you should kill and restart Apache after you have removed the `suexec` file.



Requests for CGI programs will call the suEXEC wrapper only if they are for a virtual host containing a [SuexecUserGroup](#) directive or if they are processed by [mod\\_userdir](#).

### **Virtual Hosts:**

One way to use the suEXEC wrapper is through the [SuexecUserGroup](#) directive in [VirtualHost](#) definitions. By setting this directive to values different from the main server user ID, all requests for CGI resources will be executed as the *User* and *Group* defined for that [<VirtualHost>](#). If this directive is not specified for a [<VirtualHost>](#) then the main server userid is assumed.

### **User directories:**

Requests that are processed by [mod\\_userdir](#) will call the suEXEC wrapper to execute CGI programs under the userid of the requested user directory. The only requirement needed for this feature to work is for CGI execution to be enabled for the user and that the script must meet the scrutiny of the [security checks](#) above. See also the `--with-suexec-userdir` [compile time option](#).



## Logging output

The suEXEC wrapper will write log information to the file defined with the `--with-suexec-logfile` option as indicated above. If you feel you have configured and installed the wrapper properly, have a look at this log and the `error_log` for the server to see where you may have gone astray.



**NOTE!** This section may not be complete. For the latest revision of this section of the documentation, see the Apache Group's [Online Documentation](#) version.

There are a few points of interest regarding the wrapper that can cause limitations on server setup. Please review these before submitting any "bugs" regarding suEXEC.

- **suEXEC Points Of Interest**
- Hierarchy limitations

For security and efficiency reasons, all suEXEC requests must remain within either a top-level document root for virtual host requests, or one top-level personal document root for userdir requests. For example, if you have four VirtualHosts configured, you would need to structure all of your VHosts' document roots off of one main Apache document hierarchy to take advantage of suEXEC for VirtualHosts. (Example forthcoming.)

- suEXEC's PATH environment variable

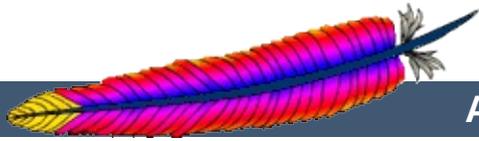
This can be a dangerous thing to change. Make certain every path you include in this define is a **trusted** directory. You don't want to open people up to having someone from across the world running a trojan horse on them.

- Altering the suEXEC code

Again, this can cause **Big Trouble** if you try this without knowing what you are doing. Stay away from it if at all possible.

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

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



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



## Apache HTTP Server Version 2.0

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

## Apache Performance Tuning

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

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



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

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

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

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



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

### HostnameLookups and other DNS considerations

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

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

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

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

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

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

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

## FollowSymLinks and SymLinksIfOwnerMatch

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

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

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

like this:

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

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

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

## AllowOverride

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

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

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

## Negotiation

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

```
DirectoryIndex index
```

Use a complete list of options:

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

where you list the most common choice first.

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

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

## Memory-mapping

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

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

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

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

## Sendfile

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

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

- Some platforms may have broken `sendfile` support that the build system did not detect, especially if the binaries were built on another box and moved to such a machine with broken `sendfile` support.
- With an NFS-mounted files, the kernel may be unable to reliably serve the network file through it's own cache.

For installations where either of these factors applies, you should

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

## Process Creation

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

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

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

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

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

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



## Choosing an MPM

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

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

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

## Modules

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

functions in their absence.

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

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

## Atomic Operations

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

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

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

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

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

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

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

## **accept Serialization - multiple sockets**

**Warning:**

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

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

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

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

But this naive implementation has a serious starvation problem.

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

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

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

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

        FD_ZERO (&accept_fds);
        for (i = first_socket; i <= last_socket; ++i) {
```

```

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

```

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

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

### **AcceptMutex flock**

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

### **AcceptMutex fcntl**

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

### **AcceptMutex sysvsem**

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

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

### **AcceptMutex pthread**

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

### **AcceptMutex posixsem**

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

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

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

the norm.

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

## **accept Serialization - single socket**

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

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

## Lingering Close

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

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

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

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

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

    /* shutdown the sending side */
    shutdown (s, 1);
}
```

```

signal (SIGALRM, lingering_death);
alarm (30);

for (;;) {
    select (s for reading, 2 second timeout);
    if (error) break;
    if (s is ready for reading) {
        if (read (s, junk_buffer, sizeof (junk_buffer)) <= 0) {
            break;
        }
        /* just toss away whatever is here */
    }
}

close (s);
}

```

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

## Scoreboard File

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

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

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

## **DYNAMIC\_MODULE\_LIMIT**

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The worker thread reads the request from the client.

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

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

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

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

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

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

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

The worker thread does a lingering close of the connection.

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

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

and blocks until the listener assigns it another connection.

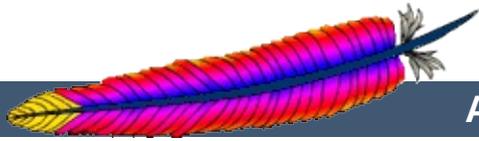
```
/67:    accept(3, 0x001FEB74, 0x001FEB94, 1) (sleeping...)
```

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

# URL Rewriting Guide

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

This document supplements the [mod\\_rewrite reference documentation](#). It describes how one can use Apache's [mod\\_rewrite](#) to solve typical URL-based problems with which webmasters are commonly confronted. We give detailed descriptions on how to solve each problem by configuring URL rewriting rulesets.



---

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

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



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

ATTENTION: Depending on your server-configuration it can be necessary to slightly change the examples for your situation, e.g. adding the [PT] flag when additionally using `mod_alias` and `mod_userdir`, etc. Or rewriting a ruleset to fit in `.htaccess` context instead of per-server context. Always try to understand what a particular ruleset really does before you use it. It avoid problems.



## Canonical URLs

### Description:

On some webservers there are more than one URL for a resource. Usually there are canonical URLs (which should be actually used and distributed) and those which are just shortcuts, internal ones, etc. Independent of which URL the user supplied with the request he should finally see the canonical one only.

### Solution:

We do an external HTTP redirect for all non-canonical URLs to fix them in the location view of the Browser and for all subsequent requests. In the example ruleset below we replace `/~user` by the canonical `/u/user` and fix a missing trailing slash for `/u/user`.

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

## Canonical Hostnames

### Description:

The goal of this rule is to force the use of a particular hostname, in preference to other hostnames which may be used to reach the same site. For example, if you wish to force the use of **www.example.com** instead of **example.com**, you might use a variant of the following recipe.

### Solution:

```
# For sites running on a port other than 80
RewriteCond %{HTTP_HOST} !^www\.example\.com [NC]
RewriteCond %{HTTP_HOST} !^$
```

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

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

## Moved DocumentRoot

### Description:

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

### Solution:

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

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

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

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

## Trailing Slash Problem

## Description:

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

## Solution:

The solution to this subtle problem is to let the server add the trailing slash automatically. To do this correctly we have to use an external redirect, so the browser correctly requests subsequent images etc. If we only did a internal rewrite, this would only work for the directory page, but would go wrong when any images are included into this page with relative URLs, because the browser would request an in-lined object. For instance, a request for `image.gif` in `/~quux/foo/index.html` would become `/~quux/image.gif` without the external redirect!

So, to do this trick we write:

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

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

```
RewriteEngine on
```

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

## Webcluster through Homogeneous URL Layout

### Description:

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

### Solution:

First, the knowledge of the target servers come from (distributed) external maps which contain information where our users, groups and entities stay. They have the form

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

We put them into files `map.xxx-to-host`. Second we need to instruct all servers to redirect URLs of the forms

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

to

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

when the URL is not locally valid to a server. The following ruleset does this for us by the help of the map files (assuming that server0 is a default server which will be used if a user has no entry in the map):

```
RewriteEngine on

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

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

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

## Move Homedirs to Different Webserver

### Description:

Many webmasters have asked for a solution to the following situation: They wanted to redirect just all homedirs on a webserver to another webserver. They usually need such things when establishing a newer webserver which will replace the old one over time.

### Solution:

The solution is trivial with `mod_rewrite`. On the old webserver we just redirect all `/~user/anypath` URLs to

http://newserver/~user/anypath.

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

## Structured Homedirs

### Description:

Some sites with thousands of users usually use a structured homedir layout, i.e. each homedir is in a subdirectory which begins for instance with the first character of the username. So, /~foo/anypath is /home/**f**/foo/.www/anypath while /~bar/anypath is /home/**b**/bar/.www/anypath.

### Solution:

We use the following ruleset to expand the tilde URLs into exactly the above layout.

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

## Filesystem Reorganization

### Description:

This really is a hardcore example: a killer application which heavily uses per-directory RewriteRules to get a smooth look and feel on the Web while its data structure is never touched or adjusted. Background: *net.sw* is my archive of freely available Unix software packages, which I started to collect in 1992. It is both my hobby and job to do this, because while I'm studying computer science I have also worked for many years as a system and network administrator in my spare time. Every week I need some sort of software so I created a deep hierarchy of directories where I stored the

## packages:

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

In July 1996 I decided to make this archive public to the world via a nice Web interface. "Nice" means that I wanted to offer an interface where you can browse directly through the archive hierarchy. And "nice" means that I didn't want to change anything inside this hierarchy - not even by putting some CGI scripts at the top of it. Why? Because the above structure should be later accessible via FTP as well, and I didn't want any Web or CGI stuff to be there.

### Solution:

The solution has two parts: The first is a set of CGI scripts which create all the pages at all directory levels on-the-fly. I put them under `/e/netsw/.www/` as follows:

```
-rw-r--r--  1 netsw  users   1318 Aug  1 18:10 .wwwacl
```

```

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

```

The DATA/ subdirectory holds the above directory structure, i.e. the real *net.sw* stuff and gets automatically updated via *rdist* from time to time. The second part of the problem remains: how to link these two structures together into one smooth-looking URL tree? We want to hide the DATA/ directory from the user while running the appropriate CGI scripts for the various URLs. Here is the solution: first I put the following into the per-directory configuration file in the [DocumentRoot](#) of the server to rewrite the announced URL /net.sw/ to the internal path /e/netsw/:

```

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

```

The first rule is for requests which miss the trailing slash! The second rule does the real thing. And then comes the killer configuration which stays in the per-directory config file /e/netsw/.www/.wwwacl:

```

Options          ExecCGI FollowSymLinks Includes MultiViews

```

```

RewriteEngine on

# we are reached via /net.sw/ prefix
RewriteBase /net.sw/

# first we rewrite the root dir to
# the handling cgi script
RewriteRule ^$ netsw-home.cgi [L]
RewriteRule ^index\.html$ netsw-home.cgi [L]

# strip out the subdirs when
# the browser requests us from perdir pages
RewriteRule ^.+/(netsw-[^\./]+/.)$ $1 [L]

# and now break the rewriting for local files
RewriteRule ^netsw-home\.cgi.* - [L]
RewriteRule ^netsw-changes\.cgi.* - [L]
RewriteRule ^netsw-search\.cgi.* - [L]
RewriteRule ^netsw-tree\.cgi$ - [L]
RewriteRule ^netsw-about\.html$ - [L]
RewriteRule ^netsw-img/.*$ - [L]

# anything else is a subdir which gets handled
# by another cgi script
RewriteRule !^netsw-lsdir\.cgi.* - [C]
RewriteRule (.*) netsw-lsdir.cgi/$1

```

Some hints for interpretation:

1. Notice the L (last) flag and no substitution field ('-') in the forth part
2. Notice the ! (not) character and the C (chain) flag at the first rule in the last part

3. Notice the catch-all pattern in the last rule

## NCSA imagemap to Apache mod\_imap

### Description:

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

### Solution:

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

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

## Search pages in more than one directory

### Description:

Sometimes it is necessary to let the webserver search for pages in more than one directory. Here MultiViews or other techniques cannot help.

### Solution:

We program a explicit ruleset which searches for the files in the directories.

```
RewriteEngine on
```

```

# first try to find it in custom/...
# ...and if found stop and be happy:
RewriteCond          /your/docroot/dir1/{REQUEST_FILENAME}
RewriteRule  ^(.) /your/docroot/dir1/$1  [L]

# second try to find it in pub/...
# ...and if found stop and be happy:
RewriteCond          /your/docroot/dir2/{REQUEST_FILENAME}
RewriteRule  ^(.) /your/docroot/dir2/$1  [L]

# else go on for other Alias or ScriptAlias directives,
# etc.
RewriteRule  ^(.) - [PT]

```

## Set Environment Variables According To URL Parts

### Description:

Perhaps you want to keep status information between requests and use the URL to encode it. But you don't want to use a CGI wrapper for all pages just to strip out this information.

### Solution:

We use a rewrite rule to strip out the status information and remember it via an environment variable which can be later dereferenced from within XSSI or CGI. This way a URL `/foo/S=java/bar/` gets translated to `/foo/bar/` and the environment variable named `STATUS` is set to the value "java".

```

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

```

## Virtual User Hosts

## Description:

Assume that you want to provide `www.username.host.domain.com` for the homepage of `username` via just DNS A records to the same machine and without any virtualhosts on this machine.

## Solution:

For HTTP/1.0 requests there is no solution, but for HTTP/1.1 requests which contain a Host: HTTP header we can use the following ruleset to rewrite `http://www.username.host.com/anypath` internally to `/home/username/anypath`:

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

## Redirect Homedirs For Foreigners

### Description:

We want to redirect homedir URLs to another webserver `www.somewhere.com` when the requesting user does not stay in the local domain `ourdomain.com`. This is sometimes used in virtual host contexts.

### Solution:

Just a rewrite condition:

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

## Redirect Failing URLs To Other Webserver

### Description:

A typical FAQ about URL rewriting is how to redirect failing requests on webserver A to webserver B. Usually this is done via [ErrorDocument](#) CGI-scripts in Perl, but there is also a [mod\\_rewrite](#) solution. But notice that this performs more poorly than using an [ErrorDocument](#) CGI-script!

### Solution:

The first solution has the best performance but less flexibility, and is less error safe:

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

The problem here is that this will only work for pages inside the [DocumentRoot](#). While you can add more Conditions (for instance to also handle homedirs, etc.) there is better variant:

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

This uses the URL look-ahead feature of [mod\\_rewrite](#). The result is that this will work for all types of URLs and is a safe way. But it does a performance impact on the webserver, because for every request there is one more internal subrequest. So, if your webserver runs on a powerful CPU, use this one. If it is a slow machine, use the first approach or better a [ErrorDocument](#) CGI-script.

## Extended Redirection

## Description:

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

## Solution:

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

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

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

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

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

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

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

##EOF##
```

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

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

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

## Archive Access Multiplexer

### Description:

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

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

### Solution:

First we notice that from version 3.0.0 [mod\\_rewrite](#) can also use the "ftp:" scheme on redirects. And second, the location approximation can be done by a [RewriteMap](#) over the top-level domain of the client. With a tricky chained ruleset we can use this top-level domain as a key to our multiplexing map.

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

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

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

## Time-Dependent Rewriting

### Description:

When tricks like time-dependent content should happen a lot of webmasters still use CGI scripts which do for instance redirects to specialized pages. How can it be done via

## mod\_rewrite?

### **Solution:**

There are a lot of variables named TIME\_XXX for rewrite conditions. In conjunction with the special lexicographic comparison patterns <STRING, >STRING and =STRING we can do time-dependent redirects:

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

This provides the content of `foo.day.html` under the URL `foo.html` from `07:00-19:00` and at the remaining time the contents of `foo.night.html`. Just a nice feature for a homepage...

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

### **Description:**

How can we make URLs backward compatible (still existing virtually) after migrating document `.YYYY` to document `.XXXX`, e.g. after translating a bunch of `.html` files to `.phtml`?

### **Solution:**

We just rewrite the name to its basename and test for existence of the new extension. If it exists, we take that name, else we rewrite the URL to its original state.

```
# backward compatibility ruleset for
# rewriting document.html to document.phtml
```

```
# when and only when document.phtml exists
# but no longer document.html
RewriteEngine on
RewriteBase /~quux/
# parse out basename, but remember the fact
RewriteRule ^(.*)\.html$ $1 [C,E=WasHTML]
# rewrite to document.phtml if exists
RewriteCond %{REQUEST_FILENAME}.phtml -f
RewriteRule ^(.*)$ $1.phtml [S=1]
# else reverse the previous basename cutout
RewriteCond %{ENV:WasHTML} ^yes$
RewriteRule ^(.*)$ $1.html
```



## From Old to New (intern)

### Description:

Assume we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. Actually we want that users of the old URL even not recognize that the pages was renamed.

### Solution:

We rewrite the old URL to the new one internally via the following rule:

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

## From Old to New (extern)

### Description:

Assume again that we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. But this time we want that the users of the old URL get hinted to the new one, i.e. their browsers Location field should change, too.

### Solution:

We force a HTTP redirect to the new URL which leads to a change of the browsers and thus the users view:

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

## Browser Dependent Content

### Description:

At least for important top-level pages it is sometimes necessary to provide the optimum of browser dependent content, i.e. one has to provide a maximum version for the latest Netscape variants, a minimum version for the Lynx browsers and a average feature version for all others.

### Solution:

We cannot use content negotiation because the browsers do not provide their type in that form. Instead we have to act on the HTTP header "User-Agent". The following config does the following: If the HTTP header "User-Agent" begins with "Mozilla/3", the page `foo.html` is rewritten to `foo.NS.html` and the rewriting stops. If the browser is "Lynx" or "Mozilla" of version 1 or 2 the URL becomes `foo.20.html`. All other browsers receive page `foo.32.html`. This is done by the following ruleset:

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

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

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

## Dynamic Mirror

### Description:

Assume there are nice webpages on remote hosts we want to bring into our namespace. For FTP servers we would use the `mirror` program which actually maintains an explicit up-to-

date copy of the remote data on the local machine. For a webserver we could use the program webcopy which acts similar via HTTP. But both techniques have one major drawback: The local copy is always just as up-to-date as often we run the program. It would be much better if the mirror is not a static one we have to establish explicitly. Instead we want a dynamic mirror with data which gets updated automatically when there is need (updated data on the remote host).

### **Solution:**

To provide this feature we map the remote webpage or even the complete remote webarea to our namespace by the use of the *Proxy Throughput* feature (flag [P]):

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

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

## **Reverse Dynamic Mirror**

### **Description:**

...

### **Solution:**

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

## Retrieve Missing Data from Intranet

### Description:

This is a tricky way of virtually running a corporate (external) Internet webserver (`www.quux-corp.dom`), while actually keeping and maintaining its data on a (internal) Intranet webserver (`www2.quux-corp.dom`) which is protected by a firewall. The trick is that on the external webserver we retrieve the requested data on-the-fly from the internal one.

### Solution:

First, we have to make sure that our firewall still protects the internal webserver and that only the external webserver is allowed to retrieve data from it. For a packet-filtering firewall we could for instance configure a firewall ruleset like the following:

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

Just adjust it to your actual configuration syntax. Now we can establish the [mod\\_rewrite](#) rules which request the missing data in the background through the proxy throughput feature:

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

## Load Balancing

### Description:

Suppose we want to load balance the traffic to `www.foo.com` over `www[0-5].foo.com` (a total of 6 servers). How can this

be done?

## Solution:

There are a lot of possible solutions for this problem. We will discuss first a commonly known DNS-based variant and then the special one with `mod_rewrite`:

### 1. DNS Round-Robin

The simplest method for load-balancing is to use the DNS round-robin feature of BIND. Here you just configure `www[0-9].foo.com` as usual in your DNS with A(address) records, e.g.

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

Then you additionally add the following entry:

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

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

every time. This way the clients are spread over the various servers. But notice that this not a perfect load balancing scheme, because DNS resolve information gets cached by the other nameservers on the net, so once a client has resolved `www.foo.com` to a particular `wwwN.foo.com`, all subsequent requests also go to this particular name `wwwN.foo.com`. But the final result is ok, because the total sum of the requests are really spread over the various web servers.

## 2. DNS Load-Balancing

A sophisticated DNS-based method for load-balancing is to use the program `lbname` which can be found at <http://www.stanford.edu/~schemers/docs/lbname/lbname>. It is a Perl 5 program in conjunction with auxiliary tools which provides a real load-balancing for DNS.

## 3. Proxy Throughput Round-Robin

In this variant we use `mod_rewrite` and its proxy throughput feature. First we dedicate `www0.foo.com` to be actually `www.foo.com` by using a single

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

entry in the DNS. Then we convert `www0.foo.com` to a proxy-only server, i.e. we configure this machine so all arriving URLs are just pushed through the internal proxy to one of the 5 other servers (`www1-www5`). To accomplish this we first establish a ruleset which contacts a load balancing script `lb.pl` for all URLs.

```
RewriteEngine on
RewriteMap    lb      prg:/path/to/lb.pl
```

```
RewriteRule ^/(.+)$ ${lb:$1} [P,L]
```

Then we write `lb.pl`:

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

$| = 1;

$name = "www"; # the hostname base
$first = 1; # the first server (not 0 here, bec
$last = 5; # the last server in the round-robi
$domain = "foo.dom"; # the domainname

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

##EOF##
```

A last notice: Why is this useful? Seems like `www0.foo.com` still is overloaded? The answer is yes, it is overloaded, but with plain proxy throughput requests, only! All SSI, CGI, ePerl, etc. processing is completely done on the other machines. This is the essential point.

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

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

## New MIME-type, New Service

### Description:

On the net there are a lot of nifty CGI programs. But their usage is usually boring, so a lot of webmaster don't use them. Even Apache's Action handler feature for MIME-types is only appropriate when the CGI programs don't need special URLs (actually PATH\_INFO and QUERY\_STRINGS) as their input. First, let us configure a new file type with extension `.scgi` (for secure CGI) which will be processed by the popular `cgiwrap` program. The problem here is that for instance we use a Homogeneous URL Layout (see above) a file inside the user homedirs has the URL `/u/user/foo/bar.scgi`. But `cgiwrap` needs the URL in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uqe]/([^\s/]+)/\.\www/(.+)\.scgi(.*?) ...  
... /internal/cgi/user/cgiwrap/~$1/$2.scgi$3 [NS,T=applicat
```

Or assume we have some more nifty programs: `wwwlog` (which displays the `access.log` for a URL subtree and `wwwidx` (which runs Glimpse on a URL subtree). We have to provide the URL area to these programs so they know on which area they have to act on. But usually this ugly, because they are all the times still requested from that areas, i.e.

typically we would run the `swwidx` program from within `/u/user/foo/` via hyperlink to

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

which is ugly. Because we have to hard-code **both** the location of the area **and** the location of the CGI inside the hyperlink. When we have to reorganize the area, we spend a lot of time changing the various hyperlinks.

### Solution:

The solution here is to provide a special new URL format which automatically leads to the proper CGI invocation. We configure the following:

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

Now the hyperlink to search at `/u/user/foo/` reads only

```
HREF=" * "
```

which internally gets automatically transformed to

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

The same approach leads to an invocation for the access log CGI program when the hyperlink `:log` gets used.

## From Static to Dynamic

### Description:

How can we transform a static page `foo.html` into a

dynamic variant `foo.cgi` in a seamless way, i.e. without notice by the browser/user.

### **Solution:**

We just rewrite the URL to the CGI-script and force the correct MIME-type so it gets really run as a CGI-script. This way a request to `/~quux/foo.html` internally leads to the invocation of `/~quux/foo.cgi`.

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

## **On-the-fly Content-Regeneration**

### **Description:**

Here comes a really esoteric feature: Dynamically generated but statically served pages, i.e. pages should be delivered as pure static pages (read from the filesystem and just passed through), but they have to be generated dynamically by the webserver if missing. This way you can have CGI-generated pages which are statically served unless one (or a cronjob) removes the static contents. Then the contents gets refreshed.

### **Solution:**

This is done via the following ruleset:

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

Here a request to `page.html` leads to a internal run of a corresponding `page.cgi` if `page.html` is still missing or has filesize null. The trick here is that `page.cgi` is a usual CGI

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

## Document With Autorefresh

### Description:

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

### Solution:

No! We just combine the MIME multipart feature, the webserver NPH feature and the URL manipulation power of [mod\\_rewrite](#). First, we establish a new URL feature: Adding just `:refresh` to any URL causes this to be refreshed every time it gets updated on the filesystem.

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

Now when we reference the URL

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

this leads to the internal invocation of the URL

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

The only missing part is the NPH-CGI script. Although one would usually say "left as an exercise to the reader" ;-) I will provide this, too.

---

```

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

#  split the QUERY_STRING variable
@pairs = split(/&/, $ENV{'QUERY_STRING'});
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))
    eval "\$$name = \"\$value\"";
}
$QS_s = 1 if ($QS_s eq '');
$QS_n = 3600 if ($QS_n eq '');
if ($QS_f eq '') {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;; No file given\n";
    exit(0);
}
if (! -f $QS_f) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;; File $QS_f not found\n";
    exit(0);
}

sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
}

```

```
    print "Content-type: multipart/x-mixed-replace;boundary="
    &print_http_headers_multipart_next;
}

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

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

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

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

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

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

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

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ($n = 0; $n < $QS_n; $n++) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ($mtime ne $mtimeL) {
            $mtimeL = $mtime;
            sleep(2);
            $buffer = &readfile($QS_f);
            &print_http_headers_multipart_next;
            &displayhtml($buffer);
            sleep(5);
            $mtimeL = &mystat($QS_f);
            last;
        }
        sleep($QS_s);
    }
}

&print_http_headers_multipart_end;

exit(0);

##EOF##
```

## Mass Virtual Hosting

### Description:

The `<VirtualHost>` feature of Apache is nice and works great when you just have a few dozens virtual hosts. But when you are an ISP and have hundreds of virtual hosts to provide this feature is not the best choice.

### Solution:

To provide this feature we map the remote webpage or even the complete remote webarea to our namespace by the use of the *Proxy Throughput* feature (flag [P]):

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

```
##
## httpd.conf
##
:
# use the canonical hostname on redirects, etc.
UseCanonicalName on

:
# add the virtual host in front of the CLF-format
CustomLog /path/to/access_log "%{VHOST}e %h %l %u %t \"%r\
:

# enable the rewriting engine in the main server
```

RewriteEngine on

```
# define two maps: one for fixing the URL and one which de
# the available virtual hosts with their corresponding
# DocumentRoot.
```

```
RewriteMap lowercase int:tolower
```

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

```
# Now do the actual virtual host mapping
# via a huge and complicated single rule:
#
```

```
# 1. make sure we don't map for common locations
```

```
RewriteCond %{REQUEST_URI} !^/commonurl1/.*
```

```
RewriteCond %{REQUEST_URI} !^/commonurl2/.*
```

```
:
```

```
RewriteCond %{REQUEST_URI} !^/commonurlN/.*
```

```
#
```

```
# 2. make sure we have a Host header, because
# currently our approach only supports
# virtual hosting through this header
```

```
RewriteCond %{HTTP_HOST} !^$
```

```
#
```

```
# 3. lowercase the hostname
```

```
RewriteCond ${lowercase:%{HTTP_HOST}|NONE} ^(.+)$
```

```
#
```

```
# 4. lookup this hostname in vhost.map and
# remember it only when it is a path
# (and not "NONE" from above)
```

```
RewriteCond ${vhost:%1} ^(/.*)$
```

```
#
```

```
# 5. finally we can map the URL to its docroot location
# and remember the virtual host for logging puposes
```

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

```
:
```



## Blocking of Robots

### Description:

How can we block a really annoying robot from retrieving pages of a specific webarea? A `/robots.txt` file containing entries of the "Robot Exclusion Protocol" is typically not enough to get rid of such a robot.

### Solution:

We use a ruleset which forbids the URLs of the webarea `/~quux/foo/arc/` (perhaps a very deep directory indexed area where the robot traversal would create big server load). We have to make sure that we forbid access only to the particular robot, i.e. just forbidding the host where the robot runs is not enough. This would block users from this host, too. We accomplish this by also matching the User-Agent HTTP header information.

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

## Blocked Inline-Images

### Description:

Assume we have under `http://www.quux-corp.de/~quux/` some pages with inlined GIF graphics. These graphics are nice, so others directly incorporate them via hyperlinks to their pages. We don't like this practice because it adds useless traffic to our server.

### Solution:

While we cannot 100% protect the images from inclusion, we

can at least restrict the cases where the browser sends a HTTP Referer header.

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

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

## Host Deny

### Description:

How can we forbid a list of externally configured hosts from using our server?

### Solution:

For Apache >= 1.3b6:

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

For Apache <= 1.3b6:

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

```
RewriteRule    !^NOT-FOUND/. * - [F]
RewriteRule    ^NOT-FOUND/(.*)$ /$1
```

```
##
##  hosts.deny
##
##  ATTENTION! This is a map, not a list, even when we treat
##              mod_rewrite parses it for key/value pairs, so
##              dummy value "-" must be present for each entr
##
193.102.180.41 -
bsdti1.sdm.de  -
192.76.162.40 -
```

## Proxy Deny

### Description:

How can we forbid a certain host or even a user of a special host from using the Apache proxy?

### Solution:

We first have to make sure `mod_rewrite` is below(!) `mod_proxy` in the Configuration file when compiling the Apache webserver. This way it gets called *before* `mod_proxy`. Then we configure the following for a host-dependent deny...

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

...and this one for a user@host-dependent deny:

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

## Special Authentication Variant

### Description:

Sometimes a very special authentication is needed, for instance a authentication which checks for a set of explicitly configured users. Only these should receive access and without explicit prompting (which would occur when using the Basic Auth via [mod\\_auth](#)).

### Solution:

We use a list of rewrite conditions to exclude all except our friends:

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

## Referer-based Deflector

### Description:

How can we program a flexible URL Deflector which acts on the "Referer" HTTP header and can be configured with as many referring pages as we like?

### Solution:

Use the following really tricky ruleset...

```
RewriteMap deflector txt:/path/to/deflector.map  
  
RewriteCond %{HTTP_REFERER} !="
```

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

... in conjunction with a corresponding rewrite map:

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

This automatically redirects the request back to the referring page (when "-" is used as the value in the map) or to a specific URL (when an URL is specified in the map as the second argument).



## External Rewriting Engine

### Description:

A FAQ: How can we solve the FOO/BAR/QUUX/etc. problem?  
There seems no solution by the use of `mod_rewrite`...

### Solution:

Use an external `RewriteMap`, i.e. a program which acts like a `RewriteMap`. It is run once on startup of Apache receives the requested URLs on STDIN and has to put the resulting (usually rewritten) URL on STDOUT (same order!).

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

```
#!/path/to/perl

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

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

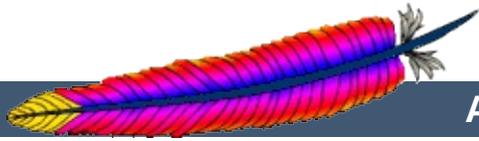
This is a demonstration-only example and just rewrites all URLs `/~quux/foo/...` to `/~quux/bar/...`. Actually you can program whatever you like. But notice that while such

maps can be **used** also by an average user, only the system administrator can **define** it.

---

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

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



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

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

# Unterstützung namensbasierter virtueller Hosts

Das Dokument beschreibt, wann und wie namensbasierte virtuelle Hosts zu verwenden sind.

## Siehe auch

[Unterstützung IP-basierter virtueller Hosts](#)

[Tiefergehende Erörterung der Zuweisung virtueller Hosts](#)

[Dynamisch konfiguriertes Massen-Virtual-Hosting](#)

[Beispiele für virtuelle Hosts in typischen Installationen](#)

[ServerPath-Beispielkonfiguration](#)



## Hosts

IP-basierte virtuelle Hosts verwenden die IP-Adresse der Verbindung, um den korrekten virtuellen Host zur Bedienung einer Anfrage zu ermitteln. Folglich benötigen Sie eine IP-Adresse für jeden virtuellen Host. Bei der Verwendung von namensbasierten virtuellen Hosts verläßt sich der Server darauf, dass der Client den Hostnamen als Bestandteil der HTTP-Header angibt. Durch Anwendung dieser Technik können sich mehrere verschiedene Hosts die gleiche IP-Adresse teilen.

Die Verwendung von namensbasierten virtuellen Hosts ist gewöhnlich einfacher. Sie müssen lediglich Ihren DNS-Server darauf einstellen, jeden Hostnamen auf die richtige IP-Adresse abzubilden, und dann den Apache HTTP Server so konfigurieren, dass er die verschiedenen Hostnamen erkennt. Namensbasierte virtuelle Hosts entschärfen auch den Bedarf an knappen IP-Adressen. Daher sollten Sie namensbasierte virtuelle Hosts verwenden, sofern kein besonderer Grund dafür existiert, IP-basierte virtuelle Hosts zu wählen. Mögliche Gründe für die Verwendung IP-basierter virtueller Hosts sind:

- Einige antike Clients sind nicht kompatibel zu namensbasierten virtuellen Hosts. Damit namensbasierte virtuelle Hosts funktionieren, muss der Client den HTTP-Host-Header senden. Dies ist bei HTTP/1.1 vorgeschrieben und in allen modernen HTTP/1.0-Browsern als Erweiterung implementiert. Wenn Sie Unterstützung für veraltete Clients benötigen und dennoch namensbasierte virtuelle Hosts verwenden, dann finden Sie eine mögliche Lösung dafür am Ende des Dokuments.
- Namensbasierte virtuelle Hosts können aufgrund der Natur des SSL-Protokolls nicht mit SSL-gesicherten Servern verwendet werden.
- Einige Betriebssysteme und Netzwerkanlagen setzen

Techniken zum Bandbreiten-Management ein, die nicht zwischen Hosts unterscheiden können, wenn diese nicht auf verschiedenen IP-Adressen liegen.



## Hosts

Referenzierte Module	Referenzierte Direktiven
<a href="#">core</a>	<a href="#">DocumentRoot</a> <a href="#">NameVirtualHost</a> <a href="#">ServerAlias</a> <a href="#">ServerName</a> <a href="#">ServerPath</a> <a href="#">&lt;VirtualHost&gt;</a>

Um namensbasierte virtuelle Hosts zu verwenden, müssen Sie die IP-Adresse (und möglicherweise den Port) des Servers benennen, an der Anfragen für die Hosts entgegengenommen werden. Dies wird mit der Direktive [NameVirtualHost](#) eingestellt. Im Normalfall, wenn alle IP-Adressen des Server verwendet werden sollen, können Sie `*` als Argument für [NameVirtualHost](#) verwenden. Wenn Sie vorhaben, mehrere Ports zu nutzen (etwa wenn SSL läuft), sollten Sie dem Argument einen Port hinzufügen, wie zum Beispiel `* : 80`. Beachten Sie, dass die Angabe einer IP-Adresse in einer [NameVirtualHost](#)-Anweisung den Server nicht automatisch an dieser Adresse lauschen lässt. Lesen Sie bitte "[Bestimmen der vom Apache verwendeten Adressen und Ports](#)" für weitere Details. Zusätzlich muss jede hier angegebene IP-Adresse einer Netzwerkkarte des Servers zugeordnet sein.

Der nächste Schritt ist die Erstellung eines [<VirtualHost>](#)-Blocks für jeden einzelnen Host, den Sie bedienen wollen. Das Argument der Direktive [<VirtualHost>](#) sollte das gleiche sein wie das Argument der [NameVirtualHost](#)-Anweisung (d.h. eine IP-Adresse oder `*` für alle Adressen). Innerhalb jedes [<VirtualHost>](#)-Blocks benötigen Sie zumindestens eine [ServerName](#)-Anweisung, um zu bestimmen, welcher Host bedient wird, und eine [DocumentRoot](#)-Anweisung, um

anzugeben, wo im Dateisystem der Inhalt des Hosts abgelegt ist.

### Der Hauptserver verschwindet

Wenn Sie virtuelle Hosts zu einem bestehenden Webserver hinzufügen, müssen Sie auch einen `<VirtualHost>`-Block für den bestehenden Host (*Anm.d.Ü.:* und bisherigen Hauptserver) erstellen. Die `ServerName`- und `DocumentRoot`-Anweisungen zu diesem virtuellen Host sollten die gleichen sein wie die globalen `ServerName`- und `DocumentRoot`-Anweisungen. Führen Sie diesen virtuellen Host als erstes in der Konfigurationsdatei auf, so dass er als Standard-Host fungiert.

Vorausgesetzt, Sie bedienen z.B. die Domain `www.domain.tld` und möchten den virtuellen Host `www.otherdomain.tld` hinzufügen, welcher auf die gleiche IP-Adresse zeigt. Dann fügen Sie einfach Folgendes der `httpd.conf` hinzu:

```
NameVirtualHost *:80

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

<VirtualHost *:80>
    ServerName www.otherdomain.tld
    DocumentRoot /www/otherdomain
</VirtualHost>
```

Sie können anstelle des `*` bei den beiden Anweisungen `NameVirtualHost` und `<VirtualHost>` alternativ eine eindeutige IP-Adresse angeben. Das kann man beispielsweise machen, um einige namensbasierte virtuelle Hosts auf einer IP-Adresse zu betreiben und entweder IP-basierte oder ein anderes Set von namensbasierten virtuellen Hosts auf einer anderen Adresse.

Viele Server wollen unter mehr als einem Namen erreichbar sein. Die Direktive `ServerAlias`, die innerhalb des `<VirtualHost>`-Abschnittes angegeben wird, ermöglicht dies. Zum Beispiel zeigt die `ServerAlias`-Anweisung in dem ersten `<VirtualHost>`-Block oben an, dass die aufgeführten Namen alternative Namen sind, die man verwenden kann, um das gleiche Webangebot zu erreichen:

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

Anfragen für alle Hosts der Domain `domain.tld` werden von dem virtuellen Host `www.domain.tld` bedient. Die Platzhalter `*` und `?` können anstelle entsprechender Namen verwendet werden. Natürlich können Sie nicht einfach Namen erfinden und diese bei `ServerName` oder `ServerAlias` angeben, Sie müssen zunächst Ihren DNS Server entsprechend konfigurieren, dass er diese Namen auf die mit Ihrem Server verknüpfte IP-Adresse abbildet.

Und schließlich können Sie die Konfiguration der virtuellen Hosts mittels Angabe weiterer Direktiven innerhalb der `<VirtualHost>`-Container feineinstellen. Die meisten Direktiven können in diesen Containern angegeben werden und verändern dann ausschließlich die Konfiguration des entsprechenden virtuellen Hosts. Prüfen Sie den `Kontext` einer Direktive, um herauszufinden, ob eine bestimmte Direktive zulässig ist. Im *Hauptserver-Kontext* (außerhalb der `<VirtualHost>`-Container) definierte Konfigurationsanweisungen werden nur dann angewendet, wenn sie nicht durch Einstellungen des virtuellen Hosts außer Kraft gesetzt wurden.

Wenn nun eine Anfrage eintrifft, prüft der Server zuerst, ob sie eine IP-Adresse verwendet, die der `NameVirtualHost`-Anweisung entspricht. Ist dies der Fall, dann sieht er sich jeden `<VirtualHost>`-Abschnitt mit einer passenden IP-Adresse an

und versucht den einen zu finden, dessen ServerName- oder ServerAlias-Anweisung mit dem gewünschten Hostnamen übereinstimmt. Findet er einen, dann verwendet er die Konfiguration dieses Servers. Wird kein passender virtueller Host gefunden, dann wird **der erste angegebene virtuelle Host** verwendet, dessen IP-Adresse paßt.

Die Folge davon ist, dass der erste aufgeführte virtuelle Host der *Standard*-Virtual-Host ist. Die DocumentRoot-Anweisung des *Hauptservers* wird **niemals** verwendet, wenn eine IP-Adresse mit einer NameVirtualHost-Anweisung übereinstimmt. Wenn Sie eine spezielle Konfiguration für Anfragen angeben möchten, die keinem bestimmten virtuellen Host entsprechen, packen Sie diese Konfiguration einfach in einen <VirtualHost>-Container und führen diesen als erstes in der Konfigurationsdatei auf.



Wie zuvor erwähnt gibt es einige Clients, die nicht die notwendigen Daten senden, mit denen namensbasierte virtuelle Hosts korrekt funktionieren. Diesen Clients werden stets die Seiten des ersten, für diese IP-Adresse aufgeführten virtuellen Hosts gesendet werden (des *primären* namensbasierten virtuellen Hosts).

### Was bedeutet älter?

Beachten Sie bitte, wenn wir von älter sprechen, meinen wir auch älter. Es ist sehr unwahrscheinlich, dass sie einen dieser Browser heutzutage in Verwendung finden werden. Alle aktuellen Browser-Versionen senden den Host-Header, so wie er für namensbasierte virtuelle Hosts benötigt wird.

Mit der Direktive [ServerPath](#) existiert eine mögliche Behelfskonstruktion, obgleich sie etwas schwerfällig ist:

Beispielkonfiguration:

```
NameVirtualHost 111.22.33.44

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

Was bedeutet das? Es bedeutet, dass eine Anfrage für eine mit `/domain` beginnende URI von dem virtuellen Host `www.domain.tld` bedient wird. Dies heißt, dass die Seiten für alle Clients unter `http://www.domain.tld/domain/` abrufbar sind, wenngleich Clients, die den Header `Host:` senden, auch über `http://www.domain.tld/` auf sie zugreifen können.

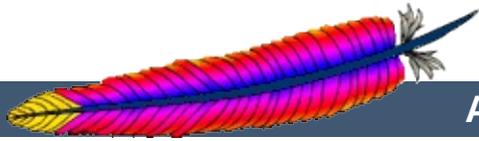
Legen Sie einen Link auf der Seite Ihres primären virtuellen Hosts zu `http://www.domain.tld/domain/`, um die Behelfslösung verfügbar zu machen. Bei den Seiten der virtuellen Hosts müssen Sie dann sicherstellen, entweder ausschließlich relative Links (z.B. `file.html` oder `../icons/image.gif`) zu verwenden oder Links, die das einleitende `/domain/` enthalten (z.B., `http://www.domain.tld/domain/misc/file.html` oder `/domain/misc/file.html`).

Dies erfordert etwas Disziplin, die Befolgung dieser Richtlinien stellt jedoch größtenteils sicher, dass Ihre Seiten mit allen Browsern funktionieren, alten wie neuen.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



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



## Apache HTTP Server Version 2.0

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

# Apache IP-based Virtual Host Support

## See also

[Name-based Virtual Hosts Support](#)



## System Requirements

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



## How to Setup Apache

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

Use multiple daemons when:

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

Use a single daemon when:

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



## Setting up multiple domains

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

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

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



## Setting up a single instance with virtual hosts

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

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

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

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

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

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

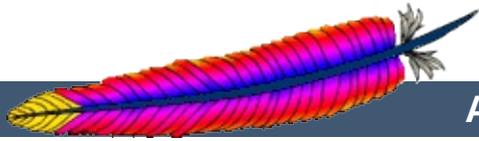
**SECURITY:** When specifying where to write log files, be aware of some security risks which are present if anyone other than the user that starts Apache has write access to the directory where

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

# Dynamically Configured Mass Virtual Hosting

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



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

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

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

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

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

etc.).



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

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

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

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



## Example 2: Name-based Virtual Hosts

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

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

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

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

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



## VirtualHosts for a Homepages System

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

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

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

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



## Server

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

```
UseCanonicalName Off

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

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

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

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

    CustomLog logs/access_log.commercial vcommon

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

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

    CustomLog logs/access_log.homepages vcommon

    VirtualDocumentRoot /www/homepages/%0/docs
    ScriptAlias /cgi-bin/ /www/std-cgi/
</VirtualHost>
```



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

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

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

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



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

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

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

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

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

# now for the hard bit

RewriteEngine On

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

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

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

# that's it!
```



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

```
RewriteEngine on

RewriteMap lowercase int:tolower

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

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

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

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

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



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

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

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

The `httpd.conf` should contain the following:

```
RewriteEngine on

RewriteMap lowercase int:tolower

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

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

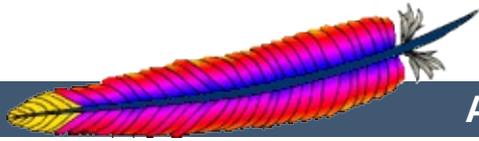
RewriteCond %{REQUEST_URI} ^/cgi-bin/
RewriteCond ${lowercase:%{SERVER_NAME}} ^(.+)$
RewriteCond ${vhost:%1} ^(/.*)$
RewriteRule ^(/.*)$ %1/cgi-bin/$1 [T=application/x-httpd-cgi]
```

---

Copyright 2013 The Apache Software Foundation.

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

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



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



## Apache HTTP Server Version 2.0

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

## VirtualHost Examples

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



## IP address.

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

### Note

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

### Server configuration

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

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

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

    # Other directives here

</VirtualHost>

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

    # Other directives here

</VirtualHost>
```

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

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

### Note

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

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

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

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



## Note

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

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

## Server configuration

```
Listen 80

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

# This is the other address
NameVirtualHost 172.20.30.50

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

    # Other directives here ...
</VirtualHost>

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

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

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



(such as an internal and external address).

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

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

### Server configuration

```
NameVirtualHost 192.168.1.1
NameVirtualHost 172.20.30.40

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

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

### Note:

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

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



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

### Server configuration

```
Listen 80
Listen 8080

NameVirtualHost 172.20.30.40:80
NameVirtualHost 172.20.30.40:8080

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

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

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

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



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

### Server configuration

```
Listen 80

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

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

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



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

### Server configuration

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

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

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

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

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



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

## Server configuration

```
Listen 80

NameVirtualHost 172.20.30.40

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

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

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

# IP-based
<VirtualHost 172.20.30.50>
  DocumentRoot /www/example4
  ServerName www.example4.edu
</VirtualHost>

<VirtualHost 172.20.30.60>
  DocumentRoot /www/example5
  ServerName www.example5.gov
</VirtualHost>
```



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

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



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

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

### **Server configuration**

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

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

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

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

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

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

### **Server configuration**

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

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

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

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

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

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

### **Server configuration**

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

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



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

### Server configuration

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

NameVirtualHost 172.20.30.40

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

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

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



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

### Server configuration

```
NameVirtualHost 172.20.30.40

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

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

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

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

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

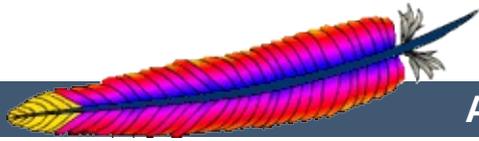
Please note that there is one oddity: A request to `http://www.sub2.domain.tld/sub1/` is also served from the `sub1-vhost` if the client sent no `Host :` header.

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

## An In-Depth Discussion of Virtual Host Matching

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

NameVirtualHost
111.22.33.44
NameVirtualHost
111.22.33.55
```

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

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

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

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

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

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

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

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

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

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

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

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



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

## Hash table lookup

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

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

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

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

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

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

## **Persistent connections**

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

## **Absolute URI**

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

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

## Observations

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

warning in the error logfile when it detects this.

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



**tips**

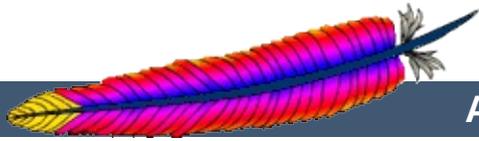
In addition to the tips on the [DNS Issues](#) page, here are some further tips:

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

## File Descriptor Limits

When using a large number of Virtual Hosts, Apache may run out of available file descriptors (sometimes called *file handles*) if each Virtual Host specifies different log files. The total number of file descriptors used by Apache is one for each distinct error log file, one for every other log file directive, plus 10-20 for internal use. Unix operating systems limit the number of file descriptors that may be used by a process; the limit is typically 64, and may usually be increased up to a large hard-limit.

Although Apache attempts to increase the limit as required, this may not work if:

1. Your system does not provide the `setrlimit()` system call.
2. The `setrlimit(RLIMIT_NOFILE)` call does not function on your system (such as Solaris 2.3)
3. The number of file descriptors required exceeds the hard limit.
4. Your system imposes other limits on file descriptors, such as a limit on stdio streams only using file descriptors below 256. (Solaris 2)

In the event of problems you can:

- Reduce the number of log files; don't specify log files in the `<VirtualHost>` sections, but only log to the main log files. (See [Splitting up your log files](#), below, for more information on doing this.)
- If you system falls into 1 or 2 (above), then increase the file descriptor limit before starting Apache, using a script like

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

Please see the [Descriptors and Apache](#) document containing further details about file descriptor problems and how they can be solved on your operating system.



## Splitting up your log files

If you want to log multiple virtual hosts to the same log file, you may want to split up the log files afterwards in order to run statistical analysis of the various virtual hosts. This can be accomplished in the following manner.

First, you will need to add the virtual host information to the log entries. This can be done using the [LogFormat](#) directive, and the `%v` variable. Add this to the beginning of your log format string:

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

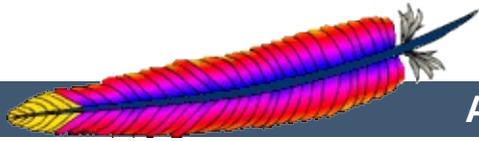
This will create a log file in the common log format, but with the canonical virtual host (whatever appears in the [ServerName](#) directive) prepended to each line. (See [Custom Log Formats](#) for more about customizing your log files.)

When you wish to split your log file into its component parts (one file per virtual host) you can use the program [split-logfile](#) to accomplish this. You'll find this program in the support directory of the Apache distribution.

Run this program with the command:

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

This program, when run with the name of your vhost log file, will generate one file for each virtual host that appears in your log file. Each file will be called `hostname.log`.



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



## Apache HTTP Server Version 2.0

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

## Issues Regarding DNS and Apache

This page could be summarized with the statement: don't configure Apache in such a way that it relies on DNS resolution for parsing of the configuration files. If Apache requires DNS resolution to parse the configuration files then your server may be subject to reliability problems (ie. it might not boot), or denial and theft of service attacks (including users able to steal hits from other users).



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

In order for Apache to function properly, it absolutely needs to have two pieces of information about each virtual host: the [ServerName](#) and at least one IP address that the server will bind and respond to. The above example does not include the IP address, so Apache must use DNS to find the address of `www.abc.dom`. If for some reason DNS is not available at the time your server is parsing its config file, then this virtual host **will not be configured**. It won't be able to respond to any hits to this virtual host (prior to Apache version 1.2 the server would not even boot).

Suppose that `www.abc.dom` has address `10.0.0.1`. Then consider this configuration snippet:

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

This time Apache needs to use reverse DNS to find the `ServerName` for this virtualhost. If that reverse lookup fails then it will partially disable the virtualhost (prior to Apache version 1.2 the server would not even boot). If the virtual host is name-based then it will effectively be totally disabled, but if it is IP-based then it will mostly work. However, if Apache should ever have to generate a full URL for the server which includes the server name, then it will fail to generate a valid URL.

Here is a snippet that avoids both of these problems:

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



There are (at least) two forms that denial of service can come in. If you are running a version of Apache prior to version 1.2 then your server will not even boot if one of the two DNS lookups mentioned above fails for any of your virtual hosts. In some cases this DNS lookup may not even be under your control; for example, if `abc . dom` is one of your customers and they control their own DNS, they can force your (pre-1.2) server to fail while booting simply by deleting the `www . abc . dom` record.

Another form is far more insidious. Consider this configuration snippet:

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

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

Suppose that you've assigned `10.0.0.1` to `www . abc . dom` and `10.0.0.2` to `www . def . dom`. Furthermore, suppose that `def . dom` has control of their own DNS. With this config you have put `def . dom` into a position where they can steal all traffic destined to `abc . dom`. To do so, all they have to do is set `www . def . dom` to `10.0.0.1`. Since they control their own DNS you can't stop them from pointing the `www . def . dom` record wherever they wish.

Requests coming in to `10.0.0.1` (including all those where users typed in URLs of the form `http://www.abc.dom/whatever`) will all be served by the `def . dom` virtual host. To better understand why this happens requires a more in-depth discussion of how Apache matches up incoming requests with the virtual host

that will serve it. A rough document describing this [is available](#).



## THE MAIN SERVER ADDRESS

The addition of [name-based virtual host support](#) in Apache 1.1 requires Apache to know the IP address(es) of the host that [httpd](#) is running on. To get this address it uses either the global [ServerName](#) (if present) or calls the C function `gethostname` (which should return the same as typing "hostname" at the command prompt). Then it performs a DNS lookup on this address. At present there is no way to avoid this lookup.

If you fear that this lookup might fail because your DNS server is down then you can insert the hostname in `/etc/hosts` (where you probably already have it so that the machine can boot properly). Then ensure that your machine is configured to use `/etc/hosts` in the event that DNS fails. Depending on what OS you are using this might be accomplished by editing `/etc/resolv.conf`, or maybe `/etc/nsswitch.conf`.

If your server doesn't have to perform DNS for any other reason then you might be able to get away with running Apache with the `HOSTRESORDER` environment variable set to "local". This all depends on what OS and resolver libraries you are using. It also affects CGIs unless you use [mod\\_env](#) to control the environment. It's best to consult the man pages or FAQs for your OS.



## Tips to Avoid These Problems

- use IP addresses in [VirtualHost](#)
- use IP addresses in [Listen](#)
- ensure all virtual hosts have an explicit [ServerName](#)
- create a `<VirtualHost _default_*>` server that has no pages to serve



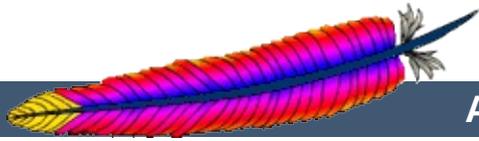
## Appendix: Future Directions

The situation regarding DNS is highly undesirable. For Apache 1.2 we've attempted to make the server at least continue booting in the event of failed DNS, but it might not be the best we can do. In any event, requiring the use of explicit IP addresses in configuration files is highly undesirable in today's Internet where renumbering is a necessity.

A possible work around to the theft of service attack described above would be to perform a reverse DNS lookup on the IP address returned by the forward lookup and compare the two names -- in the event of a mismatch, the virtualhost would be disabled. This would require reverse DNS to be configured properly (which is something that most admins are familiar with because of the common use of "double-reverse" DNS lookups by FTP servers and TCP wrappers).

In any event, it doesn't seem possible to reliably boot a virtual-hosted web server when DNS has failed unless IP addresses are used. Partial solutions such as disabling portions of the configuration might be worse than not booting at all depending on what the webserver is supposed to accomplish.

As HTTP/1.1 is deployed and browsers and proxies start issuing the Host header it will become possible to avoid the use of IP-based virtual hosts entirely. In this case, a webserver has no requirement to do DNS lookups during configuration. But as of March 1997 these features have not been deployed widely enough to be put into use on critical web servers.



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



## Apache HTTP Server Version 2.0

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

## Support - Frequently Asked Questions



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

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

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

### Check the errorlog!

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

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

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

### Check the Apache bug database

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

## Ask in a user support forum

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

[Users mailing list](#)

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

USENET newsgroups:

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

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

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

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

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

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

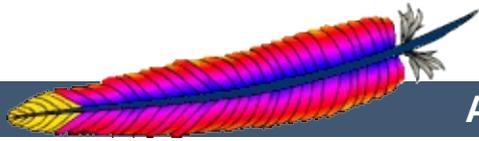
## Whom do I contact for support?

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

# Error Messages - Frequently Asked Questions



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

## **Invalid argument: core\_output\_filter: writing data to the network**

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

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

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

## **AcceptEx Failed**

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

## **Premature end of script headers**

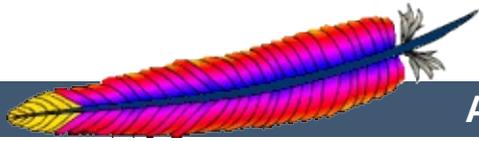
Most problems with CGI scripts result in this message written in the error log together with an `Internal Server Error` delivered to the browser. A guide to helping debug this type of

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

# SSL/TLS Strong Encryption: An Introduction

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

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

As an introduction this chapter is aimed at readers who are familiar with the Web, HTTP, and Apache, but are not security experts. It is not intended to be a definitive guide to the SSL protocol, nor does it discuss specific techniques for managing certificates in an organization, or the important legal issues of patents and import and export restrictions. Rather, it is intended to provide a common background to `mod_ssl` users by pulling together various concepts, definitions, and examples as a starting point for further exploration.

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



Understanding SSL requires an understanding of cryptographic algorithms, message digest functions (aka. one-way or hash functions), and digital signatures. These techniques are the subject of entire books (see for instance [\[AC96\]](#)) and provide the basis for privacy, integrity, and authentication.

### **Cryptographic Algorithms**

Suppose Alice wants to send a message to her bank to transfer some money. Alice would like the message to be private, since it will include information such as her account number and transfer amount. One solution is to use a cryptographic algorithm, a technique that would transform her message into an encrypted form, unreadable except by those it is intended for. Once in this form, the message may only be interpreted through the use of a secret key. Without the key the message is useless: good cryptographic algorithms make it so difficult for intruders to decode the original text that it isn't worth their effort.

There are two categories of cryptographic algorithms: conventional and public key.

#### **Conventional cryptography**

also known as symmetric cryptography, requires the sender and receiver to share a key: a secret piece of information that may be used to encrypt or decrypt a message. If this key is secret, then nobody other than the sender or receiver may read the message. If Alice and the bank know a secret key, then they may send each other private messages. The task of privately choosing a key before communicating, however, can be problematic.

#### **Public key cryptography**

also known as asymmetric cryptography, solves the key exchange problem by defining an algorithm which uses two

keys, each of which may be used to encrypt a message. If one key is used to encrypt a message then the other must be used to decrypt it. This makes it possible to receive secure messages by simply publishing one key (the public key) and keeping the other secret (the private key).

Anyone may encrypt a message using the public key, but only the owner of the private key will be able to read it. In this way, Alice may send private messages to the owner of a key-pair (the bank), by encrypting it using their public key. Only the bank will be able to decrypt it.

## **Message Digests**

Although Alice may encrypt her message to make it private, there is still a concern that someone might modify her original message or substitute it with a different one, in order to transfer the money to themselves, for instance. One way of guaranteeing the integrity of Alice's message is to create a concise summary of her message and send this to the bank as well. Upon receipt of the message, the bank creates its own summary and compares it with the one Alice sent. If they agree then the message was received intact.

A summary such as this is called a *message digest*, *one-way function* or *hash function*. Message digests are used to create short, fixed-length representations of longer, variable-length messages. Digest algorithms are designed to produce unique digests for different messages. Message digests are designed to make it too difficult to determine the message from the digest, and also impossible to find two different messages which create the same digest -- thus eliminating the possibility of substituting one message for another while maintaining the same digest.

Another challenge that Alice faces is finding a way to send the

digest to the bank securely; when this is achieved, the integrity of the associated message is assured. One way to do this is to include the digest in a digital signature.

## Digital Signatures

When Alice sends a message to the bank, the bank needs to ensure that the message is really from her, so an intruder does not request a transaction involving her account. A *digital signature*, created by Alice and included with the message, serves this purpose.

Digital signatures are created by encrypting a digest of the message, and other information (such as a sequence number) with the sender's private key. Though anyone may *decrypt* the signature using the public key, only the signer knows the private key. This means that only they may have signed it. Including the digest in the signature means the signature is only good for that message; it also ensures the integrity of the message since no one can change the digest and still sign it.

To guard against interception and reuse of the signature by an intruder at a later date, the signature contains a unique sequence number. This protects the bank from a fraudulent claim from Alice that she did not send the message -- only she could have signed it (non-repudiation).



Although Alice could have sent a private message to the bank, signed it, and ensured the integrity of the message, she still needs to be sure that she is really communicating with the bank. This means that she needs to be sure that the public key she is using corresponds to the bank's private key. Similarly, the bank also needs to verify that the message signature really corresponds to Alice's signature.

If each party has a certificate which validates the other's identity, confirms the public key, and is signed by a trusted agency, then they both will be assured that they are communicating with whom they think they are. Such a trusted agency is called a *Certificate Authority*, and certificates are used for authentication.

## Certificate Contents

A certificate associates a public key with the real identity of an individual, server, or other entity, known as the subject. As shown in [Table 1](#), information about the subject includes identifying information (the distinguished name), and the public key. It also includes the identification and signature of the Certificate Authority that issued the certificate, and the period of time during which the certificate is valid. It may have additional information (or extensions) as well as administrative information for the Certificate Authority's use, such as a serial number.

**Table 1: Certificate Information**

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

etc.

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

**Table 2: Distinguished Name Information**

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

A Certificate Authority may define a policy specifying which distinguished field names are optional, and which are required. It may also place requirements upon the field contents, as may users of certificates. As an example, a Netscape browser requires that the Common Name for a certificate representing a server has a name which matches a wildcard pattern for the domain name of

that server, such as \*.snakeoil.com.

The binary format of a certificate is defined using the ASN.1 notation [X208] [PKCS]. This notation defines how to specify the contents, and encoding rules define how this information is translated into binary form. The binary encoding of the certificate is defined using Distinguished Encoding Rules (DER), which are based on the more general Basic Encoding Rules (BER). For those transmissions which cannot handle binary, the binary form may be translated into an ASCII form by using Base64 encoding [MIME]. This encoded version is called PEM encoded (the name comes from "Privacy Enhanced Mail"), when placed between begin and end delimiter lines as illustrated in the following example.

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

```
-----BEGIN CERTIFICATE-----
MIIC7jCCAlEgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBqTELMAkGA1UEBhMCWFkx
FTATBgNVBAGTDFNuYWt1IERlc2VydDETMDEGA1UEBxMKU25ha2UgVG93bjEXMBUG
A1UEChM0U25ha2UgT21sLCBMdGQxHjAcBgNVBAsTFUNlcnRpZm1jYXR1IEF1dGhv
cm10eTEVMBMGA1UEAxMMU25ha2UgT21sIENBMR4wHAYJKoZIhvcNAQkBFg9jYUBz
bmFrZW9pbC5kb20wHhcNOTGxMDIxMDg1ODM2WhcNOTkxMDIxMDg1ODM2WjCBpzEL
MAkGA1UEBhMCWFkxFTATBgNVBAGTDFNuYWt1IERlc2VydDETMDEGA1UEBxMKU25h
a2UgVG93bjEXMBUGA1UEChM0U25ha2UgT21sLCBMdGQxHjAcBgNVBAsTD1d1YnNl
cnZlc1BUZWFtMRkwFwYDQDEExB3d3cuc25ha2VvaWwuZG9tMR8wHqYJKoZIhvcN
AQkBFhB3d3dAc25ha2VvaWwuZG9tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKB
gQDH9Ge/s2zcH+da+rPTx/DPRp3xGjHZ4GG6pCmvADIEtBtKBFACZ64n+Dy7Np8b
vKR+yy5DGQiijsH1D/j8H1GE+q4TZ80Fk7BNBFazHxFbYI40KMiCxdKzdif1yfaa
lWoANFlAzlSdbxeGVHoT0K+gT5w3UxwZKv2DLbCTzLZyPwIDAQABoyYwJDAPBgNV
HRMECDAGAQH/AgEAMBEGCWGCSAGG+EIBAQQEAWIAQDANBgkqhkiG9w0BAQQFAA0B
gQAZUIHAL4D09oE6Lv2k56Gp380BDuILvLg1v1KL8mQR+KFjghCrtpqaztZqcDt
2q2QoyulCgSzHbEGmi0EsdkPfg6mp0penssIFePYNI+/8u9HT4LuKMJX15hxBam7
dUHZICxBVC1lnHyYGjDuAMhe396lYAn8bCl1d/L4NMGBCQ==
-----END CERTIFICATE-----
```

## Certificate Authorities

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

identity of the private key owner of a key-pair. For instance, if Alice requests a personal certificate, the Certificate Authority must first make sure that Alice really is the person the certificate request claims.

### **Certificate Chains**

A Certificate Authority may also issue a certificate for another Certificate Authority. When examining a certificate, Alice may need to examine the certificate of the issuer, for each parent Certificate Authority, until reaching one which she has confidence in. She may decide to trust only certificates with a limited chain of issuers, to reduce her risk of a "bad" certificate in the chain.

### **Creating a Root-Level CA**

As noted earlier, each certificate requires an issuer to assert the validity of the identity of the certificate subject, up to the top-level Certificate Authority (CA). This presents a problem: Since this is who vouches for the certificate of the top-level authority, which has no issuer? In this unique case, the certificate is "self-signed", so the issuer of the certificate is the same as the subject. As a result, one must exercise extra care in trusting a self-signed certificate. The wide publication of a public key by the root authority reduces the risk in trusting this key -- it would be obvious if someone else publicized a key claiming to be the authority. Browsers are preconfigured to trust well-known certificate authorities.

A number of companies, such as [Thawte](#) and [VeriSign](#) have established themselves as Certificate Authorities. These companies provide the following services:

- Verifying certificate requests
- Processing certificate requests
- Issuing and managing certificates

It is also possible to create your own Certificate Authority.

Although risky in the Internet environment, it may be useful within an Intranet where the organization can easily verify the identities of individuals and servers.

### **Certificate Management**

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

#### **Note**

If you use a Certificate Authority that is not configured into browsers by default, it is necessary to load the Certificate Authority certificate into the browser, enabling the browser to validate server certificates signed by that Certificate Authority. Doing so may be dangerous, since once loaded, the browser will accept all certificates signed by that Certificate Authority.



The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP). SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity, and encryption for privacy.

The protocol is designed to support a range of choices for specific algorithms used for cryptography, digests, and signatures. This allows algorithm selection for specific servers to be made based on legal, export or other concerns, and also enables the protocol to take advantage of new algorithms. Choices are negotiated between client and server at the start of establishing a protocol session.

**Table 4: Versions of the SSL protocol**

Version	Source	Description	Browser Support
SSL v2.0	Vendor Standard (from Netscape Corp.) <a href="#">[SSL2]</a>	First SSL protocol for which implementations exists	- NS Navigator 1.x/2.x - MS IE 3.x - Lynx/2.8+OpenSSL
SSL v3.0	Expired Internet Draft (from Netscape Corp.) <a href="#">[SSL3]</a>	Revisions to prevent specific security attacks, add non-RSA ciphers, and support for certificate chains	- NS Navigator 2.x/3.x/4.x - MS IE 3.x/4.x - Lynx/2.8+OpenSSL
TLS v1.0	Proposed Internet	Revision of SSL 3.0 to update the MAC layer	- Lynx/2.8+OpenSSL

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

There are a number of versions of the SSL protocol, as shown in [Table 4](#). As noted there, one of the benefits in SSL 3.0 is that it adds support of certificate chain loading. This feature allows a server to pass a server certificate along with issuer certificates to the browser. Chain loading also permits the browser to validate the server certificate, even if Certificate Authority certificates are not installed for the intermediate issuers, since they are included in the certificate chain. SSL 3.0 is the basis for the Transport Layer Security [\[TLS\]](#) protocol standard, currently in development by the Internet Engineering Task Force (IETF).

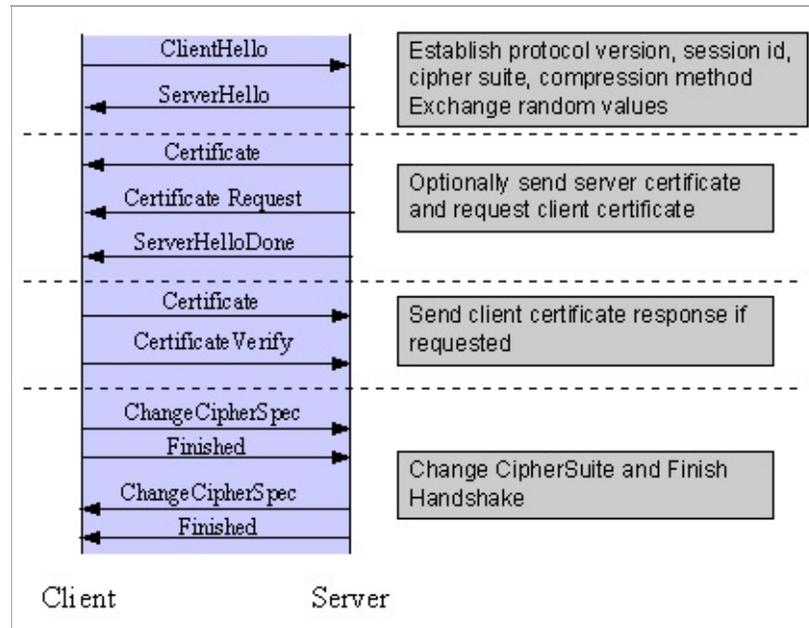
## Session Establishment

The SSL session is established by following a handshake sequence between client and server, as shown in [Figure 1](#). This sequence may vary, depending on whether the server is configured to provide a server certificate or request a client certificate. Though cases exist where additional handshake steps are required for management of cipher information, this article summarizes one common scenario: see the SSL specification for the full range of possibilities.

### Note

Once an SSL session has been established it may be reused, thus avoiding the performance penalty of repeating the many steps needed to start a session. For this the server assigns each SSL session a unique session identifier which is cached in the server and which the client can use on forthcoming

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



**Figure 1:** Simplified SSL Handshake Sequence

The elements of the handshake sequence, as used by the client and server, are listed below:

1. Negotiate the Cipher Suite to be used during data transfer
2. Establish and share a session key between client and server
3. Optionally authenticate the server to the client
4. Optionally authenticate the client to the server

The first step, Cipher Suite Negotiation, allows the client and server to choose a Cipher Suite supportable by both of them. The SSL3.0 protocol specification defines 31 Cipher Suites. A Cipher Suite is defined by the following components:

- Key Exchange Method
- Cipher for Data Transfer
- Message Digest for creating the Message Authentication

Code (MAC)

These three elements are described in the sections that follow.

## Key Exchange Method

The key exchange method defines how the shared secret symmetric cryptography key used for application data transfer will be agreed upon by client and server. SSL 2.0 uses RSA key exchange only, while SSL 3.0 supports a choice of key exchange algorithms including the RSA key exchange when certificates are used, and Diffie-Hellman key exchange for exchanging keys without certificates and without prior communication between client and server.

One variable in the choice of key exchange methods is digital signatures -- whether or not to use them, and if so, what kind of signatures to use. Signing with a private key provides assurance against a man-in-the-middle-attack during the information exchange used in generating the shared key [[AC96](#), p516].

## Cipher for Data Transfer

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

- No encryption
- Stream Ciphers
  - RC4 with 40-bit keys
  - RC4 with 128-bit keys
- CBC Block Ciphers
  - RC2 with 40 bit key
  - DES with 40 bit key

- DES with 56 bit key
- Triple-DES with 168 bit key
- Idea (128 bit key)
- Fortezza (96 bit key)

Here "CBC" refers to Cipher Block Chaining, which means that a portion of the previously encrypted cipher text is used in the encryption of the current block. "DES" refers to the Data Encryption Standard [[AC96](#), ch12], which has a number of variants (including DES40 and 3DES\_EDE). "Idea" is one of the best and cryptographically strongest available algorithms, and "RC2" is a proprietary algorithm from RSA DSI [[AC96](#), ch13].

## Digest Function

The choice of digest function determines how a digest is created from a record unit. SSL supports the following:

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

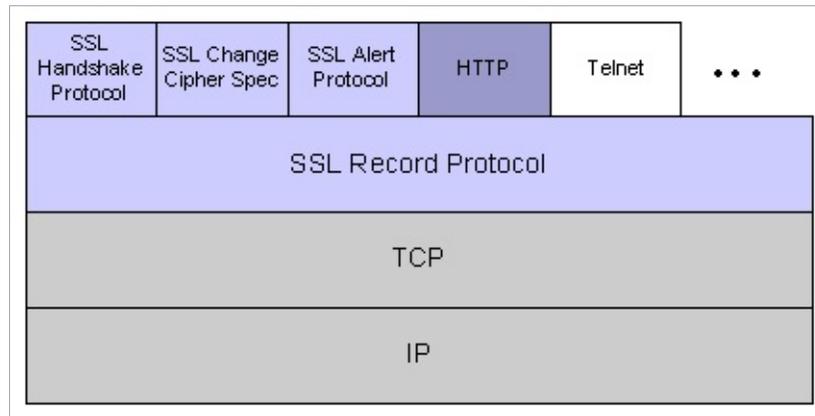
The message digest is used to create a Message Authentication Code (MAC) which is encrypted with the message to provide integrity and to prevent against replay attacks.

## Handshake Sequence Protocol

The handshake sequence uses three protocols:

- The *SSL Handshake Protocol* for performing the client and server SSL session establishment.
- The *SSL Change Cipher Spec Protocol* for actually establishing agreement on the Cipher Suite for the session.
- The *SSL Alert Protocol* for conveying SSL error messages between client and server.

These protocols, as well as application protocol data, are encapsulated in the *SSL Record Protocol*, as shown in [Figure 2](#). An encapsulated protocol is transferred as data by the lower layer protocol, which does not examine the data. The encapsulated protocol has no knowledge of the underlying protocol.

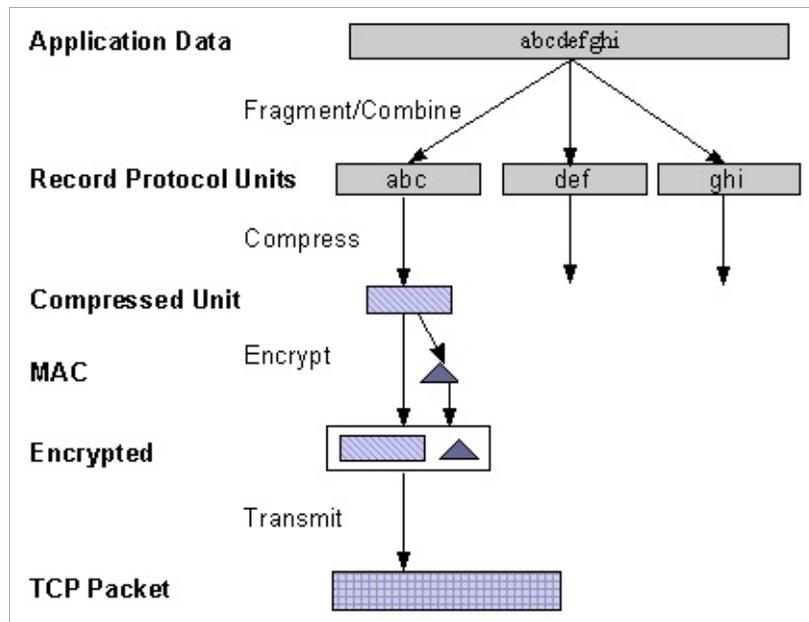


**Figure 2:** *SSL Protocol Stack*

The encapsulation of SSL control protocols by the record protocol means that if an active session is renegotiated the control protocols will be transmitted securely. If there were no session before, then the Null cipher suite is used, which means there is no encryption and messages have no integrity digests until the session has been established.

## Data Transfer

The SSL Record Protocol, shown in [Figure 3](#), is used to transfer application and SSL Control data between the client and server, possibly fragmenting this data into smaller units, or combining multiple higher level protocol data messages into single units. It may compress, attach digest signatures, and encrypt these units before transmitting them using the underlying reliable transport protocol (Note: currently all major SSL implementations lack support for compression).



**Figure 3: SSL Record Protocol**

## Securing HTTP Communication

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



## [AC96]

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

## [X208]

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

## [X509]

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

## [PKCS]

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

## [MIME]

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

## [SSL2]

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

## [SSL3]

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

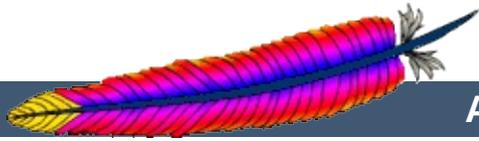
## [TLS1]

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

## SSL/TLS Strong Encryption: Compatibility

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

-- Unknown

Here we talk about backward compatibility to other SSL solutions. As you perhaps know, mod\_ssl is not the only existing SSL solution for Apache. Actually there are four additional major products available on the market: Ben Laurie's freely available [Apache-SSL](#) (from where mod\_ssl were originally derived in 1998), Red Hat's commercial [Secure Web Server](#) (which is based on mod\_ssl), Covalent's commercial [Raven SSL Module](#) (also based on mod\_ssl) and finally C2Net's commercial product [Stronghold](#) (based on a different evolution branch named Sioux up to Stronghold 2.x and based on mod\_ssl since Stronghold 3.x).

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



## Configuration Directives

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

**Table 1: Configuration Directive Mapping**

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

SSLCacheServerRunDir <i>dir</i>	-
<b>Sioux 1.x compatibility:</b>	
SSL_CertFile <i>file</i>	SSLCertificateFile <i>file</i>
SSL_KeyFile <i>file</i>	SSLCertificateKeyFile <i>file</i>
SSL_CipherSuite <i>arg</i>	SSLCipherSuite <i>arg</i>
SSL_X509VerifyDir <i>arg</i>	SSLCACertificatePath <i>arg</i>
SSL_Log <i>file</i>	SSLLogFile <i>file</i>
SSL_Connect <i>flag</i>	SSL Engine <i>flag</i>
SSL_ClientAuth <i>arg</i>	SSLVerifyClient <i>arg</i>
SSL_X509VerifyDepth <i>arg</i>	SSLVerifyDepth <i>arg</i>
SSL_FetchKeyPhraseFrom <i>arg</i>	-
SSL_SessionDir <i>dir</i>	-
SSL_Require <i>expr</i>	-
SSL_CertFileType <i>arg</i>	-
SSL_KeyFileType <i>arg</i>	-
SSL_X509VerifyPolicy <i>arg</i>	-
SSL_LogX509Attributes <i>arg</i>	-
<b>Stronghold 2.x compatibility:</b>	
StrongholdAccelerator <i>dir</i>	-

StrongholdKey <i>dir</i>	-
StrongholdLicenseFile <i>dir</i>	-
SSLFlag <i>flag</i>	SSLEngine <i>flag</i>
SSLSessionLockFile <i>file</i>	SSLMutex <i>file</i>
SSLCipherList <i>spec</i>	SSLCipherSuite <i>spec</i>
RequireSSL	SSLRequireSSL
SSLErrorFile <i>file</i>	-
SSLRoot <i>dir</i>	-
SSL_CertificateLogDir <i>dir</i>	-
AuthCertDir <i>dir</i>	-
SSL_Group <i>name</i>	-
SSLProxyMachineCertPath <i>dir</i>	-
SSLProxyMachineCertFile <i>file</i>	-
SSLProxyCACertificatePath <i>dir</i>	-
SSLProxyCACertificateFile <i>file</i>	-
SSLProxyVerifyDepth <i>number</i>	-
SSLProxyCipherList <i>spec</i>	-



When you use ``SSLOptions +CompatEnvVars" additional environment variables are generated. They all correspond to existing official mod\_ssl variables. The currently implemented variable derivation is listed in [Table 2](#).

**Table 2: Environment Variable Derivation**

Old Variable	mod_ssl Variable
SSL_PROTOCOL_VERSION	SSL_PROTOCOL
SSLEAY_VERSION	SSL_VERSION_LIBRAR
HTTPS_SECRETKEYSIZE	SSL_CIPHER_USEKEYS
HTTPS_KEYSIZE	SSL_CIPHER_ALGKEYS
HTTPS_CIPHER	SSL_CIPHER
HTTPS_EXPORT	SSL_CIPHER_EXPORT
SSL_SERVER_KEY_SIZE	SSL_CIPHER_ALGKEYS
SSL_SERVER_CERTIFICATE	SSL_SERVER_CERT
SSL_SERVER_CERT_START	SSL_SERVER_V_START
SSL_SERVER_CERT_END	SSL_SERVER_V_END
SSL_SERVER_CERT_SERIAL	SSL_SERVER_M_SERIA
SSL_SERVER_SIGNATURE_ALGORITHM	SSL_SERVER_A_SIG
SSL_SERVER_DN	SSL_SERVER_S_DN
SSL_SERVER_CN	SSL_SERVER_S_DN_CN
SSL_SERVER_EMAIL	SSL_SERVER_S_DN_Em
SSL_SERVER_O	SSL_SERVER_S_DN_O
SSL_SERVER_OU	SSL_SERVER_S_DN_OU
SSL_SERVER_C	SSL_SERVER_S_DN_C
SSL_SERVER_SP	SSL_SERVER_S_DN_SP
SSL_SERVER_L	SSL_SERVER_S_DN_L
SSL_SERVER_IDN	SSL_SERVER_I_DN
SSL_SERVER_ICN	SSL_SERVER_I_DN_CN

SSL_SERVER_IEMAIL	SSL_SERVER_I_DN_Em
SSL_SERVER_IO	SSL_SERVER_I_DN_O
SSL_SERVER_IOU	SSL_SERVER_I_DN_OU
SSL_SERVER_IC	SSL_SERVER_I_DN_C
SSL_SERVER_ISP	SSL_SERVER_I_DN_SP
SSL_SERVER_IL	SSL_SERVER_I_DN_L
SSL_CLIENT_CERTIFICATE	SSL_CLIENT_CERT
SSL_CLIENT_CERT_START	SSL_CLIENT_V_START
SSL_CLIENT_CERT_END	SSL_CLIENT_V_END
SSL_CLIENT_CERT_SERIAL	SSL_CLIENT_M_SERIA
SSL_CLIENT_SIGNATURE_ALGORITHM	SSL_CLIENT_A_SIG
SSL_CLIENT_DN	SSL_CLIENT_S_DN
SSL_CLIENT_CN	SSL_CLIENT_S_DN_CN
SSL_CLIENT_EMAIL	SSL_CLIENT_S_DN_Em
SSL_CLIENT_O	SSL_CLIENT_S_DN_O
SSL_CLIENT_OU	SSL_CLIENT_S_DN_OU
SSL_CLIENT_C	SSL_CLIENT_S_DN_C
SSL_CLIENT_SP	SSL_CLIENT_S_DN_SP
SSL_CLIENT_L	SSL_CLIENT_S_DN_L
SSL_CLIENT_IDN	SSL_CLIENT_I_DN
SSL_CLIENT_ICN	SSL_CLIENT_I_DN_CN
SSL_CLIENT_IEMAIL	SSL_CLIENT_I_DN_Em
SSL_CLIENT_IO	SSL_CLIENT_I_DN_O
SSL_CLIENT_IOU	SSL_CLIENT_I_DN_OU
SSL_CLIENT_IC	SSL_CLIENT_I_DN_C
SSL_CLIENT_ISP	SSL_CLIENT_I_DN_SP
SSL_CLIENT_IL	SSL_CLIENT_I_DN_L
SSL_EXPORT	SSL_CIPHER_EXPORT
SSL_KEYSIZE	SSL_CIPHER_ALGKEYS
SSL_SECKEYSIZE	SSL_CIPHER_USEKEYS

SSL_SSLEAY_VERSION	SSL_VERSION_LIBRAR
SSL_STRONG_CRYPTO	-
SSL_SERVER_KEY_EXP	-
SSL_SERVER_KEY_ALGORITHM	-
SSL_SERVER_KEY_SIZE	-
SSL_SERVER_SESSIONDIR	-
SSL_SERVER_CERTIFICATELOGDIR	-
SSL_SERVER_CERTFILE	-
SSL_SERVER_KEYFILE	-

SSL_SERVER_KEYFILETYPE	-
SSL_CLIENT_KEY_EXP	-
SSL_CLIENT_KEY_ALGORITHM	-
SSL_CLIENT_KEY_SIZE	-



## Custom Log Functions

When `mod_ssl` is built into Apache or at least loaded (under DSO situation) additional functions exist for the [Custom Log Format](#) of [mod\\_log\\_config](#) as documented in the Reference Chapter.

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

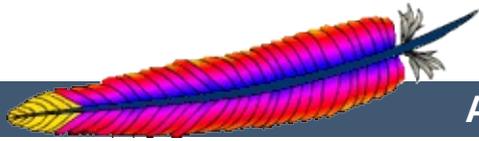
**Table 3: Custom Log Cryptography Function**

Function Call	Description
<code>%...{version}c</code>	SSL protocol version
<code>%...{cipher}c</code>	SSL cipher
<code>%... {subjectdn}c</code>	Client Certificate Subject Distinguished Name
<code>%...{issuerdn}c</code>	Client Certificate Issuer Distinguished Name
<code>%...{errcode}c</code>	Certificate Verification Error (numerical)
<code>%...{errstr}c</code>	Certificate Verification Error (string)

---

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

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



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



## Apache HTTP Server Version 2.0

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

## SSL/TLS Strong Encryption: How-To

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

-- Standard textbook cookie

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



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

## How can I create a real SSLv2-only server?

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

### httpd.conf

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

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

The following enables only the strongest ciphers:

### httpd.conf

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

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

This facility is called Server Gated Cryptography (SGC) and details you can find in the README.GlobalID document in the mod\_ssl distribution. In short: The server has a Global ID server certificate, signed by a special CA certificate from Verisign which enables strong encryption in export browsers. This works as following: The browser connects with an export cipher, the server sends its Global ID certificate, the browser verifies it and

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

### httpd.conf

```
# allow all ciphers for the initial handshake,
# so export browsers can upgrade via SGC facility
SSLCipherSuite
ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

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

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

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

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

<Location /strong/area>
# but https://hostname/strong/area/ and below
# requires strong ciphers
SSLCipherSuite HIGH:!aNULL:!MD5
</Location>
```



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

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

When you know your user community (i.e. a closed user group situation), as it's the case for instance in an Intranet, you can use plain certificate authentication. All you have to do is to create client certificates signed by your own CA certificate `ca.crt` and then verify the clients against this certificate.

### httpd.conf

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

## How can I authenticate my clients for a particular URL based on certificates but still allow arbitrary clients to access the remaining parts of the server?

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

### httpd.conf

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

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

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

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

The first method:

### httpd.conf

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

SSLVerifyClient      require
SSLVerifyDepth       5
SSLCACertificateFile conf/ssl.crt/ca.crt
SSLCACertificatePath conf/ssl.crt
SSLOptions            +FakeBasicAuth
SSLRequireSSL
AuthName              "Snake Oil Authentication"
AuthType              Basic
AuthUserFile          /usr/local/apache2/conf/httpd.passwd
require               valid-user
</Directory>
```

The password used in this example is the DES encrypted string "password". See the [SSLOptions](#) docs for more information.

### httpd.passwd

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

The second method:

### httpd.conf

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

    SSLVerifyClient      require
    SSLVerifyDepth      5
    SSLCACertificateFile conf/ssl.crt/ca.crt
    SSLCACertificatePath conf/ssl.crt
    SSLOptions          +FakeBasicAuth
    SSLRequireSSL
    SSLRequire          %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
                        and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"
</Directory>
```

## How can I require HTTPS with strong ciphers and either basic authentication or client certificates for access to a subarea on the Intranet website for clients coming from the Internet but still allow plain HTTP access for clients on the Intranet?

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

### httpd.conf

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

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

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

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

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

#   Allow Network Access and/or Basic Auth
Satisfy              any

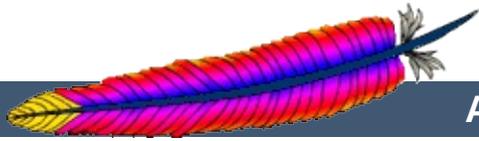
#   Network Access Control
Order                deny,allow
Deny                 from all
Allow                192.168.1.0/24

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

---

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

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



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



## Apache HTTP Server Version 2.0

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

## SSL/TLS Strong Encryption: FAQ

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

-- Claude Levi-Strauss

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

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



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

## What is the history of mod\_ssl?

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

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

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

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

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

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

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

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



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

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

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

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

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

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

SSLRandomSeed directive.



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

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

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

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

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

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

While you usually just use

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

## **What are RSA Private Keys, CSRs and Certificates?**

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

A Certificate Signing Request (CSR) is a digital file which contains your public key and your name. You send the CSR to a Certifying Authority (CA), who will convert it into a real Certificate, by signing it.

A Certificate contains your RSA public key, your name, the name of the CA, and is digitally signed by the CA. Browsers that know the CA can verify the signature on that Certificate, thereby obtaining your RSA public key. That enables them to send messages which only you can decrypt.

See the [Introduction](#) chapter for a general description of the SSL protocol.

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

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

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

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

1. Make sure OpenSSL is installed and in your PATH.
2. Run the following command, to create `server.key` and `server.crt` files:

```
$ openssl req -new -x509 -nodes -out  
server.crt -keyout server.key
```

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

```
SSLCertificateFile    /path/to/th  
SSLCertificateKeyFile /path/to/th
```

3. It is important that you are aware that this `server.key` does *not* have any passphrase. To add a passphrase to the key, you should run the following command, and enter & verify the passphrase as requested.

```
$ openssl rsa -des3 -in server.key -out  
server.key.new  
$ mv server.key.new server.key
```

Please backup the `server.key` file, and the passphrase you entered, in a secure location.

## How do I create a real SSL Certificate?

Here is a step-by-step description:

1. Make sure OpenSSL is installed and in your PATH.
2. Create a RSA private key for your Apache server (will be Triple-DES encrypted and PEM formatted):

```
$ openssl genrsa -des3 -out server.key 1024
```

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

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

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

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

3. Create a Certificate Signing Request (CSR) with the server RSA private key (output will be PEM formatted):

```
$ openssl req -new -key server.key -out  
server.csr
```

Make sure you enter the FQDN ("Fully Qualified Domain Name") of the server when OpenSSL prompts you for the "CommonName", i.e. when you generate a CSR for a website which will be later accessed via `https://www.foo.dom/`, enter "www.foo.dom" here. You can see the details of this CSR by using

```
$ openssl req -noout -text -in server.csr
```

4. You now have to send this Certificate Signing Request (CSR) to a Certifying Authority (CA) to be signed. Once the CSR has been signed, you will have a real Certificate, which can be used by Apache. You can have a CSR signed by a commercial CA, or you can create your own CA to sign it. Commercial CAs usually ask you to post the CSR into a web form, pay for the signing, and then send a signed Certificate, which you can store in a `server.crt` file. For more information about commercial CAs see the following locations:

1. Verisign

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

2. Thawte

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

3. CertiSign Certificadora Digital Ltda.

<http://www.certisign.com.br>

4. IKS GmbH

<http://www.iks-jena.de/leistungen/ca/>

5. Uptime Commerce Ltd.

<http://www.uptimecommerce.com>

6. BelSign NV/SA

<http://www.belsign.be>

For details on how to create your own CA, and use this to sign a CSR, see [below](#).

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

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

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

```
SSLCertificateFile    /path/to/this/ser  
SSLCertificateKeyFile /path/to/this/ser
```

The `server.csr` file is no longer needed.

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

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

1. Create a RSA private key for your server (will be Triple-DES encrypted and PEM formatted):

```
$ openssl genrsa -des3 -out server.key 1024
```

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

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

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

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

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

```
$ openssl req -new -x509 -nodes -sha1 -days  
365 -key server.key -out server.crt
```

This signs the server CSR and results in a server . crt file. You can see the details of this Certificate using:

```
$ openssl x509 -noout -text -in server.crt
```

## **How can I change the pass-phrase on my private key file?**

You simply have to read it with the old pass-phrase and write it again, specifying the new pass-phrase. You can accomplish this

with the following commands:

```
$ openssl rsa -des3 -in server.key -out  
server.key.new  
$ mv server.key.new server.key
```

The first time you're asked for a PEM pass-phrase, you should enter the old pass-phrase. After that, you'll be asked again to enter a pass-phrase - this time, use the new pass-phrase. If you are asked to verify the pass-phrase, you'll need to enter the new pass-phrase a second time.

## **How can I get rid of the pass-phrase dialog at Apache startup time?**

The reason this dialog pops up at startup and every re-start is that the RSA private key inside your server.key file is stored in encrypted format for security reasons. The pass-phrase is needed to decrypt this file, so it can be read and parsed. Removing the pass-phrase removes a layer of security from your server - proceed with caution!

1. Remove the encryption from the RSA private key (while keeping a backup copy of the original file):

```
$ cp server.key server.key.org  
$ openssl rsa -in server.key.org -out  
server.key
```

2. Make sure the server.key file is only readable by root:

```
$ chmod 400 server.key
```

Now server . key contains an unencrypted copy of the key. If you

point your server at this file, it will not prompt you for a pass-phrase. HOWEVER, if anyone gets this key they will be able to impersonate you on the net. PLEASE make sure that the permissions on this file are such that only root or the web server user can read it (preferably get your web server to start as root but run as another user, and have the key readable only by root).

As an alternative approach you can use the ```SSLPassPhraseDialog exec:/path/to/program``` facility. Bear in mind that this is neither more nor less secure, of course.

## **How do I verify that a private key matches its Certificate?**

A private key contains a series of numbers. Two of these numbers form the "public key", the others are part of the "private key". The "public key" bits are included when you generate a CSR, and subsequently form part of the associated Certificate.

To check that the public key in your Certificate matches the public portion of your private key, you simply need to compare these numbers. To view the Certificate and the key run the commands:

```
$ openssl x509 -noout -text -in server.crt  
$ openssl rsa -noout -text -in server.key
```

The ``modulus`` and the ``public exponent`` portions in the key and the Certificate must match. As the public exponent is usually 65537 and it's difficult to visually check that the long modulus numbers are the same, you can use the following approach:

```
$ openssl x509 -noout -modulus -in server.crt |  
openssl md5  
$ openssl rsa -noout -modulus -in server.key |  
openssl md5
```

This leaves you with two rather shorter numbers to compare. It is, in theory, possible that these numbers may be the same, without the modulus numbers being the same, but the chances of this are overwhelmingly remote.

Should you wish to check to which key or certificate a particular CSR belongs you can perform the same calculation on the CSR as follows:

```
$ openssl req -noout -modulus -in server.csr |  
openssl md5
```

### **Why do connections fail with an "alert bad certificate" error?**

Errors such as `OpenSSL: error:14094412: SSL routines:SSL3_READ_BYTES:sslv3 alert bad certificate` in the SSL logfile, are usually caused by a browser which is unable to handle the server certificate/private-key. For example, Netscape Navigator 3.x is unable to handle RSA key lengths not equal to 1024 bits.

### **Why does my 2048-bit private key not work?**

The private key sizes for SSL must be either 512 or 1024 bits, for compatibility with certain web browsers. A keysize of 1024 bits is recommended because keys larger than 1024 bits are incompatible with some versions of Netscape Navigator and Microsoft Internet Explorer, and with other browsers that use RSA's BSAFE cryptography toolkit.

### **Why is client authentication broken after upgrading from SSLeay version 0.8 to 0.9?**

The CA certificates under the path you configured with `SSLCACertificatePath` are found by SSLeay through hash

symlinks. These hash values are generated by the ``openssl x509 -noout -hash'` command. However, the algorithm used to calculate the hash for a certificate changed between SSLeay 0.8 and 0.9. You will need to remove all old hash symlinks and create new ones after upgrading. Use the `Makefile` provided by [mod\\_ssl](#).

## How can I convert a certificate from PEM to DER format?

The default certificate format for SSLeay/OpenSSL is PEM, which is simply Base64 encoded DER, with header and footer lines. For some applications (e.g. Microsoft Internet Explorer) you need the certificate in plain DER format. You can convert a PEM file `cert.pem` into the corresponding DER file `cert.der` using the following command: `$ openssl x509 -in cert.pem -out cert.der -outform DER`

## Why can't I find the `getca` or `getverisign` programs mentioned by Verisign, for installing my Verisign certificate?

Verisign has never provided specific instructions for Apache+`mod_ssl`. The instructions provided are for C2Net's Stronghold (a commercial Apache based server with SSL support).

To install your certificate, all you need to do is to save the certificate to a file, and give the name of that file to the [SSLCertificateFile](#) directive. You will also need to give it the key file. For more information, see the [SSLCertificateKeyFile](#) directive.

## Can I use the Server Gated Cryptography (SGC)

## facility (aka Verisign Global ID) with mod\_ssl?

Yes. [mod\\_ssl](#) has included support for the SGC facility since version 2.1. No special configuration is required - just use the Global ID as your server certificate. The *step up* of the clients is then automatically handled by [mod\\_ssl](#) at run-time.

## Why do browsers complain that they cannot verify my Verisign Global ID server certificate?

Verisign uses an intermediate CA certificate between the root CA certificate (which is installed in the browsers) and the server certificate (which you installed on the server). You should have received this additional CA certificate from Verisign. If not, complain to them. Then, configure this certificate with the [SSLCertificateChainFile](#) directive. This ensures that the intermediate CA certificate is sent to the browser, filling the gap in the certificate chain.



- [Why do I get lots of random SSL protocol errors under heavy server load?](#)
- [Why does my webserver have a higher load, now that it serves SSL encrypted traffic?](#)
- [Why do HTTPS connections to my server sometimes take up to 30 seconds to establish a connection?](#)
- [What SSL Ciphers are supported by mod\\_ssl?](#)
- [Why do I get "no shared cipher" errors, when trying to use Anonymous Diffie-Hellman \(ADH\) ciphers?](#)
- [Why do I get a 'no shared ciphers' error when connecting to my newly installed server?](#)
- [Why can't I use SSL with name-based/non-IP-based virtual hosts?](#)
- [Why is it not possible to use Name-Based Virtual Hosting to identify different SSL virtual hosts?](#)
- [How do I get SSL compression working?](#)
- [When I use Basic Authentication over HTTPS the lock icon in Netscape browsers stays unlocked when the dialog pops up. Does this mean the username/password is being sent unencrypted?](#)
- [Why do I get I/O errors when connecting via HTTPS to an Apache+mod\\_ssl server with Microsoft Internet Explorer \(MSIE\)?](#)
- [Why do I get I/O errors, or the message "Netscape has encountered bad data from the server", when connecting via HTTPS to an Apache+mod\\_ssl server with Netscape Navigator?](#)

## **Why do I get lots of random SSL protocol errors under heavy server load?**

There can be a number of reasons for this, but the main one is problems with the SSL session Cache specified by the

[SSLSessionCache](#) directive. The DBM session cache is the most likely source of the problem, so using the SHM session cache (or no cache at all) may help.

## **Why does my webserver have a higher load, now that it serves SSL encrypted traffic?**

SSL uses strong cryptographic encryption, which necessitates a lot of number crunching. When you request a webpage via HTTPS, everything (even the images) is encrypted before it is transferred. So increased HTTPS traffic leads to load increases.

## **Why do HTTPS connections to my server sometimes take up to 30 seconds to establish a connection?**

This is usually caused by a `/dev/random` device for [SSLRandomSeed](#) which blocks the `read(2)` call until enough entropy is available to service the request. More information is available in the reference manual for the [SSLRandomSeed](#) directive.

## **What SSL Ciphers are supported by mod\_ssl?**

Usually, any SSL ciphers supported by the version of OpenSSL in use, are also supported by `mod_ssl`. Which ciphers are available can depend on the way you built OpenSSL. Typically, at least the following ciphers are supported:

1. RC4 with MD5
2. RC4 with MD5 (export version restricted to 40-bit key)
3. RC2 with MD5
4. RC2 with MD5 (export version restricted to 40-bit key)
5. IDEA with MD5

6. DES with MD5
7. Triple-DES with MD5

To determine the actual list of ciphers available, you should run the following:

```
$ openssl ciphers -v
```

### **Why do I get "no shared cipher" errors, when trying to use Anonymous Diffie-Hellman (ADH) ciphers?**

By default, OpenSSL does *not* allow ADH ciphers, for security reasons. Please be sure you are aware of the potential side-effects if you choose to enable these ciphers.

In order to use Anonymous Diffie-Hellman (ADH) ciphers, you must build OpenSSL with ```-DSSL_ALLOW_ADH```, and then add ```ADH``` into your [SSLCipherSuite](#).

### **Why do I get a 'no shared ciphers' error when connecting to my newly installed server?**

Either you have made a mistake with your [SSLCipherSuite](#) directive (compare it with the pre-configured example in `httpd.conf-dist`) or you chose to use DSA/DH algorithms instead of RSA when you generated your private key and ignored or overlooked the warnings. If you have chosen DSA/DH, then your server cannot communicate using RSA-based SSL ciphers (at least until you configure an additional RSA-based certificate/key pair). Modern browsers like NS or IE can only communicate over SSL using RSA ciphers. The result is the "no shared ciphers" error. To fix this, regenerate your server certificate/key pair, using the RSA algorithm.

## **Why can't I use SSL with name-based/non-IP-based virtual hosts?**

The reason is very technical, and a somewhat "chicken and egg" problem. The SSL protocol layer stays below the HTTP protocol layer and encapsulates HTTP. When an SSL connection (HTTPS) is established Apache/mod\_ssl has to negotiate the SSL protocol parameters with the client. For this, mod\_ssl has to consult the configuration of the virtual server (for instance it has to look for the cipher suite, the server certificate, etc.). But in order to go to the correct virtual server Apache has to know the Host HTTP header field. To do this, the HTTP request header has to be read. This cannot be done before the SSL handshake is finished, but the information is needed in order to complete the SSL handshake phase. Bingo!

## **Why is it not possible to use Name-Based Virtual Hosting to identify different SSL virtual hosts?**

Name-Based Virtual Hosting is a very popular method of identifying different virtual hosts. It allows you to use the same IP address and the same port number for many different sites. When people move on to SSL, it seems natural to assume that the same method can be used to have lots of different SSL virtual hosts on the same server.

It comes as rather a shock to learn that it is impossible.

The reason is that the SSL protocol is a separate layer which encapsulates the HTTP protocol. So the SSL session is a separate transaction, that takes place before the HTTP session has begun. The server receives an SSL request on IP address X and port Y (usually 443). Since the SSL request does not contain any Host: field, the server has no way to decide which SSL virtual host to use. Usually, it will just use the first one it finds, which matches the port and IP address specified.

You can, of course, use Name-Based Virtual Hosting to identify many non-SSL virtual hosts (all on port 80, for example) and then have a single SSL virtual host (on port 443). But if you do this, you must make sure to put the non-SSL port number on the NameVirtualHost directive, e.g.

```
NameVirtualHost 192.168.1.1:80
```

Other workaround solutions include:

Using separate IP addresses for different SSL hosts. Using different port numbers for different SSL hosts.

## How do I get SSL compression working?

Although SSL compression negotiation was defined in the specification of SSLv2 and TLS, it took until May 2004 for RFC 3749 to define DEFLATE as a negotiable standard compression method.

OpenSSL 0.9.8 started to support this by default when compiled with the `zlib` option. If both the client and the server support compression, it will be used. However, most clients still try to initially connect with an SSLv2 Hello. As SSLv2 did not include an array of preferred compression algorithms in its handshake, compression cannot be negotiated with these clients. If the client disables support for SSLv2, either an SSLv3 or TLS Hello may be sent, depending on which SSL library is used, and compression may be set up. You can verify whether clients make use of SSL compression by logging the  `%{SSL_COMPRESS_METHOD}x` variable.

**When I use Basic Authentication over HTTPS the lock icon in Netscape browsers stays unlocked when the dialog pops up. Does this mean the**

## username/password is being sent unencrypted?

No, the username/password is transmitted encrypted. The icon in Netscape browsers is not actually synchronized with the SSL/TLS layer. It only toggles to the locked state when the first part of the actual webpage data is transferred, which may confuse people. The Basic Authentication facility is part of the HTTP layer, which is above the SSL/TLS layer in HTTPS. Before any HTTP data communication takes place in HTTPS, the SSL/TLS layer has already completed its handshake phase, and switched to encrypted communication. So don't be confused by this icon.

## Why do I get I/O errors when connecting via HTTPS to an Apache+mod\_ssl server with Microsoft Internet Explorer (MSIE)?

The first reason is that the SSL implementation in some MSIE versions has some subtle bugs related to the HTTP keep-alive facility and the SSL close notify alerts on socket connection close. Additionally the interaction between SSL and HTTP/1.1 features are problematic in some MSIE versions. You can work around these problems by forcing Apache not to use HTTP/1.1, keep-alive connections or send the SSL close notify messages to MSIE clients. This can be done by using the following directive in your SSL-aware virtual host section:

```
SetEnvIf User-Agent ".*MSIE.*" \
nokeepalive ssl-unclean-shutdown \
downgrade-1.0 force-response-1.0
```

Further, some MSIE versions have problems with particular ciphers. Unfortunately, it is not possible to implement a MSIE-specific workaround for this, because the ciphers are needed as early as the SSL handshake phase. So a MSIE-specific [SetEnvIf](#) won't solve these problems. Instead, you will have to make more drastic adjustments to the global parameters. Before

you decide to do this, make sure your clients really have problems. If not, do not make these changes - they will affect *all* your clients, MSIE or otherwise.

The next problem is that 56bit export versions of MSIE 5.x browsers have a broken SSLv3 implementation, which interacts badly with OpenSSL versions greater than 0.9.4. You can accept this and require your clients to upgrade their browsers, you can downgrade to OpenSSL 0.9.4 (not advised), or you can work around this, accepting that your workaround will affect other browsers too:

```
SSLProtocol all -SSLv3
```

will completely disables the SSLv3 protocol and allow those browsers to work. A better workaround is to disable only those ciphers which cause trouble.

```
SSLCipherSuite  
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLV2:+EXP
```

This also allows the broken MSIE versions to work, but only removes the newer 56bit TLS ciphers.

Another problem with MSIE 5.x clients is that they refuse to connect to URLs of the form `https://12.34.56.78/` (where IP-addresses are used instead of the hostname), if the server is using the Server Gated Cryptography (SGC) facility. This can only be avoided by using the fully qualified domain name (FQDN) of the website in hyperlinks instead, because MSIE 5.x has an error in the way it handles the SGC negotiation.

And finally there are versions of MSIE which seem to require that an SSL session can be reused (a totally non standard-conforming behaviour, of course). Connecting with those MSIE versions only

work if a SSL session cache is used. So, as a work-around, make sure you are using a session cache (see the [SSLSessionCache](#) directive).

### **Why do I get I/O errors, or the message "Netscape has encountered bad data from the server", when connecting via HTTPS to an Apache+mod\_ssl server with Netscape Navigator?**

This usually occurs when you have created a new server certificate for a given domain, but had previously told your browser to always accept the old server certificate. Once you clear the entry for the old certificate from your browser, everything should be fine. Netscape's SSL implementation is correct, so when you encounter I/O errors with Netscape Navigator it is usually caused by the configured certificates.



- [What information resources are available in case of mod\\_ssl problems?](#)
- [What support contacts are available in case of mod\\_ssl problems?](#)
- [What information should I provide when writing a bug report?](#)
- [I had a core dump, can you help me?](#)
- [How do I get a backtrace, to help find the reason for my core dump?](#)

## What information resources are available in case of mod\_ssl problems?

The following information resources are available. In case of problems you should search here first.

### Answers in the User Manual's F.A.Q. List (this)

[http://httpd.apache.org/docs/2.0/ssl/ssl\\_faq.html](http://httpd.apache.org/docs/2.0/ssl/ssl_faq.html)

First check the F.A.Q. (this text). If your problem is a common one, it may have been answered several times before, and been included in this doc.

### Postings from the modssl-users Support Mailing List

<http://www.modssl.org/support/>

Search for your problem in the archives of the modssl-users mailing list. You're probably not the first person to have had this problem!

## What support contacts are available in case of mod\_ssl problems?

The following lists all support possibilities for mod\_ssl, in order of preference. Please go through these possibilities *in this order* - don't just pick the one you like the look of.

1. *Send a Problem Report to the modssl-users Support Mailing*

### *List*

[modssl-users@modssl.org](mailto:modssl-users@modssl.org)

This is the preferred way of submitting your problem report, because this way, others can see the problem, and learn from any answers. You must subscribe to the list first, but you can then easily discuss your problem with both the author and the whole mod\_ssl user community.

### 2. *Send a Problem Report to the Apache httpd Users Support Mailing List*

[users@httpd.apache.org](mailto:users@httpd.apache.org)

This is the second way of submitting your problem report. Again, you must subscribe to the list first, but you can then easily discuss your problem with the whole Apache httpd user community.

### 3. *Write a Problem Report in the Bug Database*

[http://httpd.apache.org/bug\\_report.html](http://httpd.apache.org/bug_report.html)

This is the last way of submitting your problem report. You should only do this if you've already posted to the mailing lists, and had no success. Please follow the instructions on the above page *carefully*.

## **What information should I provide when writing a bug report?**

You should always provide at least the following information:

### **Apache and OpenSSL version information**

The Apache version can be determined by running `httpd -v`. The OpenSSL version can be determined by running `openssl version`. Alternatively, if you have Lynx installed, you can run the command `lynx -mime_header http://localhost/ | grep Server` to gather this information in a single step.

## **The details on how you built and installed Apache+mod\_ssl+OpenSSL**

For this you can provide a logfile of your terminal session which shows the configuration and install steps. If this is not possible, you should at least provide the [configure](#) command line you used.

## **In case of core dumps please include a Backtrace**

If your Apache+mod\_ssl+OpenSSL dumps its core, please attach a stack-frame ``backtrace" (see [below](#) for information on how to get this). This information is required in order to find a reason for your core dump.

## **A detailed description of your problem**

Don't laugh, we really mean it! Many problem reports don't include a description of what the actual problem is. Without this, it's very difficult for anyone to help you. So, it's in your own interest (you want the problem be solved, don't you?) to include as much detail as possible, please. Of course, you should still include all the essentials above too.

## **I had a core dump, can you help me?**

In general no, at least not unless you provide more details about the code location where Apache dumped core. What is usually always required in order to help you is a backtrace (see next question). Without this information it is mostly impossible to find the problem and help you in fixing it.

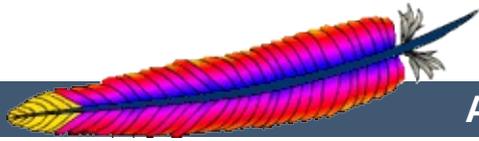
## **How do I get a backtrace, to help find the reason for my core dump?**

Following are the steps you will need to complete, to get a backtrace:

1. Make sure you have debugging symbols available, at least in

Apache. On platforms where you use GCC/GDB, you will have to build Apache+mod\_ssl with ```OPTIM="-g -ggdb3""` to get this. On other platforms at least ```OPTIM="-g""` is needed.

2. Start the server and try to reproduce the core-dump. For this you may want to use a directive like ```CoreDumpDirectory /tmp"` to make sure that the core-dump file can be written. This should result in a `/tmp/core` or `/tmp/httpd.core` file. If you don't get one of these, try running your server under a non-root UID. Many modern kernels do not allow a process to dump core after it has done a `setuid( )` (unless it does an `exec( )`) for security reasons (there can be privileged information left over in memory). If necessary, you can run `/path/to/httpd -X` manually to force Apache to not fork.
3. Analyze the core-dump. For this, run `gdb /path/to/httpd /tmp/httpd.core` or a similar command. In GDB, all you have to do then is to enter `bt`, and voila, you get the backtrace. For other debuggers consult your local debugger manual.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

# Authentication, Authorization and Access Control

Authentication is any process by which you verify that someone is who they claim they are. Authorization is any process by which someone is allowed to be where they want to go, or to have information that they want to have.



## Related Modules and Directives

Related Modules	Related Directives
<a href="#">mod_auth</a>	<a href="#">Allow</a>
<a href="#">mod_access</a>	<a href="#">AuthGroupFile</a>
	<a href="#">AuthName</a>
	<a href="#">AuthType</a>
	<a href="#">AuthUserFile</a>
	<a href="#">Deny</a>
	<a href="#">Options</a>
	<a href="#">Require</a>



---

If you have information on your web site that is sensitive or intended for only a small group of people, the techniques in this article will help you make sure that the people that see those pages are the people that you wanted to see them.

This article covers the "standard" way of protecting parts of your web site that most of you are going to use.



## Prerequisites

The directives discussed in this article will need to go either in your main server configuration file (typically in a [<Directory>](#) section), or in per-directory configuration files (.htaccess files).

If you plan to use .htaccess files, you will need to have a server configuration that permits putting authentication directives in these files. This is done with the [AllowOverride](#) directive, which specifies which directives, if any, may be put in per-directory configuration files.

Since we're talking here about authentication, you will need an [AllowOverride](#) directive like the following:

```
AllowOverride AuthConfig
```

Or, if you are just going to put the directives directly in your main server configuration file, you will of course need to have write permission to that file.

And you'll need to know a little bit about the directory structure of your server, in order to know where some files are kept. This should not be terribly difficult, and I'll try to make this clear when we come to that point.



Here's the basics of password protecting a directory on your server.

You'll need to create a password file. This file should be placed somewhere not accessible from the web. This is so that folks cannot download the password file. For example, if your documents are served out of `/usr/local/apache/htdocs` you might want to put the password file(s) in `/usr/local/apache/passwd`.

To create the file, use the `htpasswd` utility that came with Apache. This will be located in the `bin` directory of wherever you installed Apache. To create the file, type:

```
htpasswd -c /usr/local/apache/passwd/passwords rbowen
```

`htpasswd` will ask you for the password, and then ask you to type it again to confirm it:

```
# htpasswd -c /usr/local/apache/passwd/passwords rbowen
New password: mypassword
Re-type new password: mypassword
Adding password for user rbowen
```

If `htpasswd` is not in your path, of course you'll have to type the full path to the file to get it to run. On my server, it's located at `/usr/local/apache/bin/htpasswd`

Next, you'll need to configure the server to request a password and tell the server which users are allowed access. You can do this either by editing the `httpd.conf` file or using an `.htaccess` file. For example, if you wish to protect the directory `/usr/local/apache/htdocs/secret`, you can use the following directives, either placed in the file

`/usr/local/apache/htdocs/secret/.htaccess`, or placed in `httpd.conf` inside a `<Directory /usr/local/apache/apache/htdocs/secret>` section.

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /usr/local/apache/passwd/passwords
Require user rbowen
```

Let's examine each of those directives individually. The [AuthType](#) directive selects that method that is used to authenticate the user. The most common method is `Basic`, and this is the method implemented by [mod\\_auth](#). It is important to be aware, however, that `Basic` authentication sends the password from the client to the server unencrypted. This method should therefore not be used for highly sensitive data. Apache supports one other authentication method: `AuthType Digest`. This method is implemented by [mod\\_auth\\_digest](#) and is much more secure. Only the most recent versions of clients are known to support `Digest` authentication.

The [AuthName](#) directive sets the *Realm* to be used in the authentication. The realm serves two major functions. First, the client often presents this information to the user as part of the password dialog box. Second, it is used by the client to determine what password to send for a given authenticated area.

So, for example, once a client has authenticated in the "Restricted Files" area, it will automatically retry the same password for any area on the same server that is marked with the "Restricted Files" Realm. Therefore, you can prevent a user from being prompted more than once for a password by letting multiple restricted areas share the same realm. Of course, for security reasons, the client will always need to ask again for the password whenever the hostname of the server changes.

The [AuthUserFile](#) directive sets the path to the password file that we just created with [htpasswd](#). If you have a large number of users, it can be quite slow to search through a plain text file to authenticate the user on each request. Apache also has the ability to store user information in fast database files. The [mod\\_auth\\_dbm](#) module provides the [AuthDBMUserFile](#) directive. These files can be created and manipulated with the [dbmmanage](#) program. Many other types of authentication options are available from third party modules in the [Apache Modules Database](#).

Finally, the [Require](#) directive provides the authorization part of the process by setting the user that is allowed to access this region of the server. In the next section, we discuss various ways to use the [Require](#) directive.



The directives above only let one person (specifically someone with a username of rbowen) into the directory. In most cases, you'll want to let more than one person in. This is where the [AuthGroupFile](#) comes in.

If you want to let more than one person in, you'll need to create a group file that associates group names with a list of users in that group. The format of this file is pretty simple, and you can create it with your favorite editor. The contents of the file will look like this:

```
GroupName: rbowen dpitts sungo rshersey
```

That's just a list of the members of the group in a long line separated by spaces.

To add a user to your already existing password file, type:

```
htpasswd /usr/local/apache/passwd/passwords dpitts
```

You'll get the same response as before, but it will be appended to the existing file, rather than creating a new file. (It's the `-c` that makes it create a new password file).

Now, you need to modify your `.htaccess` file to look like the following:

```
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /usr/local/apache/passwd/passwords
AuthGroupFile /usr/local/apache/passwd/groups
Require group GroupName
```

Now, anyone that is listed in the group `GroupName`, and has an entry in the password file, will be let in, if they type the correct password.

There's another way to let multiple users in that is less specific. Rather than creating a group file, you can just use the following directive:

```
Require valid-user
```

Using that rather than the `Require user rbowen` line will allow anyone in that is listed in the password file, and who correctly enters their password. You can even emulate the group behavior here, by just keeping a separate password file for each group. The advantage of this approach is that Apache only has to check one file, rather than two. The disadvantage is that you have to maintain a bunch of password files, and remember to reference the right one in the [AuthUserFile](#) directive.



## Performance Problems

Because of the way that Basic authentication is specified, your username and password must be verified every time you request a document from the server. This is even if you're reloading the same page, and for every image on the page (if they come from a protected directory). As you can imagine, this slows things down a little. The amount that it slows things down is proportional to the size of the password file, because it has to open up that file, and go down the list of users until it gets to your name. And it has to do this every time a page is loaded.

A consequence of this is that there's a practical limit to how many users you can put in one password file. This limit will vary depending on the performance of your particular server machine, but you can expect to see slowdowns once you get above a few hundred entries, and may wish to consider a different authentication method at that time.



Authentication by username and password is only part of the story. Frequently you want to let people in based on something other than who they are. Something such as where they are coming from.

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.

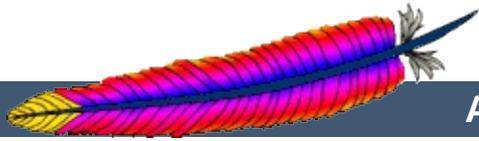


You should also read the documentation for [mod\\_auth](#) and [mod\\_access](#) which contain some more information about how this all works.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

# Apache Tutorial: Dynamic Content with CGI



## INTRODUCTION

Related Modules	Related Directives
<a href="#">mod_alias</a>	<a href="#">AddHandler</a>
<a href="#">mod_cgi</a>	<a href="#">Options</a>
	<a href="#">ScriptAlias</a>

The CGI (Common Gateway Interface) defines a way for a web server to interact with external content-generating programs, which are often referred to as CGI programs or CGI scripts. It is the simplest, and most common, way to put dynamic content on your web site. This document will be an introduction to setting up CGI on your Apache web server, and getting started writing CGI programs.



## Configuring Apache to permit CGI

In order to get your CGI programs to work properly, you'll need to have Apache configured to permit CGI execution. There are several ways to do this.

### ScriptAlias

The `ScriptAlias` directive tells Apache that a particular directory is set aside for CGI programs. Apache will assume that every file in this directory is a CGI program, and will attempt to execute it, when that particular resource is requested by a client.

The `ScriptAlias` directive looks like:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

The example shown is from your default `httpd.conf` configuration file, if you installed Apache in the default location. The `ScriptAlias` directive is much like the `Alias` directive, which defines a URL prefix that is to be mapped to a particular directory. `Alias` and `ScriptAlias` are usually used for directories that are outside of the `DocumentRoot` directory. The difference between `Alias` and `ScriptAlias` is that `ScriptAlias` has the added meaning that everything under that URL prefix will be considered a CGI program. So, the example above tells Apache that any request for a resource beginning with `/cgi-bin/` should be served from the directory `/usr/local/apache2/cgi-bin/`, and should be treated as a CGI program.

For example, if the URL `http://www.example.com/cgi-bin/test.pl` is requested, Apache will attempt to execute the file `/usr/local/apache2/cgi-bin/test.pl` and return the output. Of course, the file will have to exist, and be executable,

and return output in a particular way, or Apache will return an error message.

## CGI outside of ScriptAlias directories

CGI programs are often restricted to [ScriptAlias](#)'ed directories for security reasons. In this way, administrators can tightly control who is allowed to use CGI programs. However, if the proper security precautions are taken, there is no reason why CGI programs cannot be run from arbitrary directories. For example, you may wish to let users have web content in their home directories with the [UserDir](#) directive. If they want to have their own CGI programs, but don't have access to the main `cgi-bin` directory, they will need to be able to run CGI programs elsewhere.

There are two steps to allowing CGI execution in an arbitrary directory. First, the `cgi-script` handler must be activated using the [AddHandler](#) or [SetHandler](#) directive. Second, `ExecCGI` must be specified in the [Options](#) directive.

## Explicitly using Options to permit CGI execution

You could explicitly use the [Options](#) directive, inside your main server configuration file, to specify that CGI execution was permitted in a particular directory:

```
<Directory /usr/local/apache2/htdocs/somedir>
  Options +ExecCGI
</Directory>
```

The above directive tells Apache to permit the execution of CGI files. You will also need to tell the server what files are CGI files. The following [AddHandler](#) directive tells the server to treat all files with the `cgi` or `pl` extension as CGI programs:

```
AddHandler cgi-script .cgi .pl
```

## **.htaccess files**

The [.htaccess tutorial](#) shows how to activate CGI programs if you do not have access to `httpd.conf`.

## **User Directories**

To allow CGI program execution for any file ending in `.cgi` in users' directories, you can use the following configuration.

```
<Directory /home/*/public_html>  
  Options +ExecCGI  
  AddHandler cgi-script .cgi  
</Directory>
```

If you wish designate a `cgi-bin` subdirectory of a user's directory where everything will be treated as a CGI program, you can use the following.

```
<Directory /home/*/public_html/cgi-bin>  
  Options ExecCGI  
  SetHandler cgi-script  
</Directory>
```



## Writing a CGI program

There are two main differences between "regular" programming, and CGI programming.

First, all output from your CGI program must be preceded by a MIME-type header. This is HTTP header that tells the client what sort of content it is receiving. Most of the time, this will look like:

```
Content-type: text/html
```

Secondly, your output needs to be in HTML, or some other format that a browser will be able to display. Most of the time, this will be HTML, but occasionally you might write a CGI program that outputs a gif image, or other non-HTML content.

Apart from those two things, writing a CGI program will look a lot like any other program that you might write.

### Your first CGI program

The following is an example CGI program that prints one line to your browser. Type in the following, save it to a file called `first.pl`, and put it in your `cgi-bin` directory.

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello, World.";
```

Even if you are not familiar with Perl, you should be able to see what is happening here. The first line tells Apache (or whatever shell you happen to be running under) that this program can be executed by feeding the file to the interpreter found at the location `/usr/bin/perl`. The second line prints the content-type declaration we talked about, followed by two carriage-return newline pairs. This puts a blank line after the header, to indicate the end of the HTTP headers, and the beginning of the body. The

third line prints the string "Hello, World.". And that's the end of it.

If you open your favorite browser and tell it to get the address

```
http://www.example.com/cgi-bin/first.pl
```

or wherever you put your file, you will see the one line `He11o, Wor1d .` appear in your browser window. It's not very exciting, but once you get that working, you'll have a good chance of getting just about anything working.



---

There are four basic things that you may see in your browser when you try to access your CGI program from the web:

### **The output of your CGI program**

Great! That means everything worked fine. If the output is correct, but the browser is not processing it correctly, make sure you have the correct Content - Type set in your CGI program.

### **The source code of your CGI program or a "POST Method Not Allowed" message**

That means that you have not properly configured Apache to process your CGI program. Reread the section on [configuring Apache](#) and try to find what you missed.

### **A message starting with "Forbidden"**

That means that there is a permissions problem. Check the [Apache error log](#) and the section below on [file permissions](#).

### **A message saying "Internal Server Error"**

If you check the [Apache error log](#), you will probably find that it says "Premature end of script headers", possibly along with an error message generated by your CGI program. In this case, you will want to check each of the below sections to see what might be preventing your CGI program from emitting the proper HTTP headers.

## **File permissions**

Remember that the server does not run as you. That is, when the server starts up, it is running with the permissions of an unprivileged user - usually nobody, or www - and so it will need extra permissions to execute files that are owned by you. Usually, the way to give a file sufficient permissions to be executed by nobody is to give everyone execute permission on the file:

---

```
chmod a+x first.pl
```

Also, if your program reads from, or writes to, any other files, those files will need to have the correct permissions to permit this.

## Path information and environment

When you run a program from your command line, you have certain information that is passed to the shell without you thinking about it. For example, you have a PATH, which tells the shell where it can look for files that you reference.

When a program runs through the web server as a CGI program, it may not have the same PATH. Any programs that you invoke in your CGI program (like `sendmail`, for example) will need to be specified by a full path, so that the shell can find them when it attempts to execute your CGI program.

A common manifestation of this is the path to the script interpreter (often `perl`) indicated in the first line of your CGI program, which will look something like:

```
#!/usr/bin/perl
```

Make sure that this is in fact the path to the interpreter.

In addition, if your CGI program depends on other [environment variables](#), you will need to assure that those variables are passed by Apache.

## Program errors

Most of the time when a CGI program fails, it's because of a problem with the program itself. This is particularly true once you get the hang of this CGI stuff, and no longer make the above two mistakes. The first thing to do is to make sure that your program

runs from the command line before testing it via the web server.  
For example, try:

```
cd /usr/local/apache2/cgi-bin  
./first.pl
```

(Do not call the `perl` interpreter. The shell and Apache should find the interpreter using the [path information](#) on the first line of the script.)

The first thing you see written by your program should be a set of HTTP headers, including the Content - Type, followed by a blank line. If you see anything else, Apache will return the `Premature end of script headers` error if you try to run it through the server. See [Writing a CGI program](#) above for more details.

## Error logs

The error logs are your friend. Anything that goes wrong generates message in the error log. You should always look there first. If the place where you are hosting your web site does not permit you access to the error log, you should probably host your site somewhere else. Learn to read the error logs, and you'll find that almost all of your problems are quickly identified, and quickly solved.

## Suexec

The [suexec](#) support program allows CGI programs to be run under different user permissions, depending on which virtual host or user home directory they are located in. Suexec has very strict permission checking, and any failure in that checking will result in your CGI programs failing with `Premature end of script headers`.

To check if you are using suexec, run `apachectl -V` and check for the location of SUEXEC\_BIN. If Apache finds an [suexec](#) binary there on startup, suexec will be activated.

Unless you fully understand suexec, you should not be using it. To disable suexec, simply remove (or rename) the [suexec](#) binary pointed to by SUEXEC\_BIN and then restart the server. If, after reading about [suexec](#), you still wish to use it, then run `suexec -V` to find the location of the suexec log file, and use that log file to find what policy you are violating.



As you become more advanced in CGI programming, it will become useful to understand more about what's happening behind the scenes. Specifically, how the browser and server communicate with one another. Because although it's all very well to write a program that prints "Hello, World.", it's not particularly useful.

## Environment variables

Environment variables are values that float around you as you use your computer. They are useful things like your path (where the computer searches for the actual file implementing a command when you type it), your username, your terminal type, and so on. For a full list of your normal, every day environment variables, type `env` at a command prompt.

During the CGI transaction, the server and the browser also set environment variables, so that they can communicate with one another. These are things like the browser type (Netscape, IE, Lynx), the server type (Apache, IIS, WebSite), the name of the CGI program that is being run, and so on.

These variables are available to the CGI programmer, and are half of the story of the client-server communication. The complete list of required variables is at

<http://hoohoo.ncsa.uiuc.edu/cgi/env.html>.

This simple Perl CGI program will display all of the environment variables that are being passed around. Two similar programs are included in the `cgi-bin` directory of the Apache distribution. Note that some variables are required, while others are optional, so you may see some variables listed that were not in the official list. In addition, Apache provides many different ways for you to [add your own environment variables](#) to the basic ones provided by default.

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
foreach $key (keys %ENV) {
    print "$key --> $ENV{$key}<br>";
}
```

## STDIN and STDOUT

Other communication between the server and the client happens over standard input (STDIN) and standard output (STDOUT). In normal everyday context, STDIN means the keyboard, or a file that a program is given to act on, and STDOUT usually means the console or screen.

When you POST a web form to a CGI program, the data in that form is bundled up into a special format and gets delivered to your CGI program over STDIN. The program then can process that data as though it was coming in from the keyboard, or from a file

The "special format" is very simple. A field name and its value are joined together with an equals (=) sign, and pairs of values are joined together with an ampersand (&). Inconvenient characters like spaces, ampersands, and equals signs, are converted into their hex equivalent so that they don't gum up the works. The whole data string might look something like:

```
name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Squirrel%20Mor
```

You'll sometimes also see this type of string appended to a URL. When that is done, the server puts that string into the environment variable called QUERY\_STRING. That's called a GET request. Your HTML form specifies whether a GET or a POST is used to deliver the data, by setting the METHOD attribute in the FORM tag.

Your program is then responsible for splitting that string up into

useful information. Fortunately, there are libraries and modules available to help you process this data, as well as handle other of the aspects of your CGI program.



---

When you write CGI programs, you should consider using a code library, or module, to do most of the grunt work for you. This leads to fewer errors, and faster development.

If you're writing CGI programs in Perl, modules are available on [CPAN](#). The most popular module for this purpose is `CGI.pm`. You might also consider `CGI::Lite`, which implements a minimal set of functionality, which is all you need in most programs.

If you're writing CGI programs in C, there are a variety of options. One of these is the CGIC library, from <http://www.boutell.com/cgic/>.

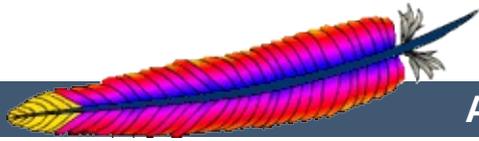


There are a large number of CGI resources on the web. You can discuss CGI problems with other users on the Usenet group [comp.infosystems.www.authoring.cgi](mailto:comp.infosystems.www.authoring.cgi). And the -servers mailing list from the HTML Writers Guild is a great source of answers to your questions. You can find out more at <http://www.hwg.org/lists/hwg-servers/>.

And, of course, you should probably read the CGI specification, which has all the details on the operation of CGI programs. You can find the original version at the [NCSA](#) and there is an updated draft at the [Common Gateway Interface RFC project](#).

When you post a question about a CGI problem that you're having, whether to a mailing list, or to a newsgroup, make sure you provide enough information about what happened, what you expected to happen, and how what actually happened was different, what server you're running, what language your CGI program was in, and, if possible, the offending code. This will make finding your problem much simpler.

Note that questions about CGI problems should **never** be posted to the Apache bug database unless you are sure you have found a problem in the Apache source code.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

## **Apache Tutorial: Introduction to Server Side Includes**

Server-side includes provide a means to add dynamic content to existing HTML documents.



Related Modules	Related Directives
<a href="#">mod_include</a>	<a href="#">Options</a>
<a href="#">mod_cgi</a>	<a href="#">XBitHack</a>
<a href="#">mod_expires</a>	<a href="#">AddType</a>
	<a href="#">SetOutputFilter</a>
	<a href="#">BrowserMatchNoCase</a>

This article deals with Server Side Includes, usually called simply SSI. In this article, I'll talk about configuring your server to permit SSI, and introduce some basic SSI techniques for adding dynamic content to your existing HTML pages.

In the latter part of the article, we'll talk about some of the somewhat more advanced things that can be done with SSI, such as conditional statements in your SSI directives.



---

SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology.

The decision of when to use SSI, and when to have your page entirely generated by some program, is usually a matter of how much of the page is static, and how much needs to be recalculated every time the page is served. SSI is a great way to add small pieces of information, such as the current time. But if a majority of your page is being generated at the time that it is served, you need to look for some other solution.



To permit SSI on your server, you must have the following directive either in your `httpd.conf` file, or in a `.htaccess` file:

```
Options +Includes
```

This tells Apache that you want to permit files to be parsed for SSI directives. Note that most configurations contain multiple `Options` directives that can override each other. You will probably need to apply the `Options` to the specific directory where you want SSI enabled in order to assure that it gets evaluated last.

Not just any file is parsed for SSI directives. You have to tell Apache which files should be parsed. There are two ways to do this. You can tell Apache to parse any file with a particular file extension, such as `.shtml`, with the following directives:

```
AddType text/html .shtml  
AddOutputFilter INCLUDES .shtml
```

One disadvantage to this approach is that if you wanted to add SSI directives to an existing page, you would have to change the name of that page, and all links to that page, in order to give it a `.shtml` extension, so that those directives would be executed.

The other method is to use the `XBitHack` directive:

```
XBitHack on
```

`XBitHack` tells Apache to parse files for SSI directives if they have the execute bit set. So, to add SSI directives to an existing page, rather than having to change the file name, you would just need to make the file executable using `chmod`.

```
chmod +x pagename.html
```

---

A brief comment about what not to do. You'll occasionally see people recommending that you just tell Apache to parse all `.html` files for SSI, so that you don't have to mess with `.shtml` file names. These folks have perhaps not heard about [XBitHack](#). The thing to keep in mind is that, by doing this, you're requiring that Apache read through every single file that it sends out to clients, even if they don't contain any SSI directives. This can slow things down quite a bit, and is not a good idea.

Of course, on Windows, there is no such thing as an execute bit to set, so that limits your options a little.

In its default configuration, Apache does not send the last modified date or content length HTTP headers on SSI pages, because these values are difficult to calculate for dynamic content. This can prevent your document from being cached, and result in slower perceived client performance. There are two ways to solve this:

1. Use the `XBitHack Full` configuration. This tells Apache to determine the last modified date by looking only at the date of the originally requested file, ignoring the modification date of any included files.
2. Use the directives provided by [mod\\_expires](#) to set an explicit expiration time on your files, thereby letting browsers and proxies know that it is acceptable to cache them.



SSI directives have the following syntax:

```
<!--#element attribute=value attribute=value ... -->
```

It is formatted like an HTML comment, so if you don't have SSI correctly enabled, the browser will ignore it, but it will still be visible in the HTML source. If you have SSI correctly configured, the directive will be replaced with its results.

The element can be one of a number of things, and we'll talk some more about most of these in the next installment of this series. For now, here are some examples of what you can do with SSI

## Today's date

```
<!--#echo var="DATE_LOCAL" -->
```

The echo element just spits out the value of a variable. There are a number of standard variables, which include the whole set of environment variables that are available to CGI programs. Also, you can define your own variables with the set element.

If you don't like the format in which the date gets printed, you can use the config element, with a timefmt attribute, to modify that formatting.

```
<!--#config timefmt="%A %B %d, %Y" -->  
Today is <!--#echo var="DATE_LOCAL" -->
```

## Modification date of the file

```
This document last modified <!--#flastmod file="index.html" -->
```

This element is also subject to `timefmt` format configurations.

## Including the results of a CGI program

This is one of the more common uses of SSI - to output the results of a CGI program, such as everybody's favorite, a ``hit counter."

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```



## Additional examples

Following are some specific examples of things you can do in your HTML documents with SSI.

### When was this document modified?

Earlier, we mentioned that you could use SSI to inform the user when the document was most recently modified. However, the actual method for doing that was left somewhat in question. The following code, placed in your HTML document, will put such a time stamp on your page. Of course, you will have to have SSI correctly enabled, as discussed above.

```
<!--#config timefmt="%A %B %d, %Y" -->  
This file last modified <!--#flastmod file="ssi.shtml" -->
```

Of course, you will need to replace the `ssi.shtml` with the actual name of the file that you're referring to. This can be inconvenient if you're just looking for a generic piece of code that you can paste into any file, so you probably want to use the `LAST_MODIFIED` variable instead:

```
<!--#config timefmt="%D" -->  
This file last modified <!--#echo var="LAST_MODIFIED" -->
```

For more details on the `timefmt` format, go to your favorite search site and look for `strftime`. The syntax is the same.

### Including a standard footer

If you are managing any site that is more than a few pages, you may find that making changes to all those pages can be a real pain, particularly if you are trying to maintain some kind of standard look across all those pages.

Using an include file for a header and/or a footer can reduce the

burden of these updates. You just have to make one footer file, and then include it into each page with the `include` SSI command. The `include` element can determine what file to include with either the `file` attribute, or the `virtual` attribute. The `file` attribute is a file path, *relative to the current directory*. That means that it cannot be an absolute file path (starting with `/`), nor can it contain `../` as part of that path. The `virtual` attribute is probably more useful, and should specify a URL relative to the document being served. It can start with a `/`, but must be on the same server as the file being served.

```
<!--#include virtual="/footer.html" -->
```

I'll frequently combine the last two things, putting a `LAST_MODIFIED` directive inside a footer file to be included. SSI directives can be contained in the included file, and includes can be nested - that is, the included file can include another file, and so on.



## What else can we config?

In addition to being able to config the time format, you can also config two other things.

Usually, when something goes wrong with your SSI directive, you get the message

```
[an error occurred while processing this directive]
```

If you want to change that message to something else, you can do so with the `errmsg` attribute to the `config` element:

```
<!--#config errmsg="[It appears that you don't know how to use SSI]" -->
```

Hopefully, end users will never see this message, because you will have resolved all the problems with your SSI directives before your site goes live. (Right?)

And you can config the format in which file sizes are returned with the `sizefmt` attribute. You can specify `bytes` for a full count in bytes, or `abbrev` for an abbreviated number in Kb or Mb, as appropriate.



## Executing Commands

I expect that I'll have an article some time in the coming months about using SSI with small CGI programs. For now, here's something else that you can do with the exec element. You can actually have SSI execute a command using the shell (/bin/sh, to be precise - or the DOS shell, if you're on Win32). The following, for example, will give you a directory listing.

```
<pre>  
<!--#exec cmd="ls" -->  
</pre>
```

or, on Windows

```
<pre>  
<!--#exec cmd="dir" -->  
</pre>
```

You might notice some strange formatting with this directive on Windows, because the output from dir contains the string ``<dir>" in it, which confuses browsers.

Note that this feature is exceedingly dangerous, as it will execute whatever code happens to be embedded in the exec tag. If you have any situation where users can edit content on your web pages, such as with a ``guestbook", for example, make sure that you have this feature disabled. You can allow SSI, but not the exec feature, with the IncludesNOEXEC argument to the Options directive.



In addition to spitting out content, Apache SSI gives you the option of setting variables, and using those variables in comparisons and conditionals.

## Caveat

Most of the features discussed in this article are only available to you if you are running Apache 1.2 or later. Of course, if you are not running Apache 1.2 or later, you need to upgrade immediately, if not sooner. Go on. Do it now. We'll wait.

## Setting variables

Using the `set` directive, you can set variables for later use. We'll need this later in the discussion, so we'll talk about it here. The syntax of this is as follows:

```
<!--#set var="name" value="Rich" -->
```

In addition to merely setting values literally like that, you can use any other variable, including [environment variables](#) or the variables discussed above (like `LAST_MODIFIED`, for example) to give values to your variables. You will specify that something is a variable, rather than a literal string, by using the dollar sign (\$) before the name of the variable.

```
<!--#set var="modified" value="$LAST_MODIFIED" -->
```

To put a literal dollar sign into the value of your variable, you need to escape the dollar sign with a backslash.

```
<!--#set var="cost" value="\$100" -->
```

Finally, if you want to put a variable in the midst of a longer string,

and there's a chance that the name of the variable will run up against some other characters, and thus be confused with those characters, you can place the name of the variable in braces, to remove this confusion. (It's hard to come up with a really good example of this, but hopefully you'll get the point.)

```
<!--#set var="date" value="{DATE_LOCAL}_{DATE_GMT}" -->
```

## Conditional expressions

Now that we have variables, and are able to set and compare their values, we can use them to express conditionals. This lets SSI be a tiny programming language of sorts. [mod\\_include](#) provides an `if`, `elif`, `else`, `endif` structure for building conditional statements. This allows you to effectively generate multiple logical pages out of one actual page.

The structure of this conditional construct is:

```
<!--#if expr="test_condition" -->  
<!--#elif expr="test_condition" -->  
<!--#else -->  
<!--#endif -->
```

A *test\_condition* can be any sort of logical comparison - either comparing values to one another, or testing the "truth" of a particular value. (A given string is true if it is nonempty.) For a full list of the comparison operators available to you, see the [mod\\_include](#) documentation. Here are some examples of how one might use this construct.

In your configuration file, you could put the following line:

```
BrowserMatchNoCase macintosh Mac  
BrowserMatchNoCase MSIE InternetExplorer
```

This will set environment variables ``Mac" and ``InternetExplorer" to true, if the client is running Internet Explorer on a Macintosh.

Then, in your SSI-enabled document, you might do the following:

```
<!--#if expr="${Mac} && ${InternetExplorer}" -->
Apologetic text goes here
<!--#else -->
Cool JavaScript code goes here
<!--#endif -->
```

Not that I have anything against IE on Macs - I just struggled for a few hours last week trying to get some JavaScript working on IE on a Mac, when it was working everywhere else. The above was the interim workaround.

Any other variable (either ones that you define, or normal environment variables) can be used in conditional statements. With Apache's ability to set environment variables with the `SetEnvIf` directives, and other related directives, this functionality can let you do some pretty involved dynamic stuff without ever resorting to CGI.

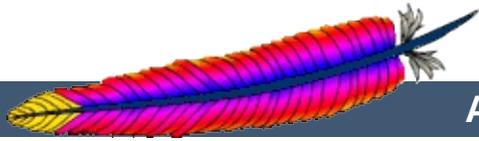


SSl is certainly not a replacement for CGI, or other technologies used for generating dynamic web pages. But it is a great way to add small amounts of dynamic content to pages, without doing a lot of extra work.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

## Apache Tutorial: .htaccess files

.htaccess files provide a way to make configuration changes on a per-directory basis.



Related Modules	Related Directives
<a href="#">core</a>	<a href="#">AccessFileName</a>
<a href="#">mod_auth</a>	<a href="#">AllowOverride</a>
<a href="#">mod_cgi</a>	<a href="#">Options</a>
<a href="#">mod_include</a>	<a href="#">AddHandler</a>
<a href="#">mod_mime</a>	<a href="#">SetHandler</a>
	<a href="#">AuthType</a>
	<a href="#">AuthName</a>
	<a href="#">AuthUserFile</a>
	<a href="#">AuthGroupFile</a>
	<a href="#">Require</a>



.htaccess files (or "distributed configuration files") provide a way to make configuration changes on a per-directory basis. A file, containing one or more configuration directives, is placed in a particular document directory, and the directives apply to that directory, and all subdirectories thereof.

**Note:**

If you want to call your .htaccess file something else, you can change the name of the file using the [AccessFileName](#) directive. For example, if you would rather call the file .config then you can put the following in your server configuration file:

```
AccessFileName .config
```

In general, .htaccess files use the same syntax as the [main configuration files](#). What you can put in these files is determined by the [AllowOverride](#) directive. This directive specifies, in categories, what directives will be honored if they are found in a .htaccess file. If a directive is permitted in a .htaccess file, the documentation for that directive will contain an [Override](#) section, specifying what value must be in [AllowOverride](#) in order for that directive to be permitted.

For example, if you look at the documentation for the [AddDefaultCharset](#) directive, you will find that it is permitted in .htaccess files. (See the Context line in the directive summary.) The [Override](#) line reads FileInfo. Thus, you must have at least `AllowOverride FileInfo` in order for this directive to be honored in .htaccess files.

**Example:**

Context: server config, virtual host, directory, .htaccess  
Override: FileInfo

If you are unsure whether a particular directive is permitted in a .htaccess file, look at the documentation for that directive, and check the Context line for ".htaccess".



---

In general, you should never use `.htaccess` files unless you don't have access to the main server configuration file. There is, for example, a prevailing misconception that user authentication should always be done in `.htaccess` files. This is simply not the case. You can put user authentication configurations in the main server configuration, and this is, in fact, the preferred way to do things.

`.htaccess` files should be used in a case where the content providers need to make configuration changes to the server on a per-directory basis, but do not have root access on the server system. In the event that the server administrator is not willing to make frequent configuration changes, it might be desirable to permit individual users to make these changes in `.htaccess` files for themselves. This is particularly true, for example, in cases where ISPs are hosting multiple user sites on a single machine, and want their users to be able to alter their configuration.

However, in general, use of `.htaccess` files should be avoided when possible. Any configuration that you would consider putting in a `.htaccess` file, can just as effectively be made in a `<Directory>` section in your main server configuration file.

There are two main reasons to avoid the use of `.htaccess` files.

The first of these is performance. When `AllowOverride` is set to allow the use of `.htaccess` files, Apache will look in every directory for `.htaccess` files. Thus, permitting `.htaccess` files causes a performance hit, whether or not you actually even use them! Also, the `.htaccess` file is loaded every time a document is requested.

Further note that Apache must look for `.htaccess` files in all

higher-level directories, in order to have a full complement of directives that it must apply. (See section on [how directives are applied](#).) Thus, if a file is requested out of a directory `/www/htdocs/example`, Apache must look for the following files:

```
/.htaccess
/www/.htaccess
/www/htdocs/.htaccess
/www/htdocs/example/.htaccess
```

And so, for each file access out of that directory, there are 4 additional file-system accesses, even if none of those files are present. (Note that this would only be the case if `.htaccess` files were enabled for `/`, which is not usually the case.)

The second consideration is one of security. You are permitting users to modify server configuration, which may result in changes over which you have no control. Carefully consider whether you want to give your users this privilege. Note also that giving users less privileges than they need will lead to additional technical support requests. Make sure you clearly tell your users what level of privileges you have given them. Specifying exactly what you have set `AllowOverride` to, and pointing them to the relevant documentation, will save yourself a lot of confusion later.

Note that it is completely equivalent to put a `.htaccess` file in a directory `/www/htdocs/example` containing a directive, and to put that same directive in a Directory section `<Directory /www/htdocs/example>` in your main server configuration:

`.htaccess` file in `/www/htdocs/example`:

### **Contents of `.htaccess` file in `/www/htdocs/example`**

```
AddType text/example .exm
```

## Section from your httpd.conf file

```
<Directory /www/htdocs/example>  
  AddType text/example .exm  
</Directory>
```

However, putting this configuration in your server configuration file will result in less of a performance hit, as the configuration is loaded once when Apache starts, rather than every time a file is requested.

The use of .htaccess files can be disabled completely by setting the [AllowOverride](#) directive to none:

```
AllowOverride None
```



## How directives are applied

The configuration directives found in a `.htaccess` file are applied to the directory in which the `.htaccess` file is found, and to all subdirectories thereof. However, it is important to also remember that there may have been `.htaccess` files in directories higher up. Directives are applied in the order that they are found. Therefore, a `.htaccess` file in a particular directory may override directives found in `.htaccess` files found higher up in the directory tree. And those, in turn, may have overridden directives found yet higher up, or in the main server configuration file itself.

Example:

In the directory `/www/htdocs/example1` we have a `.htaccess` file containing the following:

```
Options +ExecCGI
```

(Note: you must have "AllowOverride Options" in effect to permit the use of the "[Options](#)" directive in `.htaccess` files.)

In the directory `/www/htdocs/example1/example2` we have a `.htaccess` file containing:

```
Options Includes
```

Because of this second `.htaccess` file, in the directory `/www/htdocs/example1/example2`, CGI execution is not permitted, as only `Options Includes` is in effect, which completely overrides any earlier setting that may have been in place.

## Merging of `.htaccess` with the main configuration files

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:

```
<Directory />
  Allowoverride All
</Directory>

<Location />
  Options +IncludesNoExec -ExecCGI
</Location>
```



## Authentication Example

If you jumped directly to this part of the document to find out how to do authentication, it is important to note one thing. There is a common misconception that you are required to use `.htaccess` files in order to implement password authentication. This is not the case. Putting authentication directives in a `<Directory>` section, in your main server configuration file, is the preferred way to implement this, and `.htaccess` files should be used only if you don't have access to the main server configuration file. See [above](#) for a discussion of when you should and should not use `.htaccess` files.

Having said that, if you still think you need to use a `.htaccess` file, you may find that a configuration such as what follows may work for you.

You must have `AllowOverride AuthConfig` in effect for these directives to be honored.

`.htaccess` file contents:

```
AuthType Basic
AuthName "Password Required"
AuthUserFile /www/passwords/password.file
AuthGroupFile /www/passwords/group.file
Require Group admins
```

Note that `AllowOverride AuthConfig` must be in effect for these directives to have any effect.

Please see the [authentication tutorial](#) for a more complete discussion of authentication and authorization.



## Server Side Includes Example

Another common use of .htaccess files is to enable Server Side Includes for a particular directory. This may be done with the following configuration directives, placed in a .htaccess file in the desired directory:

```
Options +Includes
AddType text/html shtml
AddHandler server-parsed shtml
```

Note that AllowOverride Options and AllowOverride FileInfo must both be in effect for these directives to have any effect.

Please see the [SSI tutorial](#) for a more complete discussion of server-side includes.



Finally, you may wish to use a `.htaccess` file to permit the execution of CGI programs in a particular directory. This may be implemented with the following configuration:

```
Options +ExecCGI
AddHandler cgi-script cgi pl
```

Alternately, if you wish to have all files in the given directory be considered to be CGI programs, this may be done with the following configuration:

```
Options +ExecCGI
SetHandler cgi-script
```

Note that `AllowOverride Options` and `AllowOverride FileInfo` must both be in effect for these directives to have any effect.

Please see the [CGI tutorial](#) for a more complete discussion of CGI programming and configuration.



## Troubleshooting

When you put configuration directives in a `.htaccess` file, and you don't get the desired effect, there are a number of things that may be going wrong.

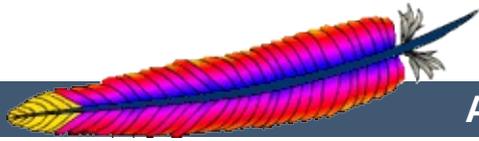
Most commonly, the problem is that `AllowOverride` is not set such that your configuration directives are being honored. Make sure that you don't have a `AllowOverride None` in effect for the file scope in question. A good test for this is to put garbage in your `.htaccess` file and reload. If a server error is not generated, then you almost certainly have `AllowOverride None` in effect.

If, on the other hand, you are getting server errors when trying to access documents, check your Apache error log. It will likely tell you that the directive used in your `.htaccess` file is not permitted. Alternately, it may tell you that you had a syntax error, which you will then need to fix.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [How-To / Tutorials](#)

## Per-user web directories

On systems with multiple users, each user can be permitted to have a web site in their home directory using the [UserDir](#) directive. Visitors to a URL `http://example.com/~username/` will get content out of the home directory of the user "username", out of the subdirectory specified by the [UserDir](#) directive.

### See also

[Mapping URLs to the Filesystem](#)



## FOR USERDIR DIRECTIVES

Related Modules	Related Directives
<a href="#">mod_userdir</a>	<a href="#">UserDir</a>
	<a href="#">DirectoryMatch</a>
	<a href="#">AllowOverride</a>



The `UserDir` directive specifies a directory out of which per-user content is loaded. This directive may take several different forms.

If a path is given which does not start with a leading slash, it is assumed to be a directory path relative to the home directory of the specified user. Given this configuration:

```
UserDir public_html
```

the URL `http://example.com/~rbowen/file.html` will be translated to the file path `/home/rbowen/public_html/file.html`

If a path is given starting with a slash, a directory path will be constructed using that path, plus the username specified. Given this configuration:

```
UserDir /var/html
```

the URL `http://example.com/~rbowen/file.html` will be translated to the file path `/var/html/rbowen/file.html`

If a path is provided which contains an asterisk (\*), a path is used in which the asterisk is replaced with the username. Given this configuration:

```
UserDir /var/www/*/docs
```

the URL `http://example.com/~rbowen/file.html` will be translated to the file path `/var/www/rbowen/docs/file.html`



## feature

Using the syntax shown in the UserDir documentation, you can restrict what users are permitted to use this functionality:

```
UserDir enabled  
UserDir disabled root jro fish
```

The configuration above will enable the feature for all users except for those listed in the `disabled` statement. You can, likewise, disable the feature for all but a few users by using a configuration like the following:

```
UserDir disabled  
UserDir enabled rbowen krietz
```

See [UserDir](#) documentation for additional examples.



## Enabling a cgi-bin directory for each user

In order to give each user their own cgi-bin directory, you can use a `<Directory>` directive to make a particular subdirectory of a user's home directory cgi-enabled.

```
<Directory /home/*/public_html/cgi-bin/>  
Options ExecCGI  
SetHandler cgi-script  
</Directory>
```

Then, presuming that `UserDir` is set to `public_html`, a cgi program `example.cgi` could be loaded from that directory as:

```
http://example.com/~rbowen/cgi-bin/example.cgi
```



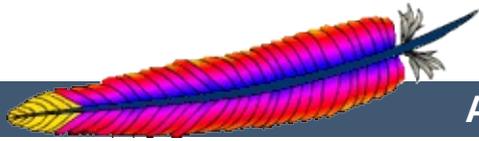
## Allowing users to alter configuration

If you want to allow users to modify the server configuration in their web space, they will need to use `.htaccess` files to make these changes. Ensure that you have set `AllowOverride` to a value sufficient for the directives that you want to permit the users to modify. See the [.htaccess tutorial](#) for additional details on how this works.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

# Apache Tutorials

**Warning:**

This document has not been fully updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

The following documents give you step-by-step instructions on how to accomplish common tasks with the Apache HTTP server. Many of these documents are located at external sites and are not the work of the Apache Software Foundation. Copyright to documents on external sites is owned by the authors or their assignees. Please consult the [official Apache Server documentation](#) to verify what you read on external sites.



## Installation and Getting Started

- [Getting Started with Apache 1.3](#) (ApacheToday)
- [Configuring Your Apache Server Installation](#) (ApacheToday)
- [Getting, Installing, and Running Apache \(on Unix\)](#) (O'Reilly Network Apache DevCenter)
- [Maximum Apache: Getting Started](#) (CNET Builder.com)
- [How to Build the Apache of Your Dreams](#) (Developer Shed)



## Basic Configuration

- [An Amble Through Apache Configuration](#) (O'Reilly Network Apache DevCenter)
- [Using .htaccess Files with Apache](#) (ApacheToday)
- [Setting Up Virtual Hosts](#) (ApacheToday)
- [Maximum Apache: Configure Apache](#) (CNET Builder.com)
- [Getting More Out of Apache](#) (Developer Shed)



- [Security and Apache: An Essential Primer](#) (LinuxPlanet)
- [Using User Authentication](#) (Apacheweek)
- [DBM User Authentication](#) (Apacheweek)
- [An Introduction to Securing Apache](#) (Linux.com)
- [Securing Apache - Access Control](#) (Linux.com)
- Apache Authentication [Part 1](#) - [Part 2](#) - [Part 3](#) - [Part 4](#) (ApacheToday)
- [mod\\_access: Restricting Access by Host](#) (ApacheToday)



## Logging

- [Log Rhythms](#) (O'Reilly Network Apache DevCenter)
- [Gathering Visitor Information: Customising Your Logfiles](#) (Apacheweek)
- Apache Guide: Logging [Part 1](#) - [Part 2](#) - [Part 3](#) - [Part 4](#) - [Part 5](#) (ApacheToday)



- 
- [Dynamic Content with CGI](#) (ApacheToday)
  - [The Idiot's Guide to Solving Perl CGI Problems](#) (CPAN)
  - [Executing CGI Scripts as Other Users](#) (LinuxPlanet)
  - [CGI Programming FAQ](#) (Web Design Group)
  - Introduction to Server Side Includes [Part 1](#) - [Part 2](#) (ApacheToday)
  - [Advanced SSI Techniques](#) (ApacheToday)
  - [Setting up CGI and SSI with Apache](#) (CNET Builder.com)



## Other Resources

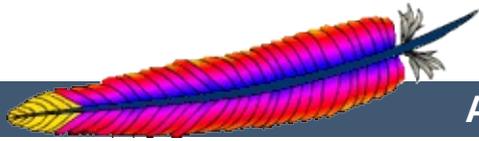
- [Content Negotiation Explained](#) (Apacheweek)
- [Using Apache Imagemaps](#) (Apacheweek)
- [Keeping Your Images from Adorning Other Sites](#) (ApacheToday)
- [Language Negotiation Notes](#) (Alan J. Flavell)

If you have a pointer to an accurate and well-written tutorial not included here, please let us know by submitting it to the [Apache Bug Database](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

## Using Apache with Microsoft Windows

This document explains how to install, configure and run Apache 2.0 under Microsoft Windows. If you find any bugs, or wish to contribute in other ways, please use our [bug reporting page](#).

This document assumes that you are installing a binary distribution of Apache. If you want to compile Apache yourself (possibly to help with development or tracking down bugs), see [Compiling Apache for Microsoft Windows](#).

**Because of the current versioning policies on Microsoft Windows operating system families, this document assumes the following:**

- **Windows NT:** This means all versions of Windows that are based on the Windows NT kernel. Includes Windows NT, Windows 2000, Windows XP and Windows .Net Server 2003.
- **Windows 9x:** This means older, consumer-oriented versions of Windows. Includes Windows 95 (also OSR2), Windows 98 and Windows ME.



## Operating System Requirements

The primary Windows platform for running Apache 2.0 is Windows NT. The binary installer only works with the x86 family of processors, such as Intel and AMD processors. Running Apache on Windows 9x is not thoroughly tested, and it is never recommended on production systems.

On all operating systems, TCP/IP networking must be installed and working. If running on Windows 95, the Winsock 2 upgrade must be installed. Winsock 2 for Windows 95 can be downloaded from [here](#).

On Windows NT 4.0, installing Service Pack 6 is strongly recommended, as Service Pack 4 created known issues with TCP/IP and Winsock integrity that were resolved in later Service Packs.



## Installing Apache for Windows

Information on the latest versions of Apache can be found on the web site of the Apache web server at <http://httpd.apache.org/download.cgi>. There you will find the current release, as well as more recent alpha or beta test versions, and a list of HTTP and FTP mirrors from which you can download the Apache web server. Please use a mirror near to you for a fast and reliable download.

For Windows installations you should download the version of Apache for Windows with the `.msi` extension. This is a single Microsoft Installer file, which contains a ready-to-run version of Apache. There is a separate `.zip` file, which contains only the source code. You can compile Apache yourself with the Microsoft Visual C++ (Visual Studio) tools.



## Installing Apache for Windows

You need Microsoft Installer 1.2 or above for the installation to work. On Windows 9x you can update your Microsoft Installer to version 2.0 [here](#) and on Windows NT 4.0 and 2000 the version 2.0 update can be found [here](#). Windows XP does not need this update.

Note that you cannot install two versions of Apache 2.0 on the same computer with the binary installer. You can, however, install a version of the 1.3 series **and** a version of the 2.0 series on the same computer without problems. If you need to have two different 2.0 versions on the same computer, you have to [compile and install Apache from the source](#).

Run the Apache .msi file you downloaded above. The installation will ask you for these things:

1. **Network Domain.** Enter the DNS domain in which your server is or will be registered in. For example, if your server's full DNS name is server.mydomain.net, you would type mydomain.net here.
2. **Server Name.** Your server's full DNS name. From the example above, you would type server.mydomain.net here.
3. **Administrator's Email Address.** Enter the server administrator's or webmaster's email address here. This address will be displayed along with error messages to the client by default.
4. **For whom to install Apache** Select for All Users, on Port 80, as a Service - Recommended if you'd like your new Apache to listen at port 80 for incoming traffic. It will run as a service (that is, Apache will run even if no one is

logged in on the server at the moment) Select `only` for the `Current User`, on `Port 8080`, when started `Manually` if you'd like to install Apache for your personal experimenting or if you already have another WWW server running on port 80.

5. **The installation type.** Select `Typical` for everything except the source code and libraries for module development. With `Custom` you can specify what to install. A full install will require about 13 megabytes of free disk space. This does *not* include the size of your web site(s).
6. **Where to install.** The default path is `C:\Program Files\Apache Group` under which a directory called `Apache2` will be created by default.

During the installation, Apache will configure the files in the `conf` subdirectory to reflect the chosen installation directory. However, if any of the configuration files in this directory already exist, they will not be overwritten. Instead, the new copy of the corresponding file will be left with the extension `.default`. So, for example, if `conf\httpd.conf` already exists, it will be renamed as `conf\httpd.conf.default`. After the installation you should manually check to see what new settings are in the `.default` file, and if necessary, update your existing configuration file.

Also, if you already have a file called `htdocs\index.html`, it will not be overwritten (and no `index.html.default` will be installed either). This means it should be safe to install Apache over an existing installation, although you would have to stop the existing running server before doing the installation, and then start the new one after the installation is finished.

After installing Apache, you must edit the configuration files in the

conf subdirectory as required. These files will be configured during the installation so that Apache is ready to be run from the directory it was installed into, with the documents server from the subdirectory ht docs. There are lots of other options which you should set before you really start using Apache. However, to get started quickly, the files should work as installed.



## Configuring Apache for Windows

Apache is configured by the files in the `conf` subdirectory. These are the same files used to configure the Unix version, but there are a few different directives for Apache on Windows. See the [directive index](#) for all the available directives.

The main differences in Apache for Windows are:

- Because Apache for Windows is multithreaded, it does not use a separate process for each request, as Apache does on Unix. Instead there are usually only two Apache processes running: a parent process, and a child which handles the requests. Within the child process each request is handled by a separate thread.

The process management directives are also different:

**`MaxRequestsPerChild`**: Like the Unix directive, this controls how many requests a single child process will serve before exiting. However, unlike on Unix, a single process serves all the requests at once, not just one. If this is set, it is recommended that a very high number is used. The recommended default, `MaxRequestsPerChild 0`, causes the child process to never exit.

**Warning: The server configuration file is reread when a new child process is started. If you have modified `httpd.conf`, the new child may not start or you may receive unexpected results.**

**`ThreadsPerChild`**: This directive is new. It tells the server how many threads it should use. This is the maximum number of connections the server can handle at once, so be sure to set this number high enough for your site if you get a lot of

hits. The recommended default is `ThreadsPerChild 50`.

- The directives that accept filenames as arguments must use Windows filenames instead of Unix ones. However, because Apache uses Unix-style names internally, you must use forward slashes, not backslashes. Drive letters can be used; if omitted, the drive with the Apache executable will be assumed.
- While filenames are generally case-insensitive on Windows, URLs are still treated internally as case-sensitive before they are mapped to the filesystem. For example, the `<Location>`, `Alias`, and `ProxyPass` directives all use case-sensitive arguments. For this reason, it is particularly important to use the `<Directory>` directive when attempting to limit access to content in the filesystem, since this directive applies to any content in a directory, regardless of how it is accessed. If you wish to assure that only lowercase is used in URLs, you can use something like:

```
RewriteEngine On
RewriteMap lowercase int:tolower
RewriteCond %{REQUEST_URI} [A-Z]
RewriteRule (.*) ${lowercase:$1} [R,L]
```

- Apache for Windows contains the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the `\Apache2\modules` directory. To activate these or other modules, the new `LoadModule` directive must be used. For example, to activate the status module, use the following (in addition to the status-activating directives in `access.conf`):

```
LoadModule status_module modules/mod_status.so
```

Information on [creating loadable modules](#) is also available.

- Apache can also load ISAPI (Internet Server Application Programming Interface) extensions (i.e. internet server applications), such as those used by Microsoft IIS and other Windows servers. [More information is available](#). Note that Apache **cannot** load ISAPI Filters.
- When running CGI scripts, the method Apache uses to find the interpreter for the script is configurable using the [ScriptInterpreterSource](#) directive.
- Since it is often difficult to manage files with names like .htaccess in Windows, you may find it useful to change the name of this per-directory configuration file using the [AccessFilename](#) directive.
- Any errors during Apache startup are logged into the Windows event log when running on Windows NT. This mechanism acts as a backup for those situations where Apache cannot even access the normally used error .log file. You can view the Windows event log by using the Event Viewer application on Windows NT 4.0, and the Event Viewer MMC snap-in on newer versions of Windows.

**Note that there is no startup error logging on Windows 9x because no Windows event log exists on those operating systems.**



## Running Apache as a Service

Apache can be run as a service on Windows NT. There is some highly experimental support for similar behavior on Windows 9x.

You can install Apache as a service automatically during the installation. If you chose to install for all users, the installation will create an Apache service for you. If you specify to install for yourself only, you can manually register Apache as a service after the installation. You have to be a member of the Administrators group for the service installation to succeed.

Apache comes with a utility called the Apache Service Monitor. With it you can see and manage the state of all installed Apache services on any machine on your network. To be able to manage an Apache service with the monitor, you have to first install the service (either automatically via the installation or manually).

You can install Apache as a Windows NT service as follows from the command prompt at the Apache `bin` subdirectory:

```
httpd -k install
```

If you need to specify the name of the service you want to install, use the following command. You have to do this if you have several different service installations of Apache on your computer.

```
httpd -k install -n "MyServiceName"
```

If you need to have specifically named configuration files for different services, you must use this:

```
httpd -k install -n "MyServiceName" -f "c:\files\my.conf"
```

If you use the first command without any special parameters except `-k install`, the service will be called Apache2 and the

configuration will be assumed to be `conf\httpd.conf`.

Removing an Apache service is easy. Just use:

```
httpd -k uninstall
```

The specific Apache service to be uninstalled can be specified by using:

```
httpd -k uninstall -n "MyServiceName"
```

Normal starting, restarting and shutting down of an Apache service is usually done via the Apache Service Monitor, by using commands like `NET START Apache2` and `NET STOP Apache2` or via normal Windows service management. Before starting Apache as a service by any means, you should test the service's configuration file by using:

```
httpd -n "MyServiceName" -t
```

You can control an Apache service by its command line switches, too. To start an installed Apache service you'll use this:

```
httpd -k start
```

To stop an Apache service via the command line switches, use this:

```
httpd -k stop
```

or

```
httpd -k shutdown
```

You can also restart a running service and force it to reread its configuration file by using:

```
httpd -k restart
```

By default, all Apache services are registered to run as the system user (the LocalSystem account). The LocalSystem account has no privileges to your network via any Windows-secured mechanism, including the file system, named pipes, DCOM, or secure RPC. It has, however, wide privileges locally.

**Never grant any network privileges to the LocalSystem account! If you need Apache to be able to access network resources, create a separate account for Apache as noted below.**

You may want to create a separate account for running Apache service(s). Especially, if you have to access network resources via Apache, this is strongly recommended.

1. Create a normal domain user account, and be sure to memorize its password.
2. Grant the newly-created user a privilege of Log on as a service and Act as part of the operating system. On Windows NT 4.0 these privileges are granted via User Manager for Domains, but on Windows 2000 and XP you probably want to use Group Policy for propagating these settings. You can also manually set these via the Local Security Policy MMC snap-in.
3. Confirm that the created account is a member of the Users group.
4. Grant the account read and execute (RX) rights to all document and script folders (htdocs and cgi-bin for

example).

5. Grant the account change (RWXD) rights to the Apache logs directory.
6. Grant the account read and execute (RX) rights to the Apache .exe binary executable.

It is usually a good practice to grant the user the Apache service runs as read and execute (RX) access to the whole Apache2 directory, except the logs subdirectory, where the user has to have at least change (RWXD) rights.

If you allow the account to log in as a user and as a service, then you can log on with that account and test that the account has the privileges to execute the scripts, read the web pages, and that you can start Apache in a console window. If this works, and you have followed the steps above, Apache should execute as a service with no problems.

**Error code 2186** is a good indication that you need to review the "Log On As" configuration for the service, since Apache cannot access a required network resource. Also, pay close attention to the privileges of the user Apache is configured to run as.

When starting Apache as a service you may encounter an error message from the Windows Service Control Manager. For example, if you try to start Apache by using the Services applet in the Windows Control Panel, you may get the following message:

```
Could not start the Apache2 service on \\COMPUTER
Error 1067; The process terminated unexpectedly.
```

You will get this generic error if there is any problem with starting the Apache service. In order to see what is really causing the

problem you should follow the instructions for Running Apache for Windows from the Command Prompt.

There is some support for Apache on Windows 9x to behave in a similar manner as a service on Windows NT. It is **highly experimental**. It is not of production-class reliability, and its future is not guaranteed. It can be mostly regarded as a risky thing to play with - proceed with caution!

There are some differences between the two kinds of services you should be aware of:

- Apache will attempt to start and if successful it will run in the background. If you run the command

```
httpd -n "MyServiceName" -k start
```

via a shortcut on your desktop, for example, then if the service starts successfully, a console window will flash up but it immediately disappears. If Apache detects any errors on startup such as incorrect entries in the httpd.conf configuration file, the console window will remain visible. This will display an error message which will be useful in tracking down the cause of the problem.

- Windows 9x does not support NET START or NET STOP commands. You must control the Apache service on the command prompt via the -k switches.
- Apache and Windows 9x offer no support for running Apache as a specific user with network privileges. In fact, Windows 9x offers no security on the local machine, either. This is the simple reason because of which the Apache Software Foundation never endorses use of a Windows 9x -based system as a public Apache server. The primitive support for

Windows 9x exists only to assist the user in developing web content and learning the Apache server, and perhaps as an intranet server on a secured, private network.

Once you have confirmed that Apache runs correctly as a console application you can install, control and uninstall the pseudo-service with the same commands as on Windows NT. You can also use the Apache Service Monitor to manage Windows 9x pseudo-services.



## Running Apache as a Console Application

Running Apache as a service is usually the recommended way to use it, but it is sometimes easier to work from the command line (on Windows 9x running Apache from the command line is the recommended way due to the lack of reliable service support.)

To run Apache from the command line as a console application, use the following command:

```
httpd
```

Apache will execute, and will remain running until it is stopped by pressing Control-C.

You can also run Apache via the shortcut Start Apache in Console placed to Start Menu --> Programs --> Apache HTTP Server 2.0.xx --> Control Apache Server during the installation. This will open a console window and start Apache inside it. If you don't have Apache installed as a service, the window will remain visible until you stop Apache by pressing Control-C in the console window where Apache is running in. The server will exit in a few seconds. However, if you do have Apache installed as a service, the shortcut starts the service. If the Apache service is running already, the shortcut doesn't do anything.

You can tell a running Apache to stop by opening another console window and entering:

```
httpd -k shutdown
```

This should be preferred over pressing Control-C because this lets Apache end any current operations and clean up gracefully.

You can also tell Apache to restart. This forces it to reread the configuration file. Any operations in progress are allowed to

complete without interruption. To restart Apache, use:

```
httpd -k restart
```

Note for people familiar with the Unix version of Apache: these commands provide a Windows equivalent to `kill -TERM pid` and `kill -USR1 pid`. The command line option used, `-k`, was chosen as a reminder of the `kill` command used on Unix.

If the Apache console window closes immediately or unexpectedly after startup, open the Command Prompt from the Start Menu --> Programs. Change to the folder to which you installed Apache, type the command `apache`, and read the error message. Then change to the logs folder, and review the `error.log` file for configuration mistakes. If you accepted the defaults when you installed Apache, the commands would be:

```
c:  
cd "\\Program Files\\Apache Group\\Apache2\\bin"  
httpd
```

Then wait for Apache to stop, or press Control-C. Then enter the following:

```
cd ..\logs  
more < error.log
```

When working with Apache it is important to know how it will find the configuration file. You can specify a configuration file on the command line in two ways:

- `-f` specifies an absolute or relative path to a particular configuration file:

```
httpd -f "c:\my server files\anotherconfig.conf"
```

---

or

```
httpd -f files\anotherconfig.conf
```

- -n specifies the installed Apache service whose configuration file is to be used:

```
httpd -n "MyServiceName"
```

In both of these cases, the proper [ServerRoot](#) should be set in the configuration file.

If you don't specify a configuration file with -f or -n, Apache will use the file name compiled into the server, such as `conf\httpd.conf`. This built-in path is relative to the installation directory. You can verify the compiled file name from a value labelled as `SERVER_CONFIG_FILE` when invoking Apache with the -V switch, like this:

```
httpd -V
```

Apache will then try to determine its [ServerRoot](#) by trying the following, in this order:

1. A [ServerRoot](#) directive via the -C command line switch.
2. The -d switch on the command line.
3. Current working directory.
4. A registry entry which was created if you did a binary installation.
5. The server root compiled into the server. This is /apache by default, you can verify it by using `apache -V` and looking for

a value labelled as HTTPD\_ROOT.

During the installation, a version-specific registry key is created in the Windows registry. The location of this key depends on the type of the installation. If you chose to install Apache for all users, the key is located under the HKEY\_LOCAL\_MACHINE hive, like this (the version numbers will of course vary between different versions of Apache:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Apache Group\Apache\2.0.43
```

Correspondingly, if you chose to install Apache for the current user only, the key is located under the HKEY\_CURRENT\_USER hive, the contents of which are dependent of the user currently logged on:

```
HKEY_CURRENT_USER\SOFTWARE\Apache Group\Apache\2.0.43
```

This key is compiled into the server and can enable you to test new versions without affecting the current version. Of course, you must take care not to install the new version in the same directory as another version.

If you did not do a binary install, Apache will in some scenarios complain about the missing registry key. This warning can be ignored if the server was otherwise able to find its configuration file.

The value of this key is the [ServerRoot](#) directory which contains the conf subdirectory. When Apache starts it reads the httpd.conf file from that directory. If this file contains a [ServerRoot](#) directive which contains a different directory from the one obtained from the registry key above, Apache will forget the registry key and use the directory from the configuration file. If you copy the Apache directory or configuration files to a new

location it is vital that you update the ServerRoot directive in the `httpd.conf` file to reflect the new location.



## Testing the installation

After starting Apache (either in a console window or as a service) it will be listening on port 80 (unless you changed the `Listen` directive in the configuration files or installed Apache only for the current user). To connect to the server and access the default page, launch a browser and enter this URL:

```
http://localhost/
```

Apache should respond with a welcome page and a link to the Apache manual. If nothing happens or you get an error, look in the `error.log` file in the `logs` subdirectory. If your host is not connected to the net, or if you have serious problems with your DNS (Domain Name Service) configuration, you may have to use this URL:

```
http://127.0.0.1/
```

If you happen to be running Apache on an alternate port, you need to explicitly put that in the URL:

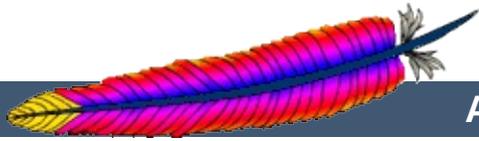
```
http://127.0.0.1:8080/
```

Once your basic installation is working, you should configure it properly by editing the files in the `conf` subdirectory. Again, if you change the configuration of the Windows NT service for Apache, first attempt to start it from the command line to make sure that the service starts with no errors.

Because Apache **cannot** share the same port with another TCP/IP application, you may need to stop, uninstall or reconfigure certain other services before running Apache. These conflicting services include other WWW servers and some firewall implementations.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

## Compiling Apache for Microsoft Windows

There are many important points before you begin compiling Apache. See [Using Apache with Microsoft Windows](#) before you begin.



## Requirements

Compiling Apache requires the following environment to be properly installed:

- Disk Space

Make sure you have at least 50 MB of free disk space available. After installation Apache requires approximately 10 MB of disk space, plus space for log and cache files, which can grow rapidly. The actual disk space requirements will vary considerably based on your chosen configuration and any third-party modules or libraries.

- Microsoft Visual C++ 5.0 or higher.

Apache can be built using the command line tools, or from within the Visual Studio IDE Workbench. The command line build requires the environment to reflect the PATH, INCLUDE, LIB and other variables that can be configured with the vcvars32 batch file:

```
"c:\Program Files\DevStudio\VC\Bin\vcvars32.bat"
```

- The Windows Platform SDK.

Visual C++ 5.0 builds require an updated Microsoft Windows Platform SDK to enable some Apache features. For command line builds, the Platform SDK environment is prepared by the setenv batch file:

```
"c:\Program Files\Platform SDK\setenv.bat"
```

The Platform SDK files distributed with Visual C++ 6.0 and later are sufficient, so users of later version may skip this requirement.

Note that the Windows Platform SDK update is required to enable all supported [mod\\_isapi](#) features. Without a recent update, Apache will issue warnings under MSVC++ 5.0 that some [mod\\_isapi](#) features will be disabled. Look for the update at <http://msdn.microsoft.com/downloads/sdks/platform/platform.>

- The awk utility (awk, gawk or similar).

To install Apache within the build system, several files are modified using the awk . exe utility. awk was chosen since it is a very small download (compared with Perl or WSH/VB) and accomplishes the task of generating files. Brian Kernighan's <http://cm.bell-labs.com/cm/cs/who/bwk/> site has a compiled native Win32 binary, <http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe> which you must save with the name awk . exe rather than awk95 . exe.

Note that Developer Studio IDE will only find awk . exe from the Tools menu Options... Directories tab (the Projects - VC++ Directories pane in Developer Studio 7.0) listing Executable file paths. Add the path for awk . exe to this list, and your system PATH environment variable, as needed.

Also note that if you are using Cygwin (<http://www.cygwin.com/>) the awk utility is named gawk . exe and that the file awk . exe is really a symlink to the gawk . exe file. The Windows command shell does not recognize symlinks, and because of that building InstallBin will fail. A workaround is to delete awk . exe from the cygwin installation and rename gawk . exe to awk . exe.

- [Optional] OpenSSL libraries (for [mod\\_ssl](#) and `ab.exe` with ssl support)

**Caution: there are significant restrictions and prohibitions on the use and distribution of strong cryptography and patented intellectual property throughout the world.** OpenSSL includes strong cryptography controlled by both export regulations and domestic law, as well as intellectual property protected by patent, in the United States and elsewhere. Neither the Apache Software Foundation nor the OpenSSL project can provide legal advice regarding possession, use, or distribution of the code provided by the OpenSSL project. **Consult your own legal counsel, you are responsible for your own actions.**

OpenSSL must be installed into a `src\lib` subdirectory named `openssl`, obtained from <http://www.openssl.org/source/>, in order to compile [mod\\_ssl](#) or the `abs` project (`ab.exe` with SSL support.) To prepare OpenSSL for both `release` and `debug` builds of Apache, and disable the patent protected features in OpenSSL, you might use the following build commands:

```
perl Configure VC-WIN32
perl util\mkfiles.pl >MINFO
perl util\mk1mf.pl dll no-asm no-mdc2 no-rc5 no-idea VC-
WIN32 >makefile.rel
perl util\mk1mf.pl dll debug no-asm no-mdc2 no-rc5 no-idea
VC-WIN32 >makefile.dbg
perl util\mkdef.pl 32 libeay no-asm no-mdc2 no-rc5 no-idea
>ms\libeay32.def
perl util\mkdef.pl 32 ssleay no-asm no-mdc2 no-rc5 no-idea
>ms\ssleay32.def
nmake -f makefile.rel
nmake -f makefile.dbg
```

Note; you can use the scripts in the `ms\` subdirectory, however, it's rather tricky to force `ms\do_masm.bat`, for example, to perform the patent encumbrances as mentioned above. Patches to add the `$*` argument list to the appropriate `.bat` lines in these scripts aren't incorporated, thus far.

- [Optional] zlib sources (for `mod_deflate`)

Zlib must be installed into a `src\lib` subdirectory named `zlib`, however those sources need not be compiled. The build system will compile the compression sources directly into the `mod_deflate` module. Zlib can be obtained from <http://www.zlib.net/> -- `mod_deflate` is confirmed to build correctly with version 1.1.4. To use a later version of zlib, upgrade to Apache HTTP Server release 2.2 or later.



First, unpack the Apache distribution into an appropriate directory. Open a command-line prompt and `cd` to that directory.

The master Apache makefile instructions are contained in the `Makefile.win` file. To compile Apache on Windows NT, simply use one of the following commands to compile the release or debug build, respectively:

```
nmake /f Makefile.win _apacher  
nmake /f Makefile.win _apached
```

Either command will compile Apache. The latter will include debugging information in the resulting files, making it easier to find bugs and track down problems.



---

Apache can also be compiled using VC++'s Visual Studio development environment. To simplify this process, a Visual Studio workspace, `Apache.dsw`, is provided. This workspace exposes the entire list of working `.dsp` projects that are required for the complete Apache binary release. It includes dependencies between the projects to assure that they are built in the appropriate order.

Open the `Apache.dsw` workspace, and select `InstallBin` (Release or Debug build, as desired) as the Active Project. `InstallBin` causes all related project to be built, and then invokes `Makefile.win` to move the compiled executables and dlls. You may personalize the `INSTDIR=` choice by changing `InstallBin`'s Settings, General tab, Build command line entry. `INSTDIR` defaults to the `/Apache2` directory. If you only want a test compile (without installing) you may build the `BuildBin` project instead.

The `.dsp` project files are distributed in Visual C++ 6.0 format. Visual C++ 5.0 (97) will recognize them. Visual C++ 7.0 (.net) must convert `Apache.dsw` plus the `.dsp` files into an `Apache.sln` plus `.msproj` files, be sure you reconvert the `.msproj` file if any of the source `.dsp` files change! This is really trivial, just open `Apache.dsw` in the VC++ 7.0 IDE once again.

Visual C++ 7.0 (.net) users should also use the Build menu, Configuration Manager dialog to uncheck both the Debug and Release Solution modules `abs`, `mod_ssl` and `mod_deflate`. These modules are built by invoking `nmake` or the IDE directly with the `BinBuild` target to build those modules explicitly, only if the `src\lib` directories `openssl` and/or `zlib` exist.

Exported `.mak` files pose a greater hassle, but they are required

for Visual C++ 5.0 users to build `mod_ssl`, `abs` (`ab` with SSL support) and/or `mod_deflate`. VC++ 7.0 (.net) users also benefit, `nmake` builds are faster than `binenv` builds. Build the entire project from within the VC++ 5.0 or 6.0 IDE, then use the Project Menu Export for all makefiles. You must build the projects first in order to create all dynamic auto-generated targets, so that dependencies can be parsed correctly. Run the following command to fix the paths so they will build anywhere:

```
perl src\lib\apr\build\fixwin32mak.pl
```

You must type this command from the *top level* directory of the `httpd` source tree. Every `.mak` and `.dep` project file within the current directory and below will be corrected, and the timestamps adjusted to reflect the `.dsp`.

If you contribute back a patch that revises project files, we must commit project files in Visual Studio 6.0 format. Changes should be simple, with minimal compilation and linkage flags that will be recognized by all VC++ 5.0 through 7.0 environments.



## Project Components

The Apache .dsw workspace and makefile.win nmake script both build the .dsp projects of the Apache server in the following sequence:

1. srclib\apr\apr.dsp
2. srclib\apr\libapr.dsp
3. srclib\apr-util\uri\gen\_uri\_delims.dsp
4. srclib\apr-util\xml\expat\lib\xml.dsp
5. srclib\apr-util\aprutil.dsp
6. srclib\apr-util\libaprutil.dsp
7. srclib\pcre\dftables.dsp
8. srclib\pcre\pcre.dsp
9. srclib\pcre\pcreposix.dsp
10. server\gen\_test\_char.dsp
11. libhttpd.dsp
12. Apache.dsp

In addition, the modules\ subdirectory tree contains project files for the majority of the modules.

The support\ directory contains project files for additional programs that are not part of the Apache runtime, but are used by the administrator to test Apache and maintain password and log files. Windows-specific support projects are broken out in the support\win32\ directory.

1. support\ab.dsp
2. support\htdigest.dsp

3. support\htpasswd.dsp
4. support\logresolve.dsp
5. support\rotatelogs.dsp
6. support\win32\ApacheMonitor.dsp
7. support\win32\wintty.dsp

Once Apache has been compiled, it needs to be installed in its server root directory. The default is the \Apache2 directory, of the same drive.

To build and install all the files into the desired folder *dir* automatically, use one of the following nmake commands:

```
nmake /f Makefile.win installr INSTDIR=dir
nmake /f Makefile.win installd INSTDIR=dir
```

The *dir* argument to INSTDIR gives the installation directory; it can be omitted if Apache is to be installed into \Apache2.

This will install the following:

- *dir*\bin\Apache.exe - Apache executable
- *dir*\bin\ApacheMonitor.exe - Service monitor taskbar icon utility
- *dir*\bin\htdigest.exe - Digest auth password file utility
- *dir*\bin\htdbm.exe - SDBM auth database password file utility
- *dir*\bin\htpasswd.exe - Basic auth password file utility
- *dir*\bin\logresolve.exe - Log file dns name lookup utility
- *dir*\bin\rotatelogs.exe - Log file cycling utility

- *dir\bin\wintty.exe* - Console window utility
- *dir\bin\libapr.dll* - Apache Portable Runtime shared library
- *dir\bin\libaprutil.dll* - Apache Utility Runtime shared library
- *dir\bin\libhttpd.dll* - Apache Core library
- *dir\modules\mod\_\*.so* - Loadable Apache modules
- *dir\conf* - Configuration directory
- *dir\logs* - Empty logging directory
- *dir\include* - C language header files
- *dir\lib* - Link library files

## **Warning about building Apache from the development tree**

Note only the .dsp files are maintained between release builds. The .mak files are NOT regenerated, due to the tremendous waste of reviewer's time. Therefore, you cannot rely on the NMAKE commands above to build revised .dsp project files unless you then export all .mak files yourself from the project. This is unnecessary if you build from within the Microsoft Developer Studio environment.

Also note it is very worthwhile to build the BuildBin target project (or the command line \_apacher or \_apached target) prior to exporting the make files. Many files are autogenerated in the build process. Only a full build provides all of the dependent files required to build proper dependency trees for correct build behavior.

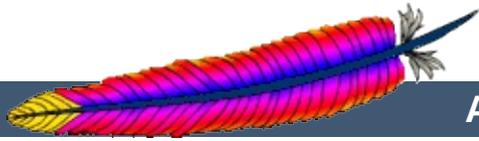
In order to create distribution .mak files, always review the generated .mak (or .dep) dependencies for Platform SDK or other garbage includes. The DevStudio\SharedIDE\bin\

(VC5) or DevStudio\Common\MSDev98\bin\ (VC6) directory contains the `sysincl.dat` file, which must list all exceptions. Update this file (including both forward and backslashed paths, such as both `sys/time.h` and `sys\time.h`) to include such dependencies. Including local-install paths in a distributed `.mak` file will cause the build to fail completely. And don't forget to run `src\lib\apr\build\fixwin32mak.pl` in order to fix absolute paths within the `.mak` files.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

## Using Apache With Novell NetWare

This document explains how to install, configure and run Apache 2.0 under Novell NetWare 6.0 and above. If you find any bugs, or wish to contribute in other ways, please use our [bug reporting page](#).

The bug reporting page and dev-httpd mailing list are *not* provided to answer questions about configuration or running Apache. Before you submit a bug report or request, first consult this document, the [Frequently Asked Questions](#) page and the other relevant documentation topics. If you still have a question or problem, post it to the [novell.devsup.webserver](#) newsgroup, where many Apache users are more than willing to answer new and obscure questions about using Apache on NetWare.

Most of this document assumes that you are installing Apache from a binary distribution. If you want to compile Apache yourself (possibly to help with development, or to track down bugs), see the section on [Compiling Apache for NetWare](#) below.



## Requirements

Apache 2.0 is designed to run on NetWare 6.0 service pack 3 and above. If you are running a service pack less than SP3, you must install the latest [NetWare Libraries for C \(LibC\)](#).

NetWare service packs are available [here](#).

Apache 2.0 for NetWare can also be run in a NetWare 5.1 environment as long as the latest service pack or the latest version of the [NetWare Libraries for C \(LibC\)](#) has been installed .

**WARNING:** Apache 2.0 for NetWare has not been targeted for or tested in this environment.



## Connecting Apache to NetWare

Information on the latest version of Apache can be found on the Apache web server at <http://www.apache.org/>. This will list the current release, any more recent alpha or beta-test releases, together with details of mirror web and anonymous ftp sites. Binary builds of the latest releases of Apache 2.0 for NetWare can be downloaded from [here](#).



## Installing Apache on NetWare

There is no Apache install program for NetWare currently. If you are building Apache 2.0 for NetWare from source, you will need to copy the files over to the server manually.

Follow these steps to install Apache on NetWare from the binary download (assuming you will install to `sys:/apache2`):

- Unzip the binary download file to the root of the SYS: volume (may be installed to any volume)
- Edit the `httpd.conf` file setting `ServerRoot` and `ServerName` along with any file path values to reflect your correct server settings
- Add `SYS:/APACHE2` to the search path, for example:

```
SEARCH ADD SYS:\APACHE2
```

Follow these steps to install Apache on NetWare manually from your own build source (assuming you will install to `sys:/apache2`):

- Create a directory called Apache2 on a NetWare volume
- Copy `APACHE2.NLM`, `APRLIB.NLM` to `SYS:/APACHE2`
- Create a directory under `SYS:/APACHE2` called `BIN`
- Copy `HTDIGEST.NLM`, `HTPASSWD.NLM`, `HTDBM.NLM`, `LOGRES.NLM`, `ROTLOGS.NLM` to `SYS:/APACHE2/BIN`
- Create a directory under `SYS:/APACHE2` called `CONF`
- Copy the `HTTPD-STD.CONF` file to the `SYS:/APACHE2/CONF` directory and rename to `HTTPD.CONF`
- Copy the `MIME.TYPES`, `CHARSET.CONV` and `MAGIC` files to `SYS:/APACHE2/CONF` directory
- Copy all files and subdirectories in `\HTTPD-2.0\DOCS\ICONS` to `SYS:/APACHE2/ICONS`

- Copy all files and subdirectories in \HTTPD-2.0\DOCS\MANUAL to SYS:/APACHE2/MANUAL
- Copy all files and subdirectories in \HTTPD-2.0\DOCS\ERROR to SYS:/APACHE2/ERROR
- Copy all files and subdirectories in \HTTPD-2.0\DOCS\DOCR00T to SYS:/APACHE2/HTDOCS
- Create the directory SYS:/APACHE2/LOGS on the server
- Create the directory SYS:/APACHE2/CGI-BIN on the server
- Create the directory SYS:/APACHE2/MODULES and copy all nlm modules into the modules directory
- Edit the HTTPD.CONF file searching for all @@Value@@ markers and replacing them with the appropriate setting
- Add SYS:/APACHE2 to the search path, for example:

```
SEARCH ADD SYS:\APACHE2
```

Apache may be installed to other volumes besides the default SYS volume.

During the build process, adding the keyword "install" to the makefile command line will automatically produce a complete distribution package under the subdirectory DIST. Install Apache by simply copying the distribution that was produced by the makfiles to the root of a NetWare volume (see: [Compiling Apache for NetWare](#) below).



## Running Apache on NetWare

To start Apache just type `apache` at the console. This will load apache in the OS address space. If you prefer to load Apache in a protected address space you may specify the address space with the load statement as follows:

```
load address space = apache2 apache2
```

This will load Apache into an address space called `apache2`. Running multiple instances of Apache concurrently on NetWare is possible by loading each instance into its own protected address space.

After starting Apache, it will be listening to port 80 (unless you changed the `Listen` directive in the configuration files). To connect to the server and access the default page, launch a browser and enter the server's name or address. This should respond with a welcome page, and a link to the Apache manual. If nothing happens or you get an error, look in the `error_log` file in the `logs` directory.

Once your basic installation is working, you should configure it properly by editing the files in the `conf` directory.

To unload Apache running in the OS address space just type the following at the console:

```
unload apache2
```

or

```
apache2 shutdown
```

If apache is running in a protected address space specify the address space in the unload statement:

```
unload address space = apache2 apache2
```

When working with Apache it is important to know how it will find the configuration files. You can specify a configuration file on the command line in two ways:

- `-f` specifies a path to a particular configuration file

```
apache2 -f "vol:/my server/conf/my.conf"
```

```
apache -f test/test.conf
```

In these cases, the proper [ServerRoot](#) should be set in the configuration file.

If you don't specify a configuration file name with `-f`, Apache will use the file name compiled into the server, usually `conf/httpd.conf`. Invoking Apache with the `-V` switch will display this value labeled as `SERVER_CONFIG_FILE`. Apache will then determine its [ServerRoot](#) by trying the following, in this order:

- A `ServerRoot` directive via a `-C` switch.
- The `-d` switch on the command line.
- Current working directory
- The server root compiled into the server.

The server root compiled into the server is usually `sys:/apache2`. Invoking `apache` with the `-V` switch will display this value labeled as `HTTPD_ROOT`.

Apache 2.0 for NetWare includes a set of command line directives that can be used to modify or display information about the running instance of the web server. These directives are only

available while Apache is running. Each of these directives must be preceded by the keyword APACHE2.

## **RESTART**

Instructs Apache to terminate all running worker threads as they become idle, reread the configuration file and restart each worker thread based on the new configuration.

## **VERSION**

Displays version information about the currently running instance of Apache.

## **MODULES**

Displays a list of loaded modules both built-in and external.

## **DIRECTIVES**

Displays a list of all available directives.

## **SETTINGS**

Enables or disables the thread status display on the console. When enabled, the state of each running threads is displayed on the Apache console screen.

## **SHUTDOWN**

Terminates the running instance of the Apache web server.

## **HELP**

Describes each of the runtime directives.

By default these directives are issued against the instance of Apache running in the OS address space. To issue a directive against a specific instance running in a protected address space, include the -p parameter along with the name of the address space. For more information type "apache2 Help" on the command line.



## Configuring Apache for NetWare

Apache is configured by reading configuration files usually stored in the conf directory. These are the same as files used to configure the Unix version, but there are a few different directives for Apache on NetWare. See the [Apache documentation](#) for all the available directives.

The main differences in Apache for NetWare are:

- Because Apache for NetWare is multithreaded, it does not use a separate process for each request, as Apache does on some Unix implementations. Instead there are only threads running: a parent thread, and multiple child or worker threads which handle the requests.

Therefore the "process"-management directives are different:

[MaxRequestsPerChild](#) - Like the Unix directive, this controls how many requests a worker thread will serve before exiting. The recommended default, `MaxRequestsPerChild 0`, causes the thread to continue servicing request indefinitely. It is recommended on NetWare, unless there is some specific reason, that this directive always remain set to 0.

[StartThreads](#) - This directive tells the server how many threads it should start initially. The recommended default is `StartThreads 50`.

[MinSpareThreads](#) - This directive instructs the server to spawn additional worker threads if the number of idle threads ever falls below this value. The recommended default is `MinSpareThreads 10`.

[MaxSpareThreads](#) - This directive instructs the server to begin terminating worker threads if the number of idle threads

ever exceeds this value. The recommended default is `MaxSpareThreads 100`.

**`MaxThreads`** - This directive limits the total number of work threads to a maximum value. The recommended default is `ThreadsPerChild 250`.

**`ThreadStackSize`** - This directive tells the server what size of stack to use for the individual worker thread. The recommended default is `ThreadStackSize 65536`.

- The directives that accept filenames as arguments must use NetWare filenames instead of Unix names. However, because Apache uses Unix-style names internally, forward slashes must be used rather than backslashes. It is recommended that all rooted file paths begin with a volume name. If omitted, Apache will assume the `SYS:` volume which may not be correct.
- Apache for NetWare has the ability to load modules at runtime, without recompiling the server. If Apache is compiled normally, it will install a number of optional modules in the `\Apache2\modules` directory. To activate these, or other modules, the **`LoadModule`** directive must be used. For example, to activate the status module, use the following:

```
LoadModule status_module modules/status.nlm
```

Information on [creating loadable modules](#) is also available.

### **Additional NetWare specific directives:**

- **`CGIMapExtension`** - This directive maps a CGI file extension to a script interpreter.

- [SecureListen](#) - Enables SSL encryption for a specified port.
- [NWSSLTrustedCerts](#) - Adds trusted certificates that are used to create secure connections to proxied servers.
- [NWSSLUpgradeable](#) - Allow a connection created on the specified address/port to be upgraded to an SSL connection.



## Compiling Apache for NetWare

Compiling Apache requires MetroWerks CodeWarrior 6.x or higher. Once Apache has been built, it can be installed to the root of any NetWare volume. The default is the sys : /Apache2 directory.

Before running the server you must fill out the conf directory. Copy the file HTTPD-STD.CONF from the distribution conf directory and rename it to HTTPD.CONF. Edit the HTTPD.CONF file searching for all @@Value@@ markers and replacing them with the appropriate setting. Copy over the conf/magic and conf/mime.types files as well. Alternatively, a complete distribution can be built by including the keyword install when invoking the makefiles.

### Requirements:

The following development tools are required to build Apache 2.0 for NetWare:

- Metrowerks CodeWarrior 6.0 or higher with the [NetWare PDK 3.0](#) or higher.
- [NetWare Libraries for C \(LibC\)](#)
- [LDAP Libraries for C](#)
- [ZLIB Compression Library source code](#)
- AWK utility (awk, gawk or similar). AWK can be downloaded from <http://developer.novell.com/ndk/apache.htm>. The utility must be found in your windows path and must be named awk.exe.
- To build using the makefiles, you will need GNU make version 3.78.1 (GMake) available at <http://developer.novell.com/ndk/apache.htm>.

### Building Apache using the NetWare makefiles:

- Set the environment variable NOVELLIBC to the location of the NetWare Libraries for C SDK, for example:

```
Set NOVELLIBC=c:\novell\ndk\libc
```

- Set the environment variable METROWERKS to the location where you installed the Metrowerks CodeWarrior compiler, for example:

```
Set METROWERKS=C:\Program Files\Metrowerks\CodeWarrior
```

If you installed to the default location C:\Program Files\Metrowerks\CodeWarrior, you don't need to set this.

- Set the environment variable LDAPSDK to the location where you installed the LDAP Libraries for C, for example:

```
Set LDAPSDK=c:\Novell\NDK\cldap sdk\NetWare\libc
```

- Set the environment variable ZLIBSDK to the location where you installed the source code for the ZLib Library, for example:

```
Set ZLIBSDK=D:\NOVELL\zlib
```

- Set the environment variable AP\_WORK to the full path of the \httpd-2.0 directory.
- Set the environment variable APR\_WORK to the full path of the \httpd-2.0\src\lib\apr directory.
- Make sure that the path to the AWK utility and the GNU make utility (gmake.exe) have been included in the system's PATH environment variable.
- Download the source code and unzip to an appropriate

directory on your workstation.

- Change directory to `\httpd-2.0\src\lib\apr-util\uri` and build `GENURI.nlm` by running `"gmake -f nwgnumakefile"`.
- Copy the file `GENURI.nlm` to the `SYS:` volume of a NetWare server and run using the following command:

```
SYS:\genuri > sys:\uri_delims.h
```

- Copy the file `uri_delims.h` to the directory `\httpd-2.0\src\lib\apr-util\uri` on the build machine.
- Change directory to `\httpd-2.0\src\lib\apr` and build `APR` by running `"gmake -f nwgnumakefile"`
- Change directory to `\httpd-2.0\src\lib\pcre` and build `DFTABLES.nlm` by running `"gmake -f nwgnumakefile"`
- Change directory to `\httpd-2.0\server` and build `GENCHARS.nlm` by running `"gmake -f nwgnumakefile"`
- Copy the files `GENCHARS.nlm` and `DFTABLES.nlm` from their respective directories to the `SYS:` volume of a NetWare server and run them using the following commands:

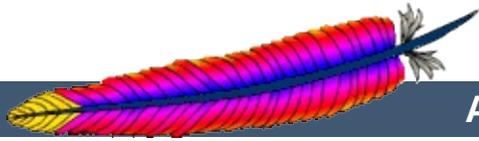
```
SYS:\genchars > sys:\test_char.h  
SYS:\dftables > sys:\chartables.c
```

- Copy the files `test_char.h` and `chartables.c` to the directory `\httpd-2.0\os\netware` on the build machine.
- Change directory to `\httpd-2.0` and build Apache by running `"gmake -f nwgnumakefile"`. You can create a distribution directory by adding an `install` parameter to the command, for example:

```
gmake -f nwgnumakefile install
```

## Additional make options

- `gmake -f nwgnmakefile`  
Builds release versions of all of the binaries and copies them to a `\release` destination directory.
- `gmake -f nwgnmakefile DEBUG=1`  
Builds debug versions of all of the binaries and copies them to a `\debug` destination directory.
- `gmake -f nwgnmakefile install`  
Creates a complete Apache distribution with binaries, docs and additional support files in a `\dist\Apache2` directory.
- `gmake -f nwgnmakefile installdev`  
Same as `install` but also creates a `\lib` and `\include` directory in the destination directory and copies headers and import files.
- `gmake -f nwgnmakefile clean`  
Cleans all object files and binaries from the `\release` or `\debug` build areas depending on whether `DEBUG` has been defined.
- `gmake -f nwgnmakefile clobber_all`  
Same as `clean` and also deletes the distribution directory if it exists.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

## Running a High-Performance Web Server on HPUX

Date: Wed, 05 Nov 1997 16:59:34 -0800  
From: Rick Jones <[raj@cup.hp.com](mailto:raj@cup.hp.com)>  
Reply-To: [raj@cup.hp.com](mailto:raj@cup.hp.com)  
Organization: Network Performance  
Subject: HP-UX tuning tips

Here are some tuning tips for HP-UX to add to the tuning page.

For HP-UX 9.X: Upgrade to 10.20

For HP-UX 10.[00|01|10]: Upgrade to 10.20

For HP-UX 10.20:

Install the latest cumulative ARPA Transport Patch. This will allow you to configure the size of the TCP connection lookup hash table. The default is 256 buckets and must be set to a power of two. This is accomplished with adb against the \*disc\* image of the kernel. The variable name is `tcp_hash_size`. Notice that it's critically important that you use "w" to write a 32 bit quantity, not "w" to write a 16 bit value when patching the disc image because the `tcp_hash_size` variable is a 32 bit quantity.

How to pick the value? Examine the output of <ftp://ftp.cup.hp.com/dist/networking/tools/connhist> and see how many total TCP connections exist on the system. You probably want that number divided by the hash table size to be reasonably small, say less than 10. Folks can look at HP's SPECweb96 disclosures for some common settings. These can be found at <http://www.specbench.org/>. If an HP-UX system was performing at 1000 SPECweb96 connections per second, the TIME\_WAIT time of 60 seconds would mean 60,000 TCP "connections" being tracked.

Folks can check their listen queue depths with <ftp://ftp.cup.hp.com/dist/networking/misc/listenq>.

If folks are running Apache on a PA-8000 based system, they should consider "chat'ring" the Apache executable to have a large page size. This would be "chatr +pi L <BINARY>". The GID of the running executable must have MLOCK privileges. Setprivgrp(1m) should be consulted for assigning MLOCK. The change can be validated by running Glance and examining the memory regions of the server(s) to make sure that they show a non-trivial fraction of the text segment being locked.

If folks are running Apache on MP systems, they might consider writing a small program that uses mpctl( ) to bind processes to processors. A simple pid % numcpu algorithm is probably sufficient. This might even go into the source code.

If folks are concerned about the number of FIN\_WAIT\_2 connections, they can use nettune to shrink the value of tcp\_keepstart. However, they should be careful there - certainly do not make it less than oh two to four minutes. If tcp\_hash\_size has been set well, it is probably OK to let the FIN\_WAIT\_2's take longer to timeout (perhaps even the default two hours) - they will not on average have a big impact on performance.

There are other things that could go into the code base, but that might be left for another email. Feel free to drop me a message if you or others are interested.

sincerely,

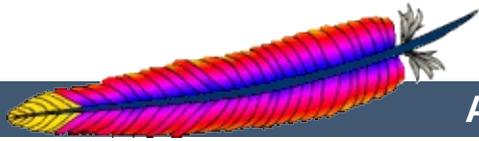
rick jones

<http://www.cup.hp.com/netperf/NetperfPage.html>

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Platform Specific Notes](#)

## The Apache EBCDIC Port

**Warning:** This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.



Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

(It is the SIEMENS family of mainframes running the [BS2000/OSD operating system](#). This mainframe OS nowadays features a SVR4-derived POSIX subsystem).

The port was started initially to

- prove the feasibility of porting [the Apache HTTP server](#) to this platform
- find a "worthy and capable" successor for the venerable [CERN-3.0](#) daemon (which was ported a couple of years ago), and to
- prove that Apache's preforking process model can on this platform easily outperform the accept-fork-serve model used by CERN by a factor of 5 or more.

This document serves as a rationale to describe some of the design decisions of the port to this machine.



One objective of the EBCDIC port was to maintain enough backwards compatibility with the (EBCDIC) CERN server to make the transition to the new server attractive and easy. This required the addition of a configurable method to define whether a HTML document was stored in ASCII (the only format accepted by the old server) or in EBCDIC (the native document format in the POSIX subsystem, and therefore the only realistic format in which the other POSIX tools like grep or sed could operate on the documents). The current solution to this is a "pseudo-MIME-format" which is intercepted and interpreted by the Apache server (see below). Future versions might solve the problem by defining an "ebcdic-handler" for all documents which must be converted.



Since all Apache input and output is based upon the BUFF data type and its methods, the easiest solution was to add the conversion to the BUFF handling routines. The conversion must be settable at any time, so a BUFF flag was added which defines whether a BUFF object has currently enabled conversion or not. This flag is modified at several points in the HTTP protocol:

- **set** before a request is received (because the request and the request header lines are always in ASCII format)
- **set/unset** when the request body is received - depending on the content type of the request body (because the request body may contain ASCII text or a binary file)
- **set** before a reply header is sent (because the response header lines are always in ASCII format)
- **set/unset** when the response body is sent - depending on the content type of the response body (because the response body may contain text or a binary file)



1. The relevant changes in the source are `#ifdef`'ed into two categories:

**`#ifdef CHARSET_EBCDIC`**

Code which is needed for any EBCDIC based machine. This includes character translations, differences in contiguity of the two character sets, flags which indicate which part of the HTTP protocol has to be converted and which part doesn't *etc.*

**`#ifdef _OSD_POSIX`**

Code which is needed for the SIEMENS BS2000/OSD mainframe platform only. This deals with include file differences and socket implementation topics which are only required on the BS2000/OSD platform.

2. The possibility to translate between ASCII and EBCDIC at the socket level (on BS2000 POSIX, there is a socket option which supports this) was intentionally *not* chosen, because the byte stream at the HTTP protocol level consists of a mixture of protocol related strings and non-protocol related raw file data. HTTP protocol strings are always encoded in ASCII (the GET request, any Header: lines, the chunking information *etc.*) whereas the file transfer parts (*i.e.*, GIF images, CGI output *etc.*) should usually be just "passed through" by the server. This separation between "protocol string" and "raw data" is reflected in the server code by functions like `bgets()` or `rvputs()` for strings, and functions like `bwrite()` for binary data. A global translation of everything would therefore be inadequate.

(In the case of text files of course, provisions must be made so that EBCDIC documents are always served in ASCII)

3. This port therefore features a built-in protocol level conversion for the server-internal strings (which the compiler translated to EBCDIC strings) and thus for all server-generated documents. The hard coded ASCII escapes `\012` and `\015` which are ubiquitous in the server code are an exception: they are already the binary encoding of the ASCII `\n` and `\r` and must not be converted to ASCII a second time. This exception is only relevant for server-generated strings; and *external* EBCDIC documents are not expected to contain ASCII newline characters.
4. By examining the call hierarchy for the BUFF management routines, I added an "ebcdic/ascii conversion layer" which would be crossed on every puts/write/get/gets, and a conversion flag which allowed enabling/disabling the conversions on-the-fly. Usually, a document crosses this layer twice from its origin source (a file or CGI output) to its destination (the requesting client): `file -> Apache`, and `Apache -> client`.

The server can now read the header lines of a CGI-script output in EBCDIC format, and then find out that the remainder of the script's output is in ASCII (like in the case of the output of a WWW Counter program: the document body contains a GIF image). All header processing is done in the native EBCDIC format; the server then determines, based on the type of document being served, whether the document body (except for the chunking information, of course) is in ASCII already or must be converted from EBCDIC.

5. For Text documents (MIME types `text/plain`, `text/html` etc.), an implicit translation to ASCII can be used, or (if the users prefer to store some documents in raw ASCII form for faster serving, or because the files reside on a NFS-mounted directory tree) can be served without conversion.

## Example:

to serve files with the suffix `.html` as a raw ASCII `text/html` document without implicit conversion (and suffix `.ascii` as ASCII `text/plain`), use the directives:

```
AddType text/x-ascii-html .html
AddType text/x-ascii-plain .ascii
```

Similarly, any `text/foo` MIME type can be served as "raw ASCII" by configuring a MIME type "`text/x-ascii-foo`" for it using `AddType`.

6. Non-text documents are always served "binary" without conversion. This seems to be the most sensible choice for, *e.g.*, GIF/ZIP/AU file types. This of course requires the user to copy them to the mainframe host using the "`rcp -b`" binary switch.
7. Server parsed files are always assumed to be in native (*i.e.*, EBCDIC) format as used on the machine, and are converted after processing.
8. For CGI output, the CGI script determines whether a conversion is needed or not: by setting the appropriate Content-Type, text files can be converted, or GIF output can be passed through unmodified. An example for the latter case is the `wwwcount` program which we ported as well.



## Binary Files

All files with a Content-Type: which does not start with text/ are regarded as *binary files* by the server and are not subject to any conversion. Examples for binary files are GIF images, gzip-compressed files and the like.

When exchanging binary files between the mainframe host and a Unix machine or Windows PC, be sure to use the ftp "binary" (TYPE I) command, or use the rcp -b command from the mainframe host (the -b switch is not supported in unix rcp's).

## Text Documents

The default assumption of the server is that Text Files (*i.e.*, all files whose Content-Type: starts with text/) are stored in the native character set of the host, EBCDIC.

## Server Side Included Documents

SSI documents must currently be stored in EBCDIC only. No provision is made to convert it from ASCII before processing.



Module	Status	Notes
<a href="#">core</a>	+	
<a href="#">mod_access</a>	+	
<a href="#">mod_actions</a>	+	
<a href="#">mod_alias</a>	+	
<a href="#">mod_asis</a>	+	
<a href="#">mod_auth</a>	+	
<a href="#">mod_auth_anon</a>	+	
<a href="#">mod_auth_dbm</a>	?	with own libdb.a
<a href="#">mod_autoindex</a>	+	
<a href="#">mod_cern_meta</a>	?	
<a href="#">mod_cgi</a>	+	
<a href="#">mod_digest</a>	+	
<a href="#">mod_dir</a>	+	
<a href="#">mod_so</a>	-	no shared libs
<a href="#">mod_env</a>	+	
<a href="#">mod_example</a>	-	(test bed only)
<a href="#">mod_expires</a>	+	
<a href="#">mod_headers</a>	+	
<a href="#">mod_imap</a>	+	
<a href="#">mod_include</a>	+	
<a href="#">mod_info</a>	+	
<a href="#">mod_log_agent</a>	+	
<a href="#">mod_log_config</a>	+	
<a href="#">mod_log_referer</a>	+	
<a href="#">mod_mime</a>	+	
<a href="#">mod_mime_magic</a>	?	not ported yet
<a href="#">mod_negotiation</a>	+	

<u>mod_proxy</u>	+	
<u>mod_rewrite</u>	+	untested
<u>mod_setenvif</u>	+	
<u>mod_speling</u>	+	
<u>mod_status</u>	+	
<u>mod_unique_id</u>	+	
<u>mod_userdir</u>	+	
<u>mod_usertrack</u>	?	untested



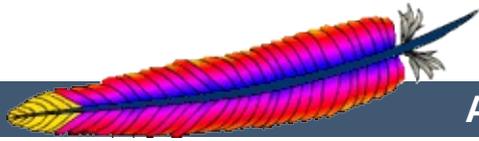
## Third Party Modules Status

Module	Status	Notes
<a href="#">mod_jserv</a>	-	JAVA still being ported.
<a href="#">mod_php3</a>	+	mod_php3 runs fine, with LDAP and GD and FreeType libraries.
<a href="#">mod_put</a>	?	untested
<a href="#">mod_session</a>	-	untested

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

# httpd - Apache Hypertext Transfer Protocol Server

httpd is the Apache HyperText Transfer Protocol (HTTP) server program. It is designed to be run as a standalone daemon process. When used like this it will create a pool of child processes or threads to handle requests.

In general, httpd should not be invoked directly, but rather should be invoked via [apachectl](#) on Unix-based systems or [as a service on Windows NT, 2000 and XP](#) and [as a console application on Windows 9x and ME](#).

## See also

[Starting Apache](#)

[Stopping Apache](#)

[Configuration Files](#)

[Platform-specific Documentation](#)

[apachectl](#)



**httpd** [ **-d** *serverroot* ] [ **-f** *config* ] [ **-C** *directive* ] [ **-c** *directive* ] [ **-D** *parameter* ] [ **-e** *level* ] [ **-E** *file* ] [ **-k** *start|restart|graceful|stop* ] [ **-R** *directory* ] [ **-h** ] [ **-l** ] [ **-L** ] [ **-S** ] [ **-t** ] [ **-v** ] [ **-V** ] [ **-X** ]

On [Windows systems](#), the following additional arguments are available:

**httpd** [ **-k** *install|config|uninstall* ] [ **-n** *name* ] [ **-w** ]



## **-d *serverroot***

Set the initial value for the [ServerRoot](#) directive to *serverroot*. This can be overridden by the `ServerRoot` directive in the configuration file. The default is `/usr/local/apache2`.

## **-f *config***

Uses the directives in the file *config* on startup. If *config* does not begin with a `/`, then it is taken to be a path relative to the [ServerRoot](#). The default is `conf/httpd.conf`.

## **-k *start|restart|graceful|stop***

Signals `httpd` to start, restart, or stop. See [Stopping Apache](#) for more information.

## **-C *directive***

Process the configuration *directive* before reading config files.

## **-c *directive***

Process the configuration *directive* after reading config files.

## **-D *parameter***

Sets a configuration *parameter* which can be used with [<IfDefine>](#) sections in the configuration files to conditionally skip or process commands at server startup and restart.

## **-e *level***

Sets the [LogLevel](#) to *level* during server startup. This is useful for temporarily increasing the verbosity of the error messages to find problems during startup.

## **-E *file***

Send error messages during server startup to *file*.

## **-R *directory***

When the server is compiled using the `SHARED_CORE` rule,

this specifies the *directory* for the shared object files.

**-h**

Output a short summary of available command line options.

**-l**

Output a list of modules compiled into the server. This will **not** list dynamically loaded modules included using the [LoadModule](#) directive.

**-L**

Output a list of directives together with expected arguments and places where the directive is valid.

**-S**

Show the settings as parsed from the config file (currently only shows the virtualhost settings).

**-t**

Run syntax tests for configuration files only. The program immediately exits after these syntax parsing tests with either a return code of 0 (Syntax OK) or return code not equal to 0 (Syntax Error). If `-D DUMP_VHOSTS` is also set, details of the virtual host configuration will be printed.

**-v**

Print the version of `httpd`, and then exit.

**-V**

Print the version and build parameters of `httpd`, and then exit.

**-X**

Run `httpd` in debug mode. Only one worker will be started and the server will not detach from the console.

The following arguments are available only on the [Windows platform](#):

**-k `install|config|uninstall`**

Install Apache as a Windows NT service; change startup options for the Apache service; and uninstall the Apache service.

**-n *name***

The *name* of the Apache service to signal.

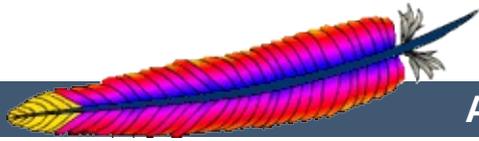
**-w**

Keep the console window open on error so that the error message can be read.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## ab - Apache HTTP server benchmarking tool

ab is a tool for benchmarking your Apache Hypertext Transfer Protocol (HTTP) server. It is designed to give you an impression of how your current Apache installation performs. This especially shows you how many requests per second your Apache installation is capable of serving.

### See also

[httpd](#)



**ab** [ **-A** *auth-username:password* ] [ **-c** *concurrency* ] [ **-C** *cookie-name=value* ] [ **-d** ] [ **-e** *csv-file* ] [ **-g** *gnuplot-file* ] [ **-h** ] [ **-H** *custom-header* ] [ **-i** ] [ **-k** ] [ **-n** *requests* ] [ **-p** *POST-file* ] [ **-P** *proxy-auth-username:password* ] [ **-q** ] [ **-s** ] [ **-S** ] [ **-t** *timelimit* ] [ **-T** *content-type* ] [ **-v** *verbosity* ] [ **-V** ] [ **-w** ] [ **-x** *<table>-attributes* ] [ **-X** *proxy[:port]* ] [ **-y** *<tr>-attributes* ] [ **-z** *<td>-attributes* ] [http://]*hostname[:port]/path*



**-A *auth-username:password***

Supply BASIC Authentication credentials to the server. The username and password are separated by a single `:` and sent on the wire base64 encoded. The string is sent regardless of whether the server needs it (*i.e.*, has sent an 401 authentication needed).

**-c *concurrency***

Number of multiple requests to perform at a time. Default is one request at a time.

**-C *cookie-name=value***

Add a `Cookie:` line to the request. The argument is typically in the form of a *name=value* pair. This field is repeatable.

**-d**

Do not display the "percentage served within XX [ms] table". (legacy support).

**-e *csv-file***

Write a Comma separated value (CSV) file which contains for each percentage (from 1% to 100%) the time (in milliseconds) it took to serve that percentage of the requests. This is usually more useful than the 'gnuplot' file; as the results are already 'binned'.

**-g *gnuplot-file***

Write all measured values out as a 'gnuplot' or TSV (Tab separate values) file. This file can easily be imported into packages like Gnuplot, IDL, Mathematica, Igor or even Excel. The labels are on the first line of the file.

**-h**

Display usage information.

**-H *custom-header***

Append extra headers to the request. The argument is

typically in the form of a valid header line, containing a colon-separated field-value pair (*i.e.*, "Accept-Encoding: zip/zop;8bit").

**-i**

Do HEAD requests instead of GET.

**-k**

Enable the HTTP KeepAlive feature, *i.e.*, perform multiple requests within one HTTP session. Default is no KeepAlive.

**-n *requests***

Number of requests to perform for the benchmarking session. The default is to just perform a single request which usually leads to non-representative benchmarking results.

**-p *POST-file***

File containing data to POST.

**-P *proxy-auth-username:password***

Supply BASIC Authentication credentials to a proxy en-route. The username and password are separated by a single : and sent on the wire base64 encoded. The string is sent regardless of whether the proxy needs it (*i.e.*, has sent an 407 proxy authentication needed).

**-q**

When processing more than 150 requests, ab outputs a progress count on `stderr` every 10% or 100 requests or so. The `-q` flag will suppress these messages.

**-s**

When compiled in (ab `-h` will show you) use the SSL protected `https` rather than the `http` protocol. This feature is experimental and very rudimentary. You probably do not want to use it.

**-S**

Do not display the median and standard deviation values, nor display the warning/error messages when the average and median are more than one or two times the standard deviation apart. And default to the min/avg/max values. (legacy support).

**-t *timelimit***

Maximum number of seconds to spend for benchmarking. This implies a -n 50000 internally. Use this to benchmark the server within a fixed total amount of time. Per default there is no timelimit.

**-T *content-type***

Content-type header to use for POST data.

**-v *verbosity***

Set verbosity level - 4 and above prints information on headers, 3 and above prints response codes (404, 200, etc.), 2 and above prints warnings and info.

**-V**

Display version number and exit.

**-w**

Print out results in HTML tables. Default table is two columns wide, with a white background.

**-x *<table>-attributes***

String to use as attributes for <table>. Attributes are inserted <table *here* >.

**-X *proxy[:port]***

Use a proxy server for the requests.

**-y *<tr>-attributes***

String to use as attributes for <tr>.

**-z *<td>-attributes***

String to use as attributes for <td>.



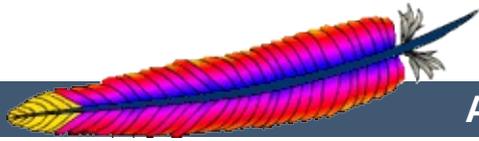
There are various statically declared buffers of fixed length. Combined with the lazy parsing of the command line arguments, the response headers from the server and other external inputs, this might bite you.

It does not implement HTTP/1.x fully; only accepts some 'expected' forms of responses. The rather heavy use of `strstr(3)` shows up top in profile, which might indicate a performance problem; *i.e.*, you would measure the ab performance rather than the server's.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

# apachectl - Apache HTTP Server Control Interface

`apachectl` is a front end to the Apache HyperText Transfer Protocol (HTTP) server. It is designed to help the administrator control the functioning of the Apache [httpd](#) daemon.

The `apachectl` script can operate in two modes. First, it can act as a simple front-end to the [httpd](#) command that simply sets any necessary environment variables and then invokes [httpd](#), passing through any command line arguments. Second, `apachectl` can act as a SysV init script, taking simple one-word arguments like `start`, `restart`, and `stop`, and translating them into appropriate signals to [httpd](#).

If your Apache installation uses non-standard paths, you will need to edit the `apachectl` script to set the appropriate paths to the [httpd](#) binary. You can also specify any necessary [httpd](#) command line arguments. See the comments in the script for details.

The `apachectl` script returns a 0 exit value on success, and >0 if an error occurs. For more details, view the comments in the script.

## See also

- [Starting Apache](#)
- [Stopping Apache](#)
- [Configuration Files](#)
- [Platform Docs](#)
- [httpd](#)



When acting in pass-through mode, `apachectl` can take all the arguments available for the [httpd](#) binary.

**`apachectl`** [ *httpd-argument* ]

When acting in SysV init mode, `apachectl` takes simple, one-word commands, defined below.

**`apachectl`** *command*



Only the SysV init-style options are defined here. Other arguments are defined on the [httpd](#) manual page.

### **start**

Start the Apache [httpd](#) daemon. Gives an error if it is already running. This is equivalent to `apachectl -k start`.

### **stop**

Stops the Apache [httpd](#) daemon. This is equivalent to `apachectl -k stop`.

### **restart**

Restarts the Apache [httpd](#) daemon. If the daemon is not running, it is started. This command automatically checks the configuration files as in `configtest` before initiating the restart to make sure the daemon doesn't die. This is equivalent to `apachectl -k restart`.

### **fullstatus**

Displays a full status report from [mod\\_status](#). For this to work, you need to have [mod\\_status](#) enabled on your server and a text-based browser such as `Lynx` available on your system. The URL used to access the status report can be set by editing the `STATUSURL` variable in the script.

### **status**

Displays a brief status report. Similar to the `fullstatus` option, except that the list of requests currently being served is omitted.

### **graceful**

Gracefully restarts the Apache [httpd](#) daemon. If the daemon is not running, it is started. This differs from a normal restart in that currently open connections are not aborted. A side effect

is that old log files will not be closed immediately. This means that if used in a log rotation script, a substantial delay may be necessary to ensure that the old log files are closed before processing them. This command automatically checks the configuration files as in `configtest` before initiating the restart to make sure Apache doesn't die. This is equivalent to `apachectl -k graceful`.

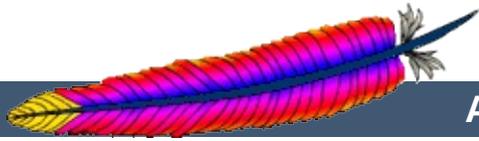
### **configtest**

Run a configuration file syntax test. It parses the configuration files and either reports `Syntax Ok` or detailed information about the particular syntax error. This is equivalent to `apachectl -t`.

The following additional option is available, but deprecated.

### **startssl**

This is equivalent to `apachectl -k start -DSSL`. We recommend that you use that command explicitly, or you adjust your `httpd.conf` to remove the `<IfDefine>` section so that SSL will always be available.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## apxs - APache eXtenSion tool

apxs is a tool for building and installing extension modules for the Apache HyperText Transfer Protocol (HTTP) server. This is achieved by building a dynamic shared object (DSO) from one or more source or object *files* which then can be loaded into the Apache server under runtime via the `LoadModule` directive from `mod_so`.

So to use this extension mechanism your platform has to support the DSO feature and your Apache `httpd` binary has to be built with the `mod_so` module. The apxs tool automatically complains if this is not the case. You can check this yourself by manually running the command

```
$ httpd -l
```

The module `mod_so` should be part of the displayed list. If these requirements are fulfilled you can easily extend your Apache server's functionality by installing your own modules with the DSO mechanism by the help of this apxs tool:

```
$ apxs -i -a -c mod_foo.c
gcc -fpic -DSHARED_MODULE -I/path/to/apache/include -c mod_foo.c
ld -Bshareable -o mod_foo.so mod_foo.o
cp mod_foo.so /path/to/apache/modules/mod_foo.so
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module `foo' in /path/to/apache/etc/httpd.conf]
$ apachectl restart
/path/to/apache/sbin/apachectl restart: httpd not running, trying
to start
[Tue Mar 31 11:27:55 1998] [debug] mod_so.c(303): loaded module
foo_module
/path/to/apache/sbin/apachectl restart: httpd started
$ _
```

The arguments *files* can be any C source file (.c), a object file (.o) or even a library archive (.a). The apxs tool automatically recognizes

these extensions and automatically used the C source files for compilation while just using the object and archive files for the linking phase. But when using such pre-compiled objects make sure they are compiled for position independent code (PIC) to be able to use them for a dynamically loaded shared object. For instance with GCC you always just have to use `-fpic`. For other C compilers consult its manual page or at watch for the flags `apxs` uses to compile the object files.

For more details about DSO support in Apache read the documentation of [mod\\_so](#) or perhaps even read the `src/modules/standard/mod_so.c` source file.

## See also

[apachectl](#)

[httpd](#)



**apxs -g** [ **-S** *name=value* ] **-n** *modname*

**apxs -q** [ **-S** *name=value* ] *query* ...

**apxs -c** [ **-S** *name=value* ] [ **-o** *dsofile* ] [ **-I** *incdir* ] [ **-D** *name=value* ] [ **-L** *libdir* ] [ **-l** *libname* ] [ **-Wc**,*compiler-flags* ] [ **-Wl**,*linker-flags* ] *files* ...

**apxs -i** [ **-S** *name=value* ] [ **-n** *modname* ] [ **-a** ] [ **-A** ] *dso-file* ...

**apxs -e** [ **-S** *name=value* ] [ **-n** *modname* ] [ **-a** ] [ **-A** ] *dso-file* ...



## Common Options

### **-n *modname***

This explicitly sets the module name for the `-i` (install) and `-g` (template generation) option. Use this to explicitly specify the module name. For option `-g` this is required, for option `-i` the `apxs` tool tries to determine the name from the source or (as a fallback) at least by guessing it from the filename.

## Query Options

### **-q**

Performs a query for `apxs`'s knowledge about certain settings. The *query* parameters can be one or more of the following strings: `CC`, `CFLAGS`, `CFLAGS_SHLIB`, `INCLUDEDIR`, `LD_SHLIB`, `LD_FLAGS_SHLIB`, `LIBEXECDIR`, `LIBS_SHLIB`, `SBINDIR`, `SYSCONFDIR`, `TARGET`. Use this for manually determining settings. For instance use

```
INC=-I`apxs -q INCLUDEDIR`
```

inside your own Makefiles if you need manual access to Apache's C header files.

## Configuration Options

### **-S *name=value***

This option changes the `apxs` settings described above.

## Template Generation Options

### **-g**

This generates a subdirectory *name* (see option `-n`) and there two files: A sample module source file named `mod_name.c`

which can be used as a template for creating your own modules or as a quick start for playing with the apxs mechanism. And a corresponding Makefile for even easier build and installing of this module.

## DSO Compilation Options

### **-c**

This indicates the compilation operation. It first compiles the C source files (.c) of *files* into corresponding object files (.o) and then builds a dynamically shared object in *dsofile* by linking these object files plus the remaining object files (.o and .a) of *files*. If no -o option is specified the output file is guessed from the first filename in *files* and thus usually defaults to `mod_name.so`.

### **-o *dsofile***

Explicitly specifies the filename of the created dynamically shared object. If not specified and the name cannot be guessed from the *files* list, the fallback name `mod_unknown.so` is used.

### **-D *name=value***

This option is directly passed through to the compilation command(s). Use this to add your own defines to the build process.

### **-I *incdir***

This option is directly passed through to the compilation command(s). Use this to add your own include directories to search to the build process.

### **-L *libdir***

This option is directly passed through to the linker command. Use this to add your own library directories to search to the build process.

**-l *libname***

This option is directly passed through to the linker command. Use this to add your own libraries to search to the build process.

**-Wc, *compiler-flags***

This option passes *compiler-flags* as additional flags to the compiler command. Use this to add local compiler-specific options.

**-Wl, *linker-flags***

This option passes *linker-flags* as additional flags to the linker command. Use this to add local linker-specific options.

## DSO Installation and Configuration Options

**-i**

This indicates the installation operation and installs one or more dynamically shared objects into the server's *modules* directory.

**-a**

This activates the module by automatically adding a corresponding [LoadModule](#) line to Apache's `httpd.conf` configuration file, or by enabling it if it already exists.

**-A**

Same as option -a but the created [LoadModule](#) directive is prefixed with a hash sign (`#`), *i.e.*, the module is just prepared for later activation but initially disabled.

**-e**

This indicates the editing operation, which can be used with the -a and -A options similarly to the -i operation to edit Apache's `httpd.conf` configuration file without attempting to install the module.



Assume you have an Apache module named `mod_foo.c` available which should extend Apache's server functionality. To accomplish this you first have to compile the C source into a shared object suitable for loading into the Apache server under runtime via the following command:

```
$ apxs -c mod_foo.c
gcc -fpic -DSHARED_MODULE -I/path/to/apache/include -c
mod_foo.c
ld -Bshareable -o mod_foo.so mod_foo.o
$ _
```

Then you have to update the Apache configuration by making sure a `LoadModule` directive is present to load this shared object. To simplify this step `apxs` provides an automatic way to install the shared object in its "modules" directory and updating the `httpd.conf` file accordingly. This can be achieved by running:

```
$ apxs -i -a mod_foo.c
cp mod_foo.so /path/to/apache/modules/mod_foo.so
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module `foo' in /path/to/apache/etc/httpd.conf]
$ _
```

This way a line named

```
LoadModule foo_module modules/mod_foo.so
```

is added to the configuration file if still not present. If you want to have this disabled per default use the `-A` option, *i.e.*

```
$ apxs -i -A mod_foo.c
```

For a quick test of the `apxs` mechanism you can create a sample Apache module template plus a corresponding Makefile via:

---

```
$ apxs -g -n foo
Creating [DIR] foo
Creating [FILE] foo/Makefile
Creating [FILE] foo/mod_foo.c
$ _
```

Then you can immediately compile this sample module into a shared object and load it into the Apache server:

```
$ cd foo
$ make all reload
apxs -c mod_foo.c
gcc -fpic -DSHARED_MODULE -I/path/to/apache/include -c
mod_foo.c
ld -Bshareable -o mod_foo.so mod_foo.o
apxs -i -a -n "foo" mod_foo.so
cp mod_foo.so /path/to/apache/modules/mod_foo.so
chmod 755 /path/to/apache/modules/mod_foo.so
[activating module `foo' in /path/to/apache/etc/httpd.conf]
apachectl restart
/path/to/apache/sbin/apachectl restart: httpd not running,
trying to start
[Tue Mar 31 11:27:55 1998] [debug] mod_so.c(303): loaded module
foo_module
/path/to/apache/sbin/apachectl restart: httpd started
$ _
```

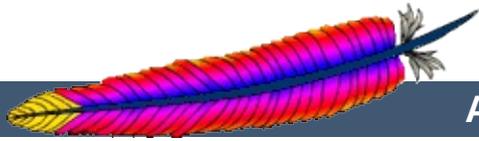
You can even use apxs to compile complex modules outside the Apache source tree, like PHP3:

```
$ cd php3
$ ./configure --with-shared-apache=../apache-1.3
$ apxs -c -o libphp3.so mod_php3.c libmodphp3-so.a
gcc -fpic -DSHARED_MODULE -I/tmp/apache/include -c mod_php3.c
ld -Bshareable -o libphp3.so mod_php3.o libmodphp3-so.a
$ _
```

because apxs automatically recognized C source files and object files. Only C source files are compiled while remaining object files are used for the linking phase.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## configure - Configure the source tree

The `configure` script configures the source tree for compiling and installing the Apache HTTP Server on your particular platform. Various options allow the compilation of a server corresponding to your personal requirements.

This script, included in the root directory of the source distribution, is for compilation on Unix and Unix-like systems only. For other platforms, see the [platform](#) documentation.

### See also

[Compiling and Installing](#)



You should call the `configure` script from within the root directory of the distribution.

```
./configure [OPTION]. . . [VAR=VALUE]. . .
```

To assign environment variables (e.g. `CC`, `CFLAGS` ...), specify them as `VAR=VALUE`. See [below](#) for descriptions of some of the useful variables.



- [Configuration options](#)
- [Installation directories](#)
- [System types](#)
- [Optional features](#)
- [Options for support programs](#)

## Configuration options

The following options influence the behavior of `configure` itself.

**-C**

**--config-cache**

This is an alias for `--cache-file=config.cache`

**--cache-file=FILE**

The test results will be cached in file *FILE*. This option is disabled by default.

**-h**

**--help [short|recursive]**

Output the help and exit. With the argument `short` only options specific to this package will be displayed. The argument `recursive` displays the short help of all the included packages.

**-n**

**--no-create**

The `configure` script is run normally but does not create output files. This is useful to check the test results before generating makefiles for compilation.

**-q**

**--quiet**

Do not print checking . . . messages during the `configure`

process.

**--srcdir=DIR**

Defines directory *DIR* to be the source file directory. Default is the directory, where configure is located, or the parent directory . . .

**--silent**

Same as --quiet

**-V**

**--version**

Display copyright information and exit.

## Installation directories

These options define the installation directory. The installation tree depends on the selected layout.

**--prefix=PREFIX**

Install architecture-independent files in *PREFIX*. By default the installation directory is set to `/usr/local/apache2`.

**--exec-prefix=EPREFIX**

Install architecture-dependent files in *EPREFIX*. By default the installation directory is set to the *PREFIX* directory.

By default, `make install` will install all the files in `/usr/local/apache2/bin`, `/usr/local/apache2/lib` etc. You can specify an installation prefix other than `/usr/local/apache2` using `--prefix`, for instance `--prefix=$HOME`.

## Define a directory layout

**--enable-layout=LAYOUT**

Configure the source code and build scripts to assume an

installation tree based on the layout *LAYOUT*. This allows you to separately specify the locations for each type of file within the Apache HTTP Server installation. The `config.layout` file contains several example configurations, and you can also create your own custom configuration following the examples. The different layouts in this file are grouped into `<Layout F00> . . . </Layout>` sections and referred to by name as in `F00`. The default layout is Apache.

### **Fine tuning of the installation directories**

For better control of the installation directories, use the options below. Please note that the directory defaults are set by `autoconf` and be overwritten by the corresponding layout setting.

#### **--bindir=*DIR***

Install user executables in *DIR*. The user executables are supporting programs like [htpasswd](#), [dbmmanage](#), etc. which are useful for site administrators. By default *DIR* is set to `EPREFIX/bin`.

#### **--datadir=*DIR***

Install read-only architecture-independent data in *DIR*. By default `datadir` is set to `PREFIX/share`. This option is offered by `autoconf` and currently unused.

#### **--includedir=*DIR***

Install C header files in *DIR*. By default `includedir` is set to `EPREFIX/include`.

#### **--infodir=*DIR***

Install info documentation in *DIR*. By default `infodir` is set to `PREFIX/info`. This option is currently unused.

#### **--libdir=*DIR***

Install object code libraries in *DIR*. By default `libdir` is set to `EPREFIX/lib`.

**--libexecdir=DIR**

Install the program executables (i.e., shared modules) in *DIR*. By default `libexecdir` is set to *EPREFIX/libexec*.

**--localstatedir=DIR**

Install modifiable single-machine data in *DIR*. By default `localstatedir` is set to *PREFIX/var*. This option is offered by `autoconf` and currently unused.

**--mandir=DIR**

Install the man documentation in *DIR*. By default `mandir` is set to *EPREFIX/man*.

**--oldincludedir=DIR**

Install C header files for non-gcc in *DIR*. By default `oldincludedir` is set to `/usr/include`. This option is offered by `autoconf` and currently unused.

**--sbindir=DIR**

Install the system administrator executables in *DIR*. Those are server programs like [httpd](#), [apachectl](#), [suexec](#), etc. which are necessary to run the Apache HTTP Server. By default `sbindir` is set to *EPREFIX/sbin*.

**--sharedstatedir=DIR**

Install modifiable architecture-independent data in *DIR*. By default `sharedstatedir` is set to *PREFIX/com*. This option is offered by `autoconf` and currently unused.

**--sysconfdir=DIR**

Install read-only single-machine data like the server configuration files `httpd.conf`, `mime.types`, etc. in *DIR*. By default `sysconfdir` is set to *PREFIX/conf*.

## System types

These options are used to cross-compile the Apache HTTP Server

to run on another system. In normal cases, when building and running the server on the same system, these options are not used.

**--build=*BUILD***

Defines the system type of the system on which the tools are being built. It defaults to the result of the script `config.guess`.

**--host=*HOST***

Defines the system type of the system on which the server will run. *HOST* defaults to *BUILD*.

**--target=*TARGET***

Configure for building compilers for the system type *TARGET*. It defaults to *HOST*. This option is offered by `autoconf` and not necessary for the Apache HTTP Server.

## Optional Features

These options are used to fine tune the features your HTTP server will have.

### General syntax

Generally you can use the following syntax to enable or disable a feature:

**--disable-*FEATURE***

Do not include *FEATURE*. This is the same as `--enable-FEATURE=no`.

**--enable-*FEATURE*[=*ARG*]**

Include *FEATURE*. The default value for *ARG* is yes.

**--enable-*MODULE*=shared**

The corresponding module will be build as DSO module.

**--enable-*MODULE*=static**

By default enabled modules are linked statically. You can force this explicitly.

### Note

`configure` will not complain about `--enable-foo` even if `foo` doesn't exist, so you need to type carefully.

### Modules enabled by default

Some modules are compiled by default and have to be disabled explicitly. Use the following options to remove discrete modules from the compilation process.

#### **--disable-actions**

Disable action triggering on requests, which is provided by [mod\\_actions](#).

#### **--disable-alias**

Disable the mapping of requests to different parts of the filesystem, which is provided by [mod\\_alias](#).

#### **--disable-asis**

Disable support for as-is filetypes, which is provided by [mod\\_asis](#).

#### **--disable-auth**

Disable user-based access control provided by [mod\\_auth](#). This module provides for HTTP Basic Authentication, where the usernames and passwords are stored in plain text files.

#### **--disable-autoindex**

Disable the directory listing functionality provided by [mod\\_autoindex](#).

#### **--disable-access**

Disable host-based access control provided by [mod\\_access](#).

#### **--disable-cgi**

[mod\\_cgi](#), which provides support for CGI scripts, is enabled by default when using a non-threaded MPM. Use this option to disable CGI support.

### **--disable-cgid**

When using the threaded MPMs [worker](#) or [perchild](#) support for CGI scripts is provided by [mod\\_cgid](#) by default. To disable CGI support use this option.

### **--disable-charset-lite**

Disable character set translation provided by [mod\\_charset\\_lite](#). This module will be installed by default only on EBCDIC systems.

### **--disable-dir**

Disable directory request handling provided by [mod\\_dir](#).

### **--disable-env**

Disable setting and clearing of environment variables, which is provided by [mod\\_env](#).

### **--disable-http**

Disable the HTTP protocol handling. The `http` module is a basic one, enabling the server to function as an HTTP server. It is only useful to disable it if you want to use another protocol module instead. **Don't disable this module unless you are really sure what you are doing.**

Note: This module will always be linked statically.

### **--disable-imap**

Disable support for server based imagemaps, which provided by [mod\\_imap](#).

### **--disable-include**

Disable Server Side Includes provided by [mod\\_include](#).

### **--disable-log-config**

Disable the logging configuration provided by

[mod\\_log\\_config](#). You won't be able to log requests to the server without this module.

#### **--disable-mime**

[mod\\_mime](#) associates the requested filename's extensions with the file's behavior and content (mime-type, language, character set and encoding). Disabling the mapping of file-extensions to MIME is normally not recommended.

#### **--disable-negotiation**

Disable content negotiation provided by [mod\\_negotiation](#).

#### **--disable-setenvif**

Disable support for basing environment variables on headers, which is provided by [mod\\_setenvif](#).

#### **--disable-status**

Disable the process/thread monitoring, which is provided by [mod\\_status](#).

#### **--disable-userdir**

Disable the mapping of requests to user-specific directories, which is provided by [mod\\_userdir](#).

### **Modules, disabled by default**

Some modules are compiled by default and have to be enabled explicitly or by using the keywords `most` or `all` (see `--enable-mods-shared` below for further explanation) to be available.

Therefore use the options below.

#### **--enable-auth-anon**

Enable anonymous user access provided by [mod\\_auth\\_anon](#).

#### **--enable-auth-dbm**

[mod\\_auth\\_dbm](#) provides for HTTP Basic Authentication, where the usernames and passwords are stored in DBM type

database files. Use this option to enable the module.

**--enable-auth-digest**

Enable RFC2617 Digest authentication provided by [mod\\_auth\\_digest](#). This module uses plain text files to store the credentials.

**--enable-auth-ldap**

Enable LDAP based authentication provided by [mod\\_auth\\_ldap](#).

**--enable-cache**

Enable dynamic file caching provided by [mod\\_cache](#). This experimental module may be interesting for servers with high load or caching proxy servers. At least one storage management module (e.g. [mod\\_disk\\_cache](#) or [mod\\_mem\\_cache](#)) is also necessary.

**--enable-cern-meta**

Enable the CERN-type meta files support provided by [mod\\_cern\\_meta](#).

**--enable-charset-lite**

Enable character set translation provided by [mod\\_charset\\_lite](#). This module will be installed by default only on EBCDIC systems. On other systems, you have to enable it.

**--enable-dav**

Enable the WebDAV protocol handling provided by [mod\\_dav](#). Support for filesystem resources is provided by the separate module [mod\\_dav\\_fs](#). This module is also automatically enabled with --enable-dav.

Note: [mod\\_dav](#) can only be used together with the http protocol module.

**--enable-dav-fs**

Enable DAV support for filesystem resources, which is

provided by [mod\\_dav\\_fs](#). This module is a provider for the [mod\\_dav](#) module, so you should also use `--enable-dav`.

**--enable-deflate**

Enable deflate transfer encoding provided by [mod\\_deflate](#).

**--enable-disk-cache**

Enable disk caching provided by [mod\\_disk\\_cache](#).

**--enable-expires**

Enable Expires header control provided by [mod\\_expires](#).

**--enable-ext-filter**

Enable the external filter support provided by [mod\\_ext\\_filter](#).

**--enable-file-cache**

Enable the file cache provided by [mod\\_file\\_cache](#).

**--enable-headers**

Enable control of HTTP headers provided by [mod\\_headers](#).

**--enable-info**

Enable the server information provided by [mod\\_info](#).

**--enable-ldap**

Enable LDAP caching and connection pooling services provided by [mod\\_ldap](#).

**--enable-logio**

Enable logging of input and output bytes including headers provided by [mod\\_logio](#).

**--enable-mem-cache**

Enable memory caching provided by [mod\\_mem\\_cache](#).

**--enable-mime-magic**

Enable automatic determining of MIME types, which is provided by [mod\\_mime\\_magic](#).

**--enable-isapi**

Enable the isapi extension support provided by [mod\\_isapi](#).

### **--enable-proxy**

Enable the proxy/gateway functionality provided by [mod\\_proxy](#). The proxying capabilities for CONNECT, FTP and HTTP are provided by the separate modules [mod\\_proxy\\_connect](#), [mod\\_proxy\\_ftp](#) and [mod\\_proxy\\_http](#). These three modules are also automatically enabled with --enable-proxy.

### **--enable-proxy-connect**

Enable proxy support for CONNECT request handling, which is provided by [mod\\_proxy\\_connect](#). This module is an extension for the [mod\\_proxy](#) module, so you should also use --enable-proxy.

### **--enable-proxy-ftp**

Enable proxy support for FTP requests, which is provided by [mod\\_proxy\\_ftp](#). This module is an extension for the [mod\\_proxy](#) module, so you should also use --enable-proxy.

### **--enable-proxy-http**

Enable proxy support for HTTP requests, which is provided by [mod\\_proxy\\_http](#). This module is an extension for the [mod\\_proxy](#) module, so you should also use --enable-proxy.

### **--enable-rewrite**

Enable rule based URL manipulation provided by [mod\\_rewrite](#).

### **--enable-so**

Enable DSO capability provided by [mod\\_so](#). This module will be automatically enabled if you use the --enable-mods-shared option.

**--enable-speling**

Enable the functionality to correct common URL misspellings, which is provided by [mod\\_speling](#).

**--enable-ssl**

Enable support for SSL/TLS provided by [mod\\_ssl](#).

**--enable-unique-id**

Enable the generation of per-request unique ids, which is provided by [mod\\_unique\\_id](#).

**--enable-usertrack**

Enable user-session tracking provided by [mod\\_usertrack](#).

**--enable-vhost-alias**

Enable mass virtual hosting provided by [mod\\_vhost\\_alias](#).

### Modules for developers

The following modules are useful only for developers and testing purposes and are disabled by default. Use the following options to enable them. If you are not sure whether you need one of these modules, omit them.

**--enable-bucketeer**

Enable the manipulation filter for buckets, which is provided by `mod_bucketeer`.

**--enable-case-filter**

Enable the example uppercase conversion output filter support of `mod_case_filter`.

**--enable-case-filter-in**

Enable the example uppercase conversion input filter support of `mod_case_filter_in`.

**--enable-echo**

Enable the ECHO server provided by [mod\\_echo](#).

**--enable-example**

Enable the example and demo module [mod\\_example](#).

**--enable-optional-fn-export**

Enable the example for an optional function exporter, which is provided by `mod_optional_fn_export`.

**--enable-optional-fn-import**

Enable the example for an optional function importer, which is provided by `mod_optional_fn_import`.

**--enable-optional-hook-export**

Enable the example for an optional hook exporter, which is provided by `mod_optional_hook_export`.

**--enable-optional-hook-import**

Enable the example optional hook importer, which is provided by `mod_optional_hook_import`.

## MPMs and third-party modules

To add the necessary Multi Processing Module and additional third-party modules use the following options:

**--with-module=*module-type:module-file*[, *module-type:module-file*]**

Add one or more third-party modules to the list of statically linked modules. The module source file *module-file* will be searched in the `modules/module-type` subdirectory of your Apache HTTP server source tree. If it is not found there `configure` is considering *module-file* to be an absolute file path and tries to copy the source file into the *module-type* subdirectory. If the subdirectory doesn't exist it will be created and populated with a standard `Makefile.in`.

This option is useful to add small external modules consisting of one source file. For more complex modules you should read the vendor's documentation.

## Note

If you want to build a DSO module instead of a statically linked use [apxs](#).

### **--with-mpm=MPM**

Choose the process model for your server. You have to select exactly one [Multi-Processing Module](#). Otherwise the [default MPM](#) for your operating system will be taken. Possible MPMs are [beos](#), [leader](#), [mpmt\\_os2](#), [perchild](#), [prefork](#), [threadpool](#) and [worker](#).

## Cumulative and other options

### **--enable-maintainer-mode**

Turn on debugging and compile time warnings.

### **--enable-mods-shared=MODULE-LIST**

Defines a list of modules to be enabled and build as dynamic shared modules. This mean, these module have to be loaded dynamically by using the [LoadModule](#) directive.

*MODULE-LIST* is a space separated list of modulenames enclosed by quotation marks. The module names are given without the preceding `mod_`. For example:

```
--enable-mods-shared='headers rewrite dav'
```

Additionally you can use the special keywords `all` and `most`. For example,

```
--enable-mods-shared=most
```

will compile most modules and build them as DSO modules.

### **--enable-modules=MODULE-LIST**

This option behaves similar to `--enable-mods-shared`, but will link the given modules statically. This means, these modules will always be present while running `httpd`. They need not be loaded with `LoadModule`.

**`--enable-v4-mapped`**

Allow IPv6 sockets to handle IPv4 connections.

**`--with-port=PORT`**

This defines the port on which `httpd` will listen. This port number is used when generating the configuration file `httpd.conf`. The default is 80.

**`--with-program-name`**

Define an alternative executable name. The default is `httpd`.

## Optional packages

These options are used to define optional packages.

### General syntax

Generally you can use the following syntax to define an optional package:

**`--with-PACKAGE[=ARG]`**

Use the package `PACKAGE`. The default value for `ARG` is `yes`.

**`--without-PACKAGE`**

Do not use the package `PACKAGE`. This is the same as `--with-PACKAGE=no`. This option is provided by `autoconf` but not very useful for the Apache HTTP Server.

### Specific packages

**`--with-apr=DIR|FILE`**

The Apache Portable Runtime (APR) is part of the `httpd`

source distribution and will automatically be build together with the HTTP server. If you want to use an already installed APR instead you have to tell configure the path to the `apr-config` script. You may set the absolute path and name or the directory to the installed APR. `apr-config` must exist within this directory or the subdirectory `bin`.

**--with-apr-util=DIR|FILE**

The Apache Portable Runtime Utilities (APU) are part of the `httpd` source distribution and will automatically be build together with the HTTP server. If you want to use an already installed APU instead you have to tell configure the path to the `apu-config` script. You may set the absolute path and name or the directory to the installed APU. `apu-config` must exist within this directory or the subdirectory `bin`.

**--with-ssl=DIR**

If `mod_ssl` has been enabled configure searches for an installed OpenSSL. You can set the directory path to the SSL/TLS toolkit instead.

**--with-z=DIR**

configure searches automatically for an installed `zlib` library if your source configuration requires one (e.g., when `mod_deflate` is enabled). You can set the directory path to the compression library instead.

Several features of the Apache HTTP Server, including `mod_authn_dbm` and `mod_rewrite`'s DBM `RewriteMap` use simple key/value databases for quick lookups of information. SDBM is included in the APU, so this database is always available. If you would like to use other database types, use the following options to enable them:

**--with-gdbm[=path]**

If no *path* is specified, `configure` will search for the include files and libraries of a GNU DBM installation in the usual search paths. An explicit *path* will cause `configure` to look in *path/lib* and *path/include* for the relevant files. Finally, the *path* may specify specific include and library paths separated by a colon.

**--with-ndbm[=*path*]**

Like `--with-gdbm`, but searches for a New DBM installation.

**--with-berkeley-db[=*path*]**

Like `--with-gdbm`, but searches for a Berkeley DB installation.

**Note**

The DBM options are provided by the APU and passed through to its configuration script. They are useless when using an already installed APU defined by `--with-apr-util`.

You may use more than one DBM implementation together with your HTTP server. The appropriated DBM type will be configured within the runtime configuration at each time.

## Options for support programs

**--enable-static-support**

Build a statically linked version of the support binaries. This means, a stand-alone executable will be built with all the necessary libraries integrated. Otherwise the support binaries are linked dynamically by default.

**--enable-suexec**

Use this option to enable [suexec](#), which allows you to set uid and gid for spawned processes. **Do not use this option unless you understand all the security implications of running a suid binary on your server.** Further options to

configure [suexec](#) are described [below](#).

It is possible to create a statically linked binary of a single support program by using the following options:

**--enable-static-ab**

Build a statically linked version of [ab](#).

**--enable-static-checkgid**

Build a statically linked version of [checkgid](#).

**--enable-static-htdbm**

Build a statically linked version of [htdbm](#).

**--enable-static-htdigest**

Build a statically linked version of [htdigest](#).

**--enable-static-htpasswd**

Build a statically linked version of [htpasswd](#).

**--enable-static-logresolve**

Build a statically linked version of [logresolve](#).

**--enable-static-rotatelog**

Build a statically linked version of [rotatelog](#).

### **suexec configuration options**

The following options are used to fine tune the behavior of [suexec](#). See [Configuring and installing suEXEC](#) or further information.

**--with-suexec-bin**

This defines the path to [suexec](#) binary. Default is `--sbindir` (see [Fine tuning of installation directories](#)).

**--with-suexec-caller**

This defines the user allowed to call [suexec](#). It should be the same as the user under which [httpd](#) normally runs.

### **--with-suexec-docroot**

This defines the directory tree under which [suexec](#) access is allowed for executables. Default value is --datadir/htdocs.

### **--with-suexec-gidmin**

Define this as the lowest GID allowed to be a target user for [suexec](#). The default value is 100.

### **--with-suexec-logfile**

This defines the filename of the [suexec](#) logfile. By default the logfile is named suexec\_log and located in --logfiledir.

### **--with-suexec-safepath**

Define the value of the environment variable PATH to be set for processes started by [suexec](#). Default value is /usr/local/bin:/usr/bin:/bin.

### **--with-suexec-userdir**

This defines the subdirectory under the user's directory that contains all executables for which [suexec](#) access is allowed. This setting is necessary when you want to use [suexec](#) together with user-specific directories (as provided by [mod\\_userdir](#)). The default is public\_html.

### **--with-suexec-uidmin**

Define this as the lowest UID allowed to be a target user for [suexec](#). The default value is 100.

### **--with-suexec-umask**

Set umask for processes started by [suexec](#). It defaults to your system settings.



---

There are some useful environment variables to override the choices made by `configure` or to help it to find libraries and programs with nonstandard names or locations.

## **CC**

Define the C compiler command to be used for compilation.

## **CFLAGS**

Set C compiler flags you want to use for compilation.

## **CPP**

Define the C preprocessor command to be used.

## **CPPFLAGS**

Set C/C++ preprocessor flags, e.g. `-Iincludedir` if you have headers in a nonstandard directory *includedir*.

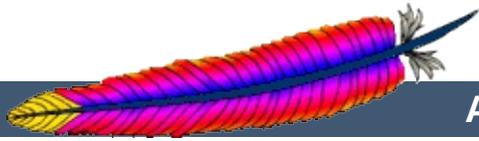
## **LDFLAGS**

Set linker flags, e.g. `-Llibdir` if you have libraries in a nonstandard directory *libdir*.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## dbmmanage - Manage user authentication files in DBM format

dbmmanage is used to create and update the DBM format files used to store usernames and password for basic authentication of HTTP users via [mod\\_auth\\_dbm](#). Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by dbmmanage. This program can only be used when the usernames are stored in a DBM file. To use a flat-file database see [htpasswd](#).

This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in [httpd](#) see the httpd manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

### See also

[httpd](#)

[mod\\_auth\\_dbm](#)



**dbmmmanage** [ *encoding* ] *filename*  
add|adduser|check|delete|update *username* [  
*encpasswd* [ *group*[,*group*...] [ *comment* ] ] ]

**dbmmmanage** *filename* view [ *username* ]

**dbmmmanage** *filename* import



### ***filename***

The filename of the DBM format file. Usually without the extension `.db`, `.pag`, or `.dir`.

### ***username***

The user for which the operations are performed. The *username* may not contain a colon (:).

### ***encpasswd***

This is the already encrypted password to use for the `update` and `add` commands. You may use a hyphen (-) if you want to get prompted for the password, but fill in the fields afterwards. Additionally when using the `update` command, a period (.) keeps the original password untouched.

### ***group***

A group, which the user is member of. A groupname may not contain a colon (:). You may use a hyphen (-) if you don't want to assign the user to a group, but fill in the comment field. Additionally when using the `update` command, a period (.) keeps the original groups untouched.

### ***comment***

This is the place for your opaque comments about the user, like `realname`, `mailaddress` or such things. The server will ignore this field.

## **Encodings**

**-d**

crypt encryption (default, except on Win32, Netware)

**-m**

MD5 encryption (default on Win32, Netware)

**-s**

SHA1 encryption

**-p**

plaintext (*not recommended*)

## Commands

### add

Adds an entry for *username* to *filename* using the encrypted password *enpasswd*.

```
dbmmanage passwords.dat add rbowen foKntnEF3KSXA
```

### adduser

Asks for a password and then adds an entry for *username* to *filename*.

```
dbmmanage passwords.dat adduser krietz
```

### check

Asks for a password and then checks if *username* is in *filename* and if its password matches the specified one.

```
dbmmanage passwords.dat check rbowen
```

### delete

Deletes the *username* entry from *filename*.

```
dbmmanage passwords.dat delete rbowen
```

### import

Reads *username:password* entries (one per line) from STDIN and adds them to *filename*. The passwords already have to be crypted.

### update

Same as the `adduser` command, except that it makes sure *username* already exists in *filename*.

```
dbmmanage passwords.dat update rbowen
```

### **view**

Just displays the contents of the DBM file. If you specify a *username*, it displays the particular record only.

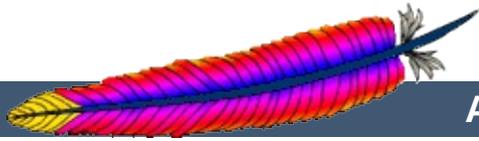
```
dbmmanage passwords.dat view
```



One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, the GNU project's GDBM, and Berkeley DB 2. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by *filename* is the same format that `dbmmanage` expects to see. `dbmmanage` currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

`dbmmanage` has a list of DBM format preferences, defined by the `@AnyDBM: :ISA` array near the beginning of the program. Since we prefer the Berkeley DB 2 file format, the order in which `dbmmanage` will look for system libraries is Berkeley DB 2, then NDBM, then GDBM and then SDBM. The first library found will be the library `dbmmanage` will attempt to use for all DBM file transactions. This ordering is slightly different than the standard `@AnyDBM: :ISA` ordering in Perl, as well as the ordering used by the simple `dbmopen( )` call in Perl, so if you use any other utilities to manage your DBM files, they must also follow this preference ordering. Similar care must be taken if using programs in other languages, like C, to access these files.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## htdbm - Manipulate DBM password databases

htdbm is used to manipulate the DBM format files used to store usernames and password for basic authentication of HTTP users via [mod\\_auth\\_dbm](#). See the [dbmmanage](#) documentation for more information about these DBM files.

### See also

[httpd](#)

[dbmmanage](#)

[mod\\_auth\\_dbm](#)



**htdbm** [ *-TDBTYPE* ] [ *-c* ] [ *-m* | *-d* | *-p* | *-s* ] [ *-t* ] [ *-v* ] [ *-x* ] *filename username*

**htdbm -b** [ *-TDBTYPE* ] [ *-c* ] [ *-m* | *-d* | *-p* | *-s* ] [ *-t* ] [ *-v* ] *filename username password*

**htdbm -n** [ *-c* ] [ *-m* | *-d* | *-p* | *-s* ] [ *-t* ] [ *-v* ] *username*

**htdbm -nb** [ *-c* ] [ *-m* | *-d* | *-p* | *-s* ] [ *-t* ] [ *-v* ] *username password*

**htdbm -v** [ *-TDBTYPE* ] [ *-c* ] [ *-m* | *-d* | *-p* | *-s* ] [ *-t* ] [ *-v* ] *filename username*

**htdbm -vb** [ *-TDBTYPE* ] [ *-c* ] [ *-m* | *-d* | *-p* | *-s* ] [ *-t* ] [ *-v* ] *filename username password*

**htdbm -x** [ *-TDBTYPE* ] [ *-m* | *-d* | *-p* | *-s* ] *filename username*

**htdbm -l** [ *-TDBTYPE* ]



## **-b**

Use batch mode; *i.e.*, get the password from the command line rather than prompting for it. This option should be used with extreme care, since **the password is clearly visible** on the command line.

## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is rewritten and truncated. This option cannot be combined with the `-n` option.

## **-n**

Display the results on standard output rather than updating a database. This option changes the syntax of the command line, since the *passwdfile* argument (usually the first one) is omitted. It cannot be combined with the `-c` option.

## **-m**

Use MD5 encryption for passwords. On Windows, Netware and TPF, this is the default.

## **-d**

Use `crypt ( )` encryption for passwords. The default on all platforms but Windows, Netware and TPF. Though possibly supported by `htdbm` on all platforms, it is not supported by the [httpd](#) server on Windows, Netware and TPF.

## **-s**

Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (Idif).

## **-p**

Use plaintext passwords. Though `htdbm` will support creation on all platforms, the [httpd](#) daemon will only accept plain text passwords on Windows, Netware and TPF.

**-l**

Print each of the usernames and comments from the database on stdout.

**-t**

Interpret the final parameter as a comment. When this option is specified, an additional string can be appended to the command line; this string will be stored in the "Comment" field of the database, associated with the specified username.

**-v**

Verify the username and password. The program will print a message indicating whether the supplied password is valid. If the password is invalid, the program exits with error code 3.

**-x**

Delete user. If the username exists in the specified DBM file, it will be deleted.

***filename***

The filename of the DBM format file. Usually without the extension `.db`, `.pag`, or `.dir`. If `-c` is given, the DBM file is created if it does not already exist, or updated if it does exist.

***username***

The username to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.

***password***

The plaintext password to be encrypted and stored in the DBM file. Used only with the `-b` flag.

***-TDBTYPE***

Type of DBM file (SDBM, GDBM, DB, or "default").



One should be aware that there are a number of different DBM file formats in existence, and with all likelihood, libraries for more than one format may exist on your system. The three primary examples are SDBM, NDBM, GNU GDBM, and Berkeley/Sleepycat DB 2/3/4. Unfortunately, all these libraries use different file formats, and you must make sure that the file format used by *filename* is the same format that `htdbm` expects to see. `htdbm` currently has no way of determining what type of DBM file it is looking at. If used against the wrong format, will simply return nothing, or may create a different DBM file with a different name, or at worst, it may corrupt the DBM file if you were attempting to write to it.

One can usually use the `file` program supplied with most Unix systems to see what format a DBM file is in.



htdbm returns a zero status ("true") if the username and password have been successfully added or updated in the DBM File. htdbm returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn't match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the [Restrictions section](#)), and 7 if the file is not a valid DBM password file.



## Examples

```
htdbm /usr/local/etc/apache/.htdbm-users jsmith
```

Adds or modifies the password for user `jsmith`. The user is prompted for the password. If executed on a Windows system, the password will be encrypted using the modified Apache MD5 algorithm; otherwise, the system's `crypt ( )` routine will be used. If the file does not exist, `htdbm` will do nothing except return an error.

```
htdbm -c /home/does/public_html/.htdbm jane
```

Creates a new file and stores a record in it for user `jane`. The user is prompted for the password. If the file exists and cannot be read, or cannot be written, it is not altered and `htdbm` will display a message and return an error status.

```
htdbm -mb /usr/web/.htdbm-all jones Pwd4Steve
```

Encrypts the password from the command line (`Pwd4Steve`) using the MD5 algorithm, and stores it in the specified file.



## Security Considerations

Web password files such as those managed by `htdbm` should *not* be within the Web server's URI space -- that is, they should not be fetchable with a browser.

The use of the `-b` option is discouraged, since when it is used the unencrypted password appears on the command line.



## Restrictions

On the Windows and MPE platforms, passwords encrypted with `htdbm` are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

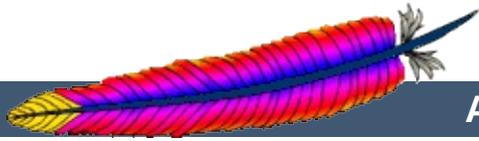
The MD5 algorithm used by `htdbm` is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Username are limited to 255 bytes and may not include the character `:`.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## htdigest - manage user files for digest authentication

htdigest is used to create and update the flat-files used to store usernames, realm and password for digest authentication of HTTP users. Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htdigest.

This manual page only lists the command line arguments. For details of the directives necessary to configure digest authentication in [httpd](#) see the Apache manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

### See also

[httpd](#)

[mod\\_auth\\_digest](#)



## Synopsis

```
htdigest [ -c ] passwdfile realm username
```



## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is deleted first.

### ***passwdfile***

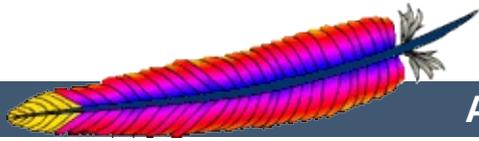
Name of the file to contain the username, realm and password. If -c is given, this file is created if it does not already exist, or deleted and recreated if it does exist.

### ***realm***

The realm name to which the user name belongs.

### ***username***

The user name to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## htpasswd - Manage user files for basic authentication

htpasswd is used to create and update the flat-files used to store usernames and password for basic authentication of HTTP users. If htpasswd cannot access a file, such as not being able to write to the output file or not being able to read the file in order to update it, it returns an error status and makes no changes.

Resources available from the Apache HTTP server can be restricted to just the users listed in the files created by htpasswd. This program can only manage usernames and passwords stored in a flat-file. It can encrypt and display password information for use in other types of data stores, though. To use a DBM database see [dbmmanage](#).

htpasswd encrypts passwords using either a version of MD5 modified for Apache, or the system's crypt ( ) routine. Files managed by htpasswd may contain both types of passwords; some user records may have MD5-encrypted passwords while others in the same file may have passwords encrypted with crypt ( ).

This manual page only lists the command line arguments. For details of the directives necessary to configure user authentication in [httpd](#) see the Apache manual, which is part of the Apache distribution or can be found at <http://httpd.apache.org/>.

### See also

[httpd](#)

The scripts in support/SHA1 which come with the distribution.



## Syntax

```
htpasswd [ -c ] [ -m ] [ -D ] passwdfile username
```

```
htpasswd -b [ -c ] [ -m | -d | -p | -s ] [ -D ]  
passwdfile username password
```

```
htpasswd -n [ -m | -d | -s | -p ] username
```

```
htpasswd -nb [ -m | -d | -s | -p ] username  
password
```



## **-b**

Use batch mode; *i.e.*, get the password from the command line rather than prompting for it. This option should be used with extreme care, since **the password is clearly visible** on the command line.

## **-c**

Create the *passwdfile*. If *passwdfile* already exists, it is rewritten and truncated. This option cannot be combined with the -n option.

## **-n**

Display the results on standard output rather than updating a file. This is useful for generating password records acceptable to Apache for inclusion in non-text data stores. This option changes the syntax of the command line, since the *passwdfile* argument (usually the first one) is omitted. It cannot be combined with the -c option.

## **-m**

Use MD5 encryption for passwords. On Windows, Netware and TPF, this is the default.

## **-d**

Use crypt ( ) encryption for passwords. The default on all platforms but Windows, Netware and TPF. Though possibly supported by `htpasswd` on all platforms, it is not supported by the [httpd](#) server on Windows, Netware and TPF.

## **-s**

Use SHA encryption for passwords. Facilitates migration from/to Netscape servers using the LDAP Directory Interchange Format (ldif).

## **-p**

Use plaintext passwords. Though `htpasswd` will support

creation on all platforms, the [httpd](#) daemon will only accept plain text passwords on Windows, Netware and TPF.

**-D**

Delete user. If the username exists in the specified htpasswd file, it will be deleted.

***passwdfile***

Name of the file to contain the user name and password. If `-c` is given, this file is created if it does not already exist, or rewritten and truncated if it does exist.

***username***

The username to create or update in *passwdfile*. If *username* does not exist in this file, an entry is added. If it does exist, the password is changed.

***password***

The plaintext password to be encrypted and stored in the file. Only used with the `-b` flag.



---

htpasswd returns a zero status ("true") if the username and password have been successfully added or updated in the *passwdfile*. htpasswd returns 1 if it encounters some problem accessing files, 2 if there was a syntax problem with the command line, 3 if the password was entered interactively and the verification entry didn't match, 4 if its operation was interrupted, 5 if a value is too long (username, filename, password, or final computed record), 6 if the username contains illegal characters (see the [Restrictions section](#)), and 7 if the file is not a valid password file.



## Examples

```
htpasswd /usr/local/etc/apache/.htpasswd-users jsmith
```

Adds or modifies the password for user `jsmith`. The user is prompted for the password. If executed on a Windows system, the password will be encrypted using the modified Apache MD5 algorithm; otherwise, the system's `crypt ( )` routine will be used. If the file does not exist, `htpasswd` will do nothing except return an error.

```
htpasswd -c /home/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 -mb /usr/web/.htpasswd-all jones Pwd4Steve
```

Encrypts the password from the command line (`Pwd4Steve`) using the MD5 algorithm, and stores it in the specified file.



## Security Considerations

Web password files such as those managed by `htpasswd` should *not* be within the Web server's URI space -- that is, they should not be fetchable with a browser.

The use of the `-b` option is discouraged, since when it is used the unencrypted password appears on the command line.



## Restrictions

On the Windows and MPE platforms, passwords encrypted with `htpasswd` are limited to no more than 255 characters in length. Longer passwords will be truncated to 255 characters.

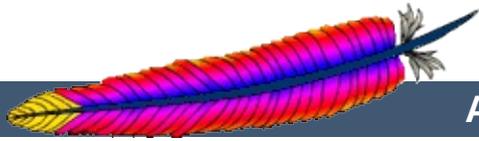
The MD5 algorithm used by `htpasswd` is specific to the Apache software; passwords encrypted using it will not be usable with other Web servers.

Username are limited to 255 bytes and may not include the character `:`.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## **logresolve - Resolve IP-addresses to hostnames in Apache log files**

logresolve is a post-processing program to resolve IP-addresses in Apache's access logfiles. To minimize impact on your nameserver, logresolve has its very own internal hash-table cache. This means that each IP number will only be looked up the first time it is found in the log file.

Takes an Apache log file on standard input. The IP addresses must be the first thing on each line and must be separated from the remainder of the line by a space.



## Synopsis

```
logresolve [ -s filename ] [ -c ] < access_log >  
access_log.new
```



## **-s *filename***

Specifies a filename to record statistics.

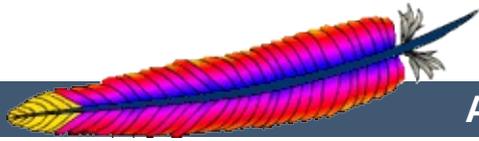
## **-c**

This causes `logresolve` to apply some DNS checks: after finding the hostname from the IP address, it looks up the IP addresses for the hostname and checks that one of these matches the original address.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## rotatelog - Piped logging program to rotate Apache logs

rotatelog is a simple program for use in conjunction with Apache's piped logfile feature. For example:

```
CustomLog "|bin/rotatelog /var/logs/logfile 86400" common
```

This creates the files `/var/logs/logfile.nnnn` where `nnnn` is the system time at which the log nominally starts (this time will always be a multiple of the rotation time, so you can synchronize cron scripts with it). At the end of each rotation time (here after 24 hours) a new log is started.

```
CustomLog "|bin/rotatelog /var/logs/logfile 5M" common
```

This configuration will rotate the logfile whenever it reaches a size of 5 megabytes.

```
ErrorLog "|bin/rotatelog /var/logs/errorlog.%Y-%m-%d-%H_%M_%S 5M"
```

This configuration will rotate the error logfile whenever it reaches a size of 5 megabytes, and the suffix to the logfile name will be created of the form `errorlog.YYYY-mm-dd-HH_MM_SS`.



## Synopsis

```
rotatelogs [ -l ] logfile [ rotationtime [ offset  
] ] | [ filesizeM ]
```



## **-1 (2.0.51 and later)**

Causes the use of local time rather than GMT as the base for the interval. Note that using `-1` in an environment which changes the GMT offset (such as for BST or DST) can lead to unpredictable results!

## ***logfile***

The path plus basename of the logfile. If *logfile* includes any '%' characters, it is treated as a format string for `strftime(3)`. Otherwise, the suffix `.nnnnnnnnnn` is automatically added and is the time in seconds. Both formats compute the start time from the beginning of the current period.

## ***rotationtime***

The time between log file rotations in seconds.

## ***offset***

The number of minutes offset from UTC. If omitted, zero is assumed and UTC is used. For example, to use local time in the zone UTC -5 hours, specify a value of `-300` for this argument.

## ***filesizeM***

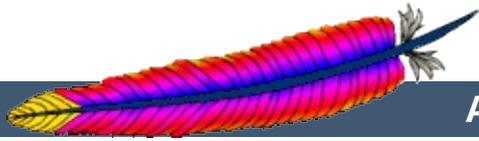
The maximum file size in megabytes followed by the letter M to specify size rather than time. Use this parameter in place of both *rotationtime* and *offset*.



The following logfile format string substitutions should be supported by all `strftime(3)` implementations, see the `strftime(3)` man page for library-specific extensions.

%A	full weekday name (localized)
%a	3-character weekday name (localized)
%B	full month name (localized)
%b	3-character month name (localized)
%c	date and time (localized)
%d	2-digit day of month
%H	2-digit hour (24 hour clock)
%I	2-digit hour (12 hour clock)
%j	3-digit day of year
%M	2-digit minute
%m	2-digit month
%p	am/pm of 12 hour clock (localized)
%S	2-digit second
%U	2-digit week of year (Sunday first day of week)
%W	2-digit week of year (Monday first day of week)
%w	1-digit weekday (Sunday first day of week)
%X	time (localized)
%x	date (localized)
%Y	4-digit year
%y	2-digit year
%Z	time zone name
%%	literal `%'

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## Other Programs

The following programs are simple support programs included with the Apache HTTP Server which do not have their own manual pages. They are not installed automatically. You can find them after the configuration process in the `support/` directory.



This perl script is designed to be run at a frequent interval by something like cron. It connects to the server and downloads the status information. It reformats the information to a single line and logs it to a file. Adjust the variables at the top of the script to specify the location of the resulting logfile.



This perl script will take a combined Web server access log file and break its contents into separate files. It assumes that the first field of each line is the virtual host identity (put there by "%v"), and that the logfiles should be named that + ".log" in the current directory.

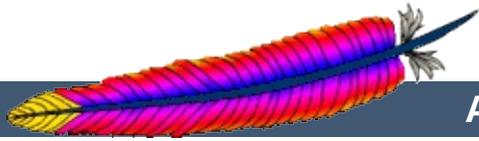
The combined log file is read from stdin. Records read will be appended to any existing log files.

```
split-logfile < access.log
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

# International Customized Server Error Messages

**Warning:**

This document has not been fully updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

This document describes an easy way to provide your Apache HTTP Server with a set of customized error messages which take advantage of [Content Negotiation](#) and `mod_include` to return error messages generated by the server in the client's native language.



By using SSI, all [ErrorDocument](#) messages can share a homogenous and consistent style and layout, and maintenance work (changing images, changing links) is kept to a minimum because all layout information can be kept in a single file.

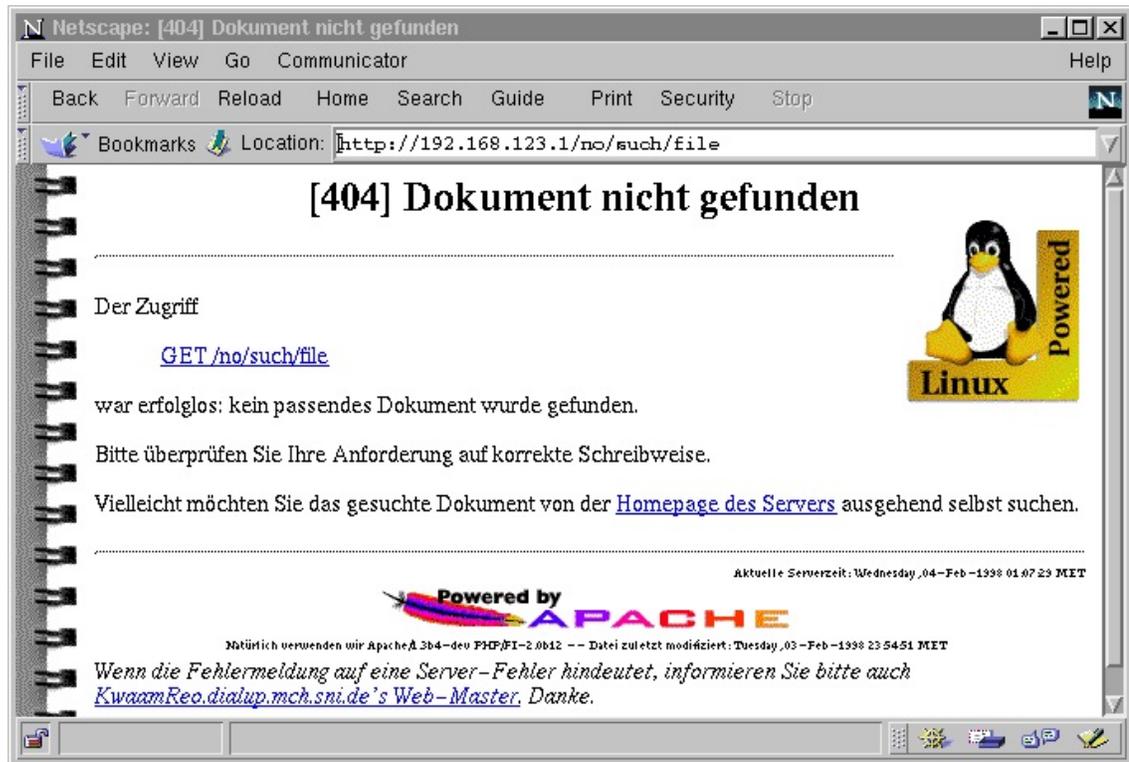
Error documents can be shared across different servers, or even hosts, because all varying information is inserted at the time the error document is returned on behalf of a failed request.

Content Negotiation then selects the appropriate language version of a particular error message text, honoring the language preferences passed in the client's request. (Users usually select their favorite languages in the preferences options menu of today's browsers). When an error document in the client's primary language version is unavailable, the secondary languages are tried or a default (fallback) version is used.

You have full flexibility in designing your error documents to your personal taste (or your company's conventions). For demonstration purposes, we present a simple generic error document scheme. For this hypothetical server, we assume that all error messages...

- possibly are served by different virtual hosts (different host name, different IP address, or different port) on the server machine,
- show a predefined company logo in the right top of the message (selectable by virtual host),
- print the error title first, followed by an explanatory text and (depending on the error context) help on how to resolve the error,
- have some kind of standardized background image,
- display an apache logo and a feedback email address at the bottom of the error message.

An example of a "document not found" message for a german client might look like this:



All links in the document as well as links to the server's administrator mail address, and even the name and port of the serving virtual host are inserted in the error document at "run-time", i.e., when the error actually occurs.



For this concept to work as easily as possible, we must take advantage of as much server support as we can get:

1. By defining the MultiViews [Options](#), we enable the language selection of the most appropriate language alternative (content negotiation).
2. By setting the [LanguagePriority](#) directive we define a set of default fallback languages in the situation where the client's browser did not express any preference at all.
3. By enabling [mod\\_include](#) (and disallowing execution of cgi scripts for security reasons), we allow the server to include building blocks of the error message, and to substitute the value of certain environment variables into the generated document (dynamic HTML) or even to conditionally include or omit parts of the text.
4. The [AddHandler](#) and [AddType](#) directives are useful for automatically SSI-expanding all files with a `.shtml` suffix to *text/html*.
5. By using the [Alias](#) directive, we keep the error document directory outside of the document tree because it can be regarded more as a server part than part of the document tree.
6. The [<Directory>](#) block restricts these "special" settings to the error document directory and avoids an impact on any of the settings for the regular document tree.
7. For each of the error codes to be handled (see RFC2068 for an exact description of each error code, or look at `src/main/http_protocol.c` if you wish to see apache's standard messages), an [ErrorDocument](#) in the aliased `/error_docs` directory is defined. Note that we only define

the basename of the document here because the MultiViews option will select the best candidate based on the language suffixes and the client's preferences. Any error situation with an error code *not* handled by a custom document will be dealt with by the server in the standard way (*i.e.*, a plain error message in english).

8. Finally, the `AllowOverride` directive tells apache that it is not necessary to look for a `.htaccess` file in the `/error_docs` directory: a minor speed optimization.

The resulting `httpd.conf` configuration would then look similar to this:

### Note

Note that you can define your own error messages using this method for only part of the document tree, e.g., a `/~user/` subtree. In this case, the configuration could as well be put into the `.htaccess` file at the root of the subtree, and the `<Directory>` and `</Directory>` directives -but not the contained directives- must be omitted.

```
LanguagePriority en fr de
Alias /error_docs /usr/local/apache/error_docs

<Directory /usr/local/apache/error_docs>
    AllowOverride none
    Options MultiViews IncludesNoExec FollowSymLinks
    AddType text/html .shtml
    <FilesMatch "\.shtml[.]">
        SetOutputFilter INCLUDES
    </FilesMatch>
</Directory>

# "400 Bad Request",
ErrorDocument 400 /error_docs/400
# "401 Authorization Required",
ErrorDocument 401 /error_docs/401
# "403 Forbidden",
ErrorDocument 403 /error_docs/403
```

```
# "404 Not Found",  
ErrorDocument 404 /error_docs/404  
# "500 Internal Server Error",  
ErrorDocument 500 /error_docs/500
```

The directory for the error messages (here: `/usr/local/apache/error_docs/`) must then be created with the appropriate permissions (readable and executable by the server uid or gid, only writable for the administrator).

## Naming the Individual Error Document files

By defining the `MULTIViews` option, the server was told to automatically scan the directory for matching variants (looking at language and content type suffixes) when a requested document was not found. In the configuration, we defined the names for the error documents to be just their error number (without any suffix).

The names of the individual error documents are now determined like this (I'm using 403 as an example, think of it as a placeholder for any of the configured error documents):

- No file `error_docs/403` should exist. Otherwise, it would be found and served (with the `DefaultType`, usually `text/plain`), all negotiation would be bypassed.
- For each language for which we have an internationalized version (note that this need not be the same set of languages for each error code - you can get by with a single language version until you actually *have* translated versions), a document `error_docs/403.shtml.lang` is created and filled with the error text in that language ([see below](#)).
- One fallback document called `error_docs/403.shtml` is created, usually by creating a symlink to the default language variant ([see below](#)).

## The Common Header and Footer Files

By putting as much layout information in two special "include files", the error documents can be reduced to a bare minimum.

One of these layout files defines the HTML document header and a configurable list of paths to the icons to be shown in the resulting error document. These paths are exported as a set of SSI environment variables and are later evaluated by the "footer" special file. The title of the current error (which is put into the TITLE tag and an H1 header) is simply passed in from the main error document in a variable called `title`.

**By changing this file, the layout of all generated error messages can be changed in a second.** (By exploiting the features of SSI, you can easily define different layouts based on the current virtual host, or even based on the client's domain name).

The second layout file describes the footer to be displayed at the bottom of every error message. In this example, it shows an apache logo, the current server time, the server version string and adds a mail reference to the site's webmaster.

For simplicity, the header file is simply called `head.shtml` because it contains server-parsed content but no language specific information. The footer file exists once for each language translation, plus a symlink for the default language.

```
for English, French and German versions (default english)

foot.shtml.en,
foot.shtml.fr,
foot.shtml.de,
foot.shtml symlink to
foot.shtml.en
```

Both files are included into the error document by using the directives `<!--#include virtual="head" -->` and `<!--`

`#include virtual="foot" -->` respectively: the rest of the magic occurs in `mod_negotiation` and in `mod_include`.

See [the listings below](#) to see an actual HTML implementation of the discussed example.

## Creating ErrorDocuments in Different Languages

After all this preparation work, little remains to be said about the actual documents. They all share a simple common structure:

```
<!--#set var="title" value="error description title" -->
<!--#include virtual="head" -->
    explanatory error text
<!--#include virtual="foot" -->
```

In the [listings section](#), you can see an example of a [400 Bad Request] error document. Documents as simple as that certainly cause no problems to translate or expand.

## The Fallback Language

Do we need a special handling for languages other than those we have translations for? We did set the `LanguagePriority`, didn't we?!

Well, the `LanguagePriority` directive is for the case where the client does not express any language priority at all. But what happens in the situation where the client wants one of the languages we do not have, and none of those we do have?

Without doing anything, the Apache server will usually return a [406 no acceptable variant] error, listing the choices from which the client may select. But we're in an error message already, and important error information might get lost when the client had to choose a language representation first.

So, in this situation it appears to be easier to define a fallback

language (by copying or linking, e.g., the english version to a language-less version). Because the negotiation algorithm prefers "more specialized" variants over "more generic" variants, these generic alternatives will only be chosen when the normal negotiation did not succeed.

A simple shell script to do it (execute within the errordocs/ dir):

```
for f in *.shtml.en
do
  ln -s $f `basename $f .en`
done
```



## Customizing Proxy Error Messages

As of Apache-1.3, it is possible to use the `ErrorDocument` mechanism for proxy error messages as well (previous versions always returned fixed predefined error messages).

Most proxy errors return an error code of [500 Internal Server Error]. To find out whether a particular error document was invoked on behalf of a proxy error or because of some other server error, and what the reason for the failure was, you can check the contents of the new `ERROR_NOTES` CGI environment variable: if invoked for a proxy error, this variable will contain the actual proxy error message text in HTML form.

The following excerpt demonstrates how to exploit the `ERROR_NOTES` variable within an error document:

```
<!--#if expr="$REDIRECT_ERROR_NOTES = ''" -->

<p>
  The server encountered an unexpected condition
  which prevented it from fulfilling the request.
</p>

<p>
  <a href="mailto:<!--#echo var="SERVER_ADMIN" -->"
  SUBJECT="Error message [<!--#echo var="REDIRECT_STATUS" -->]
  <!--#echo var="title" --> for <!--#echo var="REQUEST_URI" --
  >">
  Please forward this error screen to <!--#echo
  var="SERVER_NAME" -->'s
  WebMaster</a>; it includes useful debugging information
  about
  the Request which caused the error.

  <pre><!--#printenv --></pre>
</p>

<!--#else -->
  <!--#echo var="REDIRECT_ERROR_NOTES" -->

<!--#endif -->
```



So, to summarize our example, here's the complete listing of the `400.shtml.en` document. You will notice that it contains almost nothing but the error text (with conditional additions). Starting with this example, you will find it easy to add more error documents, or to translate the error documents to different languages.

```
<!--#set var="title" value="Bad Request"-->
<!--#include virtual="head" -->

<p>
  Your browser sent a request that this server could not
  understand:
  <blockquote>
    <strong><!--#echo var="REQUEST_URI" --></strong>
  </blockquote>

  The request could not be understood by the server due to
  malformed
  syntax. The client should not repeat the request without
  modifications.
</p>

<p>
  <!--#if expr="$HTTP_REFERER != ''" -->
    Please inform the owner of
    <a href="<!--#echo var="HTTP_REFERER" -->">the referring
    page</a> about
    the malformed link.

  <!--#else -->
    Please check your request for typing errors and retry.

  <!--#endif -->
</p>

<!--#include virtual="foot" -->
```

Here is the complete `head.shtml.en` file (the funny line breaks avoid empty lines in the document after SSI processing). Note the configuration section at top. That's where you configure the images and logos as well as the apache documentation directory. Look how this file displays two different logos depending on the

content of the virtual host name (\$SERVER\_NAME), and that an animated apache logo is shown if the browser appears to support it (the latter requires server configuration lines of the form

```
BrowserMatch "^Mozilla/[2-4]" anigif
```

for browser types which support animated GIFs).

```
<!--#if expr="$SERVER_NAME = /*\..mycompany\.com/" -->
  <!--#set var="IMG_CorpLogo"
  value="http://$SERVER_NAME:$SERVER_PORT/errordocs/CorpLogo.gif"
  -->
  <!--#set var="ALT_CorpLogo" value="Powered by Linux!" -->

<!--#else -->
  <!--#set var="IMG_CorpLogo"
  value="http://$SERVER_NAME:$SERVER_PORT/errordocs/PrivLogo.gif"
  -->
  <!--#set var="ALT_CorpLogo" value="Powered by Linux!" -->
<!--#endif-->

<!--#set var="IMG_BgImage"
value="http://$SERVER_NAME:$SERVER_PORT/errordocs/BgImage.gif"
-->
<!--#set var="DOC_Apache"
value="http://$SERVER_NAME:$SERVER_PORT/Apache/" -->

<!--#if expr="$anigif" -->
  <!--#set var="IMG_Apache"
  value="http://$SERVER_NAME:$SERVER_PORT/icons/apache_anim.gif"
  -->
<!--#else-->
  <!--#set var="IMG_Apache"
  value="http://$SERVER_NAME:$SERVER_PORT/icons/apache_pb.gif"
  -->
<!--#endif-->

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
  <title>
    [ <!--#echo var="REDIRECT_STATUS" --> ] <!--#echo var="title"
    -->
  </title>
</head>
```

```

<body bgcolor="white" background="<!--#echo var="IMG_BgImage" -
->">
  <h1 align="center">
    [<!--#echo var="REDIRECT_STATUS" -->] <!--#echo var="title"
    -->
    "
      alt="<!--#echo var="ALT_CorpLogo" -->" align="right">
  </h1>

  <hr /> <!--
  ===== -->
  <div>

```

and this is the foot.shtml.en file:

```

</div>
<hr />

<div align="right">
  <small>Local Server time: <!--#echo var="DATE_LOCAL" -->
  </small>
</div>

<div align="center">
  <a href="<!--#echo var="DOC_Apache" -->">
  " border="0"
  align="bottom"
    alt="Powered by <!--#echo var="SERVER_SOFTWARE" -->">
  </a>
  <br />
  <small><!--#set var="var" value="Powered by
  $SERVER_SOFTWARE --
  File last modified on $LAST_MODIFIED" -->
  <!--#echo var="var" --></small>
</div>

<p>If the indicated error looks like a misconfiguration,
please inform
<a href="mailto:<!--#echo var="SERVER_ADMIN" -->"
subject="Feedback about Error message [<!--#echo
var="REDIRECT_STATUS" -->]
<!--#echo var="title" -->, req=<!--#echo var="REQUEST_URI" -
->">
<!--#echo var="SERVER_NAME" -->'s WebMaster</a>.
</p>

</body>
</html>

```

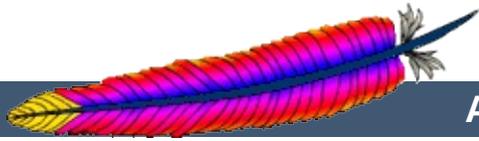
---

If you have tips to contribute, send mail to [martin@apache.org](mailto:martin@apache.org)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

## Connections in the FIN\_WAIT\_2 state and Apache

### Warning:

This document has not been fully updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

Starting with the Apache 1.2 betas, people are reporting many more connections in the FIN\_WAIT\_2 state (as reported by `netstat`) than they saw using older versions. When the server closes a TCP connection, it sends a packet with the FIN bit set to the client, which then responds with a packet with the ACK bit set. The client then sends a packet with the FIN bit set to the server, which responds with an ACK and the connection is closed. The state that the connection is in during the period between when the server gets the ACK from the client and the server gets the FIN from the client is known as FIN\_WAIT\_2. See the [TCP RFC](#) for the technical details of the state transitions.

The FIN\_WAIT\_2 state is somewhat unusual in that there is no timeout defined in the standard for it. This means that on many operating systems, a connection in the FIN\_WAIT\_2 state will stay around until the system is rebooted. If the system does not have a timeout and too many FIN\_WAIT\_2 connections build up, it can fill up the space allocated for storing information about the connections and crash the kernel. The connections in FIN\_WAIT\_2 do not tie up an `httpd` process.



There are numerous reasons for it happening, some of them may not yet be fully clear. What is known follows.

## Buggy Clients and Persistent Connections

Several clients have a bug which pops up when dealing with persistent connections (aka keepalives). When the connection is idle and the server closes the connection (based on the [KeepAliveTimeout](#)), the client is programmed so that the client does not send back a FIN and ACK to the server. This means that the connection stays in the FIN\_WAIT\_2 state until one of the following happens:

- The client opens a new connection to the same or a different site, which causes it to fully close the older connection on that socket.
- The user exits the client, which on some (most?) clients causes the OS to fully shutdown the connection.
- The FIN\_WAIT\_2 times out, on servers that have a timeout for this state.

If you are lucky, this means that the buggy client will fully close the connection and release the resources on your server. However, there are some cases where the socket is never fully closed, such as a dialup client disconnecting from their provider before closing the client. In addition, a client might sit idle for days without making another connection, and thus may hold its end of the socket open for days even though it has no further use for it. **This is a bug in the browser or in its operating system's TCP implementation.**

The clients on which this problem has been verified to exist:

- Mozilla/3.01 (X11; I; FreeBSD 2.1.5-RELEASE i386)
- Mozilla/2.02 (X11; I; FreeBSD 2.1.5-RELEASE i386)
- Mozilla/3.01Gold (X11; I; SunOS 5.5 sun4m)

- MSIE 3.01 on the Macintosh
- MSIE 3.01 on Windows 95

This does not appear to be a problem on:

- Mozilla/3.01 (Win95; I)

It is expected that many other clients have the same problem. What a client **should do** is periodically check its open socket(s) to see if they have been closed by the server, and close their side of the connection if the server has closed. This check need only occur once every few seconds, and may even be detected by a OS signal on some systems (e.g., Win95 and NT clients have this capability, but they seem to be ignoring it).

Apache **cannot** avoid these FIN\_WAIT\_2 states unless it disables persistent connections for the buggy clients, just like we recommend doing for Navigator 2.x clients due to other bugs. However, non-persistent connections increase the total number of connections needed per client and slow retrieval of an image-laden web page. Since non-persistent connections have their own resource consumptions and a short waiting period after each closure, a busy server may need persistence in order to best serve its clients.

As far as we know, the client-caused FIN\_WAIT\_2 problem is present for all servers that support persistent connections, including Apache 1.1.x and 1.2.

## **A necessary bit of code introduced in 1.2**

While the above bug is a problem, it is not the whole problem. Some users have observed no FIN\_WAIT\_2 problems with Apache 1.1.x, but with 1.2b enough connections build up in the FIN\_WAIT\_2 state to crash their server. The most likely source for additional FIN\_WAIT\_2 states is a function called

`linger_close()` which was added between 1.1 and 1.2. This function is necessary for the proper handling of persistent connections and any request which includes content in the message body (e.g., PUTs and POSTs). What it does is read any data sent by the client for a certain time after the server closes the connection. The exact reasons for doing this are somewhat complicated, but involve what happens if the client is making a request at the same time the server sends a response and closes the connection. Without lingering, the client might be forced to reset its TCP input buffer before it has a chance to read the server's response, and thus understand why the connection has closed. See the [appendix](#) for more details.

The code in `linger_close()` appears to cause problems for a number of factors, including the change in traffic patterns that it causes. The code has been thoroughly reviewed and we are not aware of any bugs in it. It is possible that there is some problem in the BSD TCP stack, aside from the lack of a timeout for the `FIN_WAIT_2` state, exposed by the `linger_close` code that causes the observed problems.



There are several possible workarounds to the problem, some of which work better than others.

## **Add a timeout for FIN\_WAIT\_2**

The obvious workaround is to simply have a timeout for the FIN\_WAIT\_2 state. This is not specified by the RFC, and could be claimed to be a violation of the RFC, but it is widely recognized as being necessary. The following systems are known to have a timeout:

- [FreeBSD](#) versions starting at 2.0 or possibly earlier.
- [NetBSD](#) version 1.2(?)
- [OpenBSD](#) all versions(?)
- [BSD/OS](#) 2.1, with the [K210-027](#) patch installed.
- [Solaris](#) as of around version 2.2. The timeout can be tuned by using `ndd` to modify `tcp_fin_wait_2_flush_interval`, but the default should be appropriate for most servers and improper tuning can have negative impacts.
- [Linux](#) 2.0.x and earlier(?)
- [HP-UX](#) 10.x defaults to terminating connections in the FIN\_WAIT\_2 state after the normal keepalive timeouts. This does not refer to the persistent connection or HTTP keepalive timeouts, but the `SO_LINGER` socket option which is enabled by Apache. This parameter can be adjusted by using `net tune` to modify parameters such as `tcp_keepstart` and `tcp_keeptop`. In later revisions, there is an explicit timer for connections in FIN\_WAIT\_2 that can be modified; contact HP support for details.
- [SGI IRIX](#) can be patched to support a timeout. For IRIX 5.3, 6.2, and 6.3, use patches 1654, 1703 and 1778 respectively. If you have trouble locating these patches, please contact your SGI support channel for help.
- [NCR's MP RAS Unix](#) 2.xx and 3.xx both have FIN\_WAIT\_2

timeouts. In 2.xx it is non-tunable at 600 seconds, while in 3.xx it defaults to 600 seconds and is calculated based on the tunable "max keep alive probes" (default of 8) multiplied by the "keep alive interval" (default 75 seconds).

- [Sequent's ptx/TCP/IP for DYNIX/ptx](#) has had a FIN\_WAIT\_2 timeout since around release 4.1 in mid-1994.

The following systems are known to not have a timeout:

- [SunOS 4.x](#) does not and almost certainly never will have one because it is at the very end of its development cycle for Sun. If you have kernel source should be easy to patch.

There is a [patch available](#) for adding a timeout to the FIN\_WAIT\_2 state; it was originally intended for BSD/OS, but should be adaptable to most systems using BSD networking code. You need kernel source code to be able to use it.

## **Compile without using `linger_close()`**

It is possible to compile Apache 1.2 without using the `linger_close()` function. This will result in that section of code being similar to that which was in 1.1. If you do this, be aware that it can cause problems with PUTs, POSTs and persistent connections, especially if the client uses pipelining. That said, it is no worse than on 1.1, and we understand that keeping your server running is quite important.

To compile without the `linger_close()` function, add `-DNO_LINGCLOSE` to the end of the `EXTRA_CFLAGS` line in your Configuration file, rerun [Configure](#) and rebuild the server.

## **Use `SO_LINGER` as an alternative to `linger_close()`**

On most systems, there is an option called `SO_LINGER` that can be set with `setsockopt(2)`. It does something very similar to `linger_close()`, except that it is broken on many systems so that it causes far more problems than `linger_close`. On some systems, it could possibly work better so it may be worth a try if you have no other alternatives.

To try it, add `-DUSE_SO_LINGER -DNO_LINGCLOSE` to the end of the `EXTRA_CFLAGS` line in your Configuration file, rerun [Configure](#) and rebuild the server.

#### **NOTE**

Attempting to use `SO_LINGER` and `linger_close()` at the same time is very likely to do very bad things, so don't.

## **Increase the amount of memory used for storing connection state**

### **BSD based networking code:**

BSD stores network data, such as connection states, in something called an mbuf. When you get so many connections that the kernel does not have enough mbufs to put them all in, your kernel will likely crash. You can reduce the effects of the problem by increasing the number of mbufs that are available; this will not prevent the problem, it will just make the server go longer before crashing.

The exact way to increase them may depend on your OS; look for some reference to the number of "mbufs" or "mbuf clusters". On many systems, this can be done by adding the line `NMBCLUSTERS="n"`, where `n` is the number of mbuf clusters you want to your kernel config file and rebuilding your kernel.

## **Disable KeepAlive**

If you are unable to do any of the above then you should, as a last resort, disable KeepAlive. Edit your httpd.conf and change "KeepAlive On" to "KeepAlive Off".



Below is a message from Roy Fielding, one of the authors of HTTP/1.1.

## **Why the lingering close functionality is necessary with HTTP**

The need for a server to linger on a socket after a close is noted a couple times in the HTTP specs, but not explained. This explanation is based on discussions between myself, Henrik Frystyk, Robert S. Thau, Dave Raggett, and John C. Mallery in the hallways of MIT while I was at W3C.

If a server closes the input side of the connection while the client is sending data (or is planning to send data), then the server's TCP stack will signal an RST (reset) back to the client. Upon receipt of the RST, the client will flush its own incoming TCP buffer back to the un-ACKed packet indicated by the RST packet argument. If the server has sent a message, usually an error response, to the client just before the close, and the client receives the RST packet before its application code has read the error message from its incoming TCP buffer and before the server has received the ACK sent by the client upon receipt of that buffer, then the RST will flush the error message before the client application has a chance to see it. The result is that the client is left thinking that the connection failed for no apparent reason.

There are two conditions under which this is likely to occur:

1. sending POST or PUT data without proper authorization
2. sending multiple requests before each response (pipelining) and one of the middle requests resulting in an error or other break-the-connection result.

The solution in all cases is to send the response, close only the

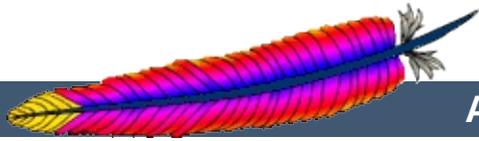
write half of the connection (what shutdown is supposed to do), and continue reading on the socket until it is either closed by the client (signifying it has finally read the response) or a timeout occurs. That is what the kernel is supposed to do if `SO_LINGER` is set. Unfortunately, `SO_LINGER` has no effect on some systems; on some other systems, it does not have its own timeout and thus the TCP memory segments just pile-up until the next reboot (planned or not).

Please note that simply removing the linger code will not solve the problem -- it only moves it to a different and much harder one to detect.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

## Known Problems in Clients

### Warning:

This document has not been fully updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

Over time the Apache Group has discovered or been notified of problems with various clients which we have had to work around, or explain. This document describes these problems and the workarounds available. It's not arranged in any particular order. Some familiarity with the standards is assumed, but not necessary.

For brevity, *Navigator* will refer to Netscape's Navigator product (which in later versions was renamed "Communicator" and various other names), and *MSIE* will refer to Microsoft's Internet Explorer product. All trademarks and copyrights belong to their respective companies. We welcome input from the various client authors to correct inconsistencies in this paper, or to provide us with exact version numbers where things are broken/fixed.

For reference, [RFC1945](#) defines HTTP/1.0, and [RFC2068](#) defines HTTP/1.1. Apache as of version 1.2 is an HTTP/1.1 server (with an optional HTTP/1.0 proxy).

Various of these workarounds are triggered by environment variables. The admin typically controls which are set, and for which clients, by using `mod_browser`. Unless otherwise noted all of these workarounds exist in versions 1.2 and later.



## Handling CRLF on the server

This is a legacy issue. The CERN webserver required POST data to have an extra CRLF following it. Thus many clients send an extra CRLF that is not included in the Content - Length of the request. Apache works around this problem by eating any empty lines which appear before a request.



Various clients have had broken implementations of *keepalive* (persistent connections). In particular the Windows versions of Navigator 2.0 get very confused when the server times out an idle connection. The workaround is present in the default config files:

```
BrowserMatch Mozilla/2 nokeepalive
```

Note that this matches some earlier versions of MSIE, which began the practice of calling themselves *Mozilla* in their user-agent strings just like Navigator.

MSIE 4.0b2, which claims to support HTTP/1.1, does not properly support *keepalive* when it is used on 301 or 302 (redirect) responses. Unfortunately Apache's `nokeepalive` code prior to 1.2.2 would not work with HTTP/1.1 clients. You must apply [this patch](#) to version 1.2.1. Then add this to your config:

```
BrowserMatch "MSIE 4\.0b2;" nokeepalive
```



To quote from section 3.1 of RFC1945:

HTTP uses a "<MAJOR>.<MINOR>" numbering scheme to indicate versions of the protocol. The protocol versioning policy is intended to allow the sender to indicate the format of a message and its capacity for understanding further HTTP communication, rather than the features obtained via that communication.

Since Apache is an HTTP/1.1 server, it indicates so as part of its response. Many client authors mistakenly treat this part of the response as an indication of the protocol that the response is in, and then refuse to accept the response.

The first major indication of this problem was with AOL's proxy servers. When Apache 1.2 went into beta it was the first widespread HTTP/1.1 server. After some discussion, AOL fixed their proxies. In anticipation of similar problems, the `force-response-1.0` environment variable was added to Apache. When present Apache will indicate "HTTP/1.0" in response to an HTTP/1.0 client, but will not in any other way change the response.

The pre-1.1 Java Development Kit (JDK) that is used in many clients (including Navigator 3.x and MSIE 3.x) exhibits this problem. As do some of the early pre-releases of the 1.1 JDK. We think it is fixed in the 1.1 JDK release. In any event the workaround:

```
BrowserMatch Java/1.0 force-response-1.0  
BrowserMatch JDK/1.0 force-response-1.0
```

RealPlayer 4.0 from Progressive Networks also exhibits this problem. However they have fixed it in version 4.01 of the player,

but version 4.01 uses the same User-Agent as version 4.0. The workaround is still:

```
BrowserMatch "RealPlayer 4.0" force-response-1.0
```



## HTTP/1.0

MSIE 4.0b2 has this problem. Its Java VM makes requests in HTTP/1.1 format but the responses must be in HTTP/1.0 format (in particular, it does not understand *chunked* responses). The workaround is to fool Apache into believing the request came in HTTP/1.0 format.

```
BrowserMatch "MSIE 4\.0b2;" downgrade-1.0 force-response-1.0
```

This workaround is available in 1.2.2, and in a [patch](#) against 1.2.1.



## Security problems with header padding

All versions of Navigator from 2.0 through 4.0b2 (and possibly later) have a problem if the trailing CRLF of the response header starts at offset 256, 257 or 258 of the response. A BrowserMatch for this would match on nearly every hit, so the workaround is enabled automatically on all responses. The workaround implemented detects when this condition would occur in a response and adds extra padding to the header to push the trailing CRLF past offset 258 of the response.



## Multipart responses and quoted boundary strings

On multipart responses some clients will not accept quotes (") around the boundary string. The MIME standard recommends that such quotes be used. But the clients were probably written based on one of the examples in RFC2068, which does not include quotes. Apache does not include quotes on its boundary strings to workaround this problem.



## byterange request

A byterange request is used when the client wishes to retrieve a portion of an object, not necessarily the entire object. There was a very old draft which included these byteranges in the URL. Old clients such as Navigator 2.0b1 and MSIE 3.0 for the MAC exhibit this behaviour, and it will appear in the servers' access logs as (failed) attempts to retrieve a URL with a trailing ";xxx-yyy". Apache does not attempt to implement this at all.

A subsequent draft of this standard defines a header Request - Range, and a response type multipart/x-byteranges. The HTTP/1.1 standard includes this draft with a few fixes, and it defines the header Range and type multipart/byteranges.

Navigator (versions 2 and 3) sends both Range and Request - Range headers (with the same value), but does not accept a multipart/byteranges response. The response must be multipart/x-byteranges. As a workaround, if Apache receives a Request - Range header it considers it "higher priority" than a Range header and in response uses multipart/x-byteranges.

The Adobe Acrobat Reader plugin makes extensive use of byteranges and prior to version 3.01 supports only the multipart/x-byterange response. Unfortunately there is no clue that it is the plugin making the request. If the plugin is used with Navigator, the above workaround works fine. But if the plugin is used with MSIE 3 (on Windows) the workaround won't work because MSIE 3 doesn't give the Range - Request clue that Navigator does. To workaround this, Apache special cases "MSIE 3" in the User - Agent and serves multipart/x-byteranges. Note that the necessity for this with MSIE 3 is actually due to the Acrobat plugin, not due to the browser.

Netscape Communicator appears to not issue the non-standard Request - Range header. When an Acrobat plugin prior to version 3.01 is used with it, it will not properly understand byteranges. The user must upgrade their Acrobat reader to 3.01.



---

The HTTP specifications say that it is legal to merge headers with duplicate names into one (separated by commas). Some browsers that support Cookies don't like merged headers and prefer that each Set-Cookie header is sent separately. When parsing the headers returned by a CGI, Apache will explicitly avoid merging any Set-Cookie headers.



---

Navigator versions 2 through 4 will erroneously re-request GIF89A animations on each loop of the animation if the first response included an Expires header. This happens regardless of how far in the future the expiry time is set. There is no workaround supplied with Apache, however there are hacks for [1.2](#) and for [1.3](#).



---

In certain situations Navigator 3.01 through 3.03 appear to incorrectly issue a POST without the request body. There is no known workaround. It has been fixed in Navigator 3.04, Netscapes provides some [information](#). There's also [some information](#) about the actual problem.



The http client in the JDK1.2beta2 and beta3 will throw away the first part of the response body when both the headers and the first part of the body are sent in the same network packet AND keep-alive's are being used. If either condition is not met then it works fine.

See also Bug-ID's 4124329 and 4125538 at the java developer connection.

If you are seeing this bug yourself, you can add the following BrowserMatch directive to work around it:

```
BrowserMatch "Java1\.2beta[23]" nokeepalive
```

We don't advocate this though since bending over backwards for beta software is usually not a good idea; ideally it gets fixed, new betas or a final release comes out, and no one uses the broken old software anymore. In theory.



## Content-type caching

Navigator (all versions?) will cache the content - type for an object "forever". Using reload or shift-reload will not cause Navigator to notice a content - type change. The only work-around is for the user to flush their caches (memory and disk). By way of an example, some folks may be using an old `mime.types` file which does not map `.htm` to `text/html`, in this case Apache will default to sending `text/plain`. If the user requests the page and it is served as `text/plain`. After the admin fixes the server, the user will have to flush their caches before the object will be shown with the correct `text/html` type.



## 2000

MSIE versions 3.00 and 3.02 (without the Y2K patch) do not handle cookie expiry dates in the year 2000 properly. Years after 2000 and before 2000 work fine. This is fixed in IE4.01 service pack 1, and in the Y2K patch for IE3.02. Users should avoid using expiry dates in the year 2000.



## Lynx incorrectly asking for transparent content negotiation

The Lynx browser versions 2.7 and 2.8 send a "negotiate: trans" header in their requests, which is an indication the browser supports transparent content negotiation (TCN). However the browser does not support TCN. As of version 1.3.4, Apache supports TCN, and this causes problems with these versions of Lynx. As a workaround future versions of Apache will ignore this header when sent by the Lynx client.



MSIE 4.0 does not handle a Vary header properly. The Vary header is generated by mod\_rewrite in apache 1.3. The result is an error from MSIE saying it cannot download the requested file. There are more details in [PR#4118](#).

A workaround is to add the following to your server's configuration files:

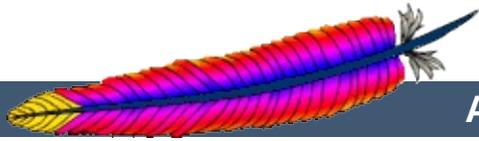
```
BrowserMatch "MSIE 4\.0" force-no-vary
```

(This workaround is only available with releases **after** 1.3.6 of the Apache Web server.)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

## Descriptors and Apache

### Warning:

This document has not been fully updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

A *descriptor*, also commonly called a *file handle* is an object that a program uses to read or write an open file, or open network socket, or a variety of other devices. It is represented by an integer, and you may be familiar with `stdin`, `stdout`, and `stderr` which are descriptors 0, 1, and 2 respectively. Apache needs a descriptor for each log file, plus one for each network socket that it listens on, plus a handful of others. Libraries that Apache uses may also require descriptors. Normal programs don't open up many descriptors at all, and so there are some latent problems that you may experience should you start running Apache with many descriptors (*i.e.*, with many virtual hosts).

The operating system enforces a limit on the number of descriptors that a program can have open at a time. There are typically three limits involved here. One is a kernel limitation, depending on your operating system you will either be able to tune the number of descriptors available to higher numbers (this is frequently called *FD\_SETSIZE*). Or you may be stuck with a (relatively) low amount. The second limit is called the *hard resource* limit, and it is sometimes set by root in an obscure operating system file, but frequently is the same as the kernel limit. The third limit is called the *soft resource* limit. The soft limit is always less than or equal to the hard limit. For example, the hard limit may be 1024, but the soft limit only 64. Any user can raise their soft limit up to the hard limit. Root can raise the hard limit up to the system maximum limit. The soft limit is the actual

limit that is used when enforcing the maximum number of files a process can have open.

To summarize:

```
#open files <= soft limit <= hard limit <= kernel limit
```

You control the hard and soft limits using the `limit` (csh) or `ulimit` (sh) directives. See the respective man pages for more information. For example you can probably use `ulimit -n unlimited` to raise your soft limit up to the hard limit. You should include this command in a shell script which starts your webserver.

Unfortunately, it's not always this simple. As mentioned above, you will probably run into some system limitations that will need to be worked around somehow. Work was done in version 1.2.1 to improve the situation somewhat. Here is a partial list of systems and workarounds (assuming you are using 1.2.1 or later).



Under BSDI 2.0 you can build Apache to support more descriptors by adding `-DFD_SETSIZE=nnn` to `EXTRA_CFLAGS` (where `nnn` is the number of descriptors you wish to support, keep it less than the hard limit). But it will run into trouble if more than approximately 240 Listen directives are used. This may be cured by rebuilding your kernel with a higher `FD_SETSIZE`.



---

Similar to the BSDI 2.0 case, you should define `FD_SETSIZE` and rebuild. But the extra Listen limitation doesn't exist.



By default Linux has a kernel maximum of 256 open descriptors per process. There are several patches available for the 2.0.x series which raise this to 1024 and beyond, and you can find them in the "unofficial patches" section of [the Linux Information HQ](#). None of these patches are perfect, and an entirely different approach is likely to be taken during the 2.1.x development. Applying these patches will raise the FD\_SETSIZE used to compile all programs, and unless you rebuild all your libraries you should avoid running any other program with a soft descriptor limit above 256. As of this writing the patches available for increasing the number of descriptors do not take this into account. On a dedicated webserver you probably won't run into trouble.



Solaris has a kernel hard limit of 1024 (may be lower in earlier versions). But it has a limitation that files using the stdio library cannot have a descriptor above 255. Apache uses the stdio library for the ErrorLog directive. When you have more than approximately 110 virtual hosts (with an error log and an access log each) you will need to build Apache with -  
DHIGH\_SLACK\_LINE=256 added to EXTRA\_CFLAGS. You will be limited to approximately 240 error logs if you do this.



---

AIX version 3.2?? appears to have a hard limit of 128 descriptors.  
End of story. Version 4.1.5 has a hard limit of 2000.



---

Edit the `/etc/conf/cf.d/stune` file or use `/etc/conf/cf.d/configure` choice 7 (User and Group configuration) and modify the `NOFILES` kernel parameter to a suitably higher value. SCO recommends a number between 60 and 11000, the default is 110. Relink and reboot, and the new number of descriptors will be available.



1. Raise `open_max_soft` and `open_max_hard` to 4096 in the `proc` subsystem. Do a `man` on `sysconfig`, `sysconfigdb`, and `sysconfigtab`.
2. Raise `max-vnodes` to a large number which is greater than the number of apache processes \* 4096 (Setting it to 250,000 should be good for most people). Do a `man` on `sysconfig`, `sysconfigdb`, and `sysconfigtab`.
3. If you are using Tru64 5.0, 5.0A, or 5.1, define `NO_SLACK` to work around a bug in the OS. `CFLAGS="-DNO_SLACK"`  
`./configure`



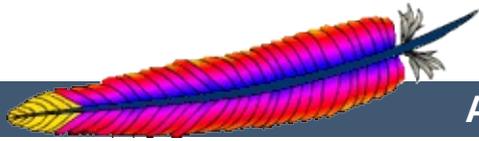
If you have details on another operating system, please submit it through our [Bug Report Page](#).

In addition to the problems described above there are problems with many libraries that Apache uses. The most common example is the bind DNS resolver library that is used by pretty much every unix, which fails if it ends up with a descriptor above 256. We suspect there are other libraries that similar limitations. So the code as of 1.2.1 takes a defensive stance and tries to save descriptors less than 16 for use while processing each request. This is called the *low slack line*.

Note that this shouldn't waste descriptors. If you really are pushing the limits and Apache can't get a descriptor above 16 when it wants it, it will settle for one below 16.

In extreme situations you may want to lower the low slack line, but you shouldn't ever need to. For example, lowering it can increase the limits 240 described above under Solaris and BSDI 2.0. But you'll play a delicate balancing game with the descriptors needed to serve a request. Should you want to play this game, the compile time parameter is `LOW_SLACK_LINE` and there's a tiny bit of documentation in the header file `ht tpd . h`.

Finally, if you suspect that all this slack stuff is causing you problems, you can disable it. Add `-DNO_SLACK` to `EXTRA_CFLAGS` and rebuild. But please report it to our [Bug Report Page](#) so that we can investigate.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Miscellaneous Documentation](#)

## Relevant Standards

This page documents all the relevant standards that the Apache HTTP Server follows, along with brief descriptions.

In addition to the information listed below, the following resources should be consulted:

- <http://purl.org/NET/http-errata> - HTTP/1.1 Specification Errata
- <http://www.rfc-editor.org/errata.html> - RFC Errata
- <http://ftp.ics.uci.edu/pub/ietf/http/#RFC> - A pre-compiled list of HTTP related RFCs

### Notice

This document is not yet complete.



Regardless of what modules are compiled and used, Apache as a basic web server complies with the following IETF recommendations:

**[RFC 1945](#) (Informational)**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. This documents HTTP/1.0.

**[RFC 2616](#) (Standards Track)**

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. This documents HTTP/1.1.

**[RFC 2396](#) (Standards Track)**

A Uniform Resource Identifier (URI) is a compact string of characters for identifying an abstract or physical resource.



---

Regarding the Hypertext Markup Language, Apache complies with the following IETF and W3C recommendations:

#### **[RFC 2854 \(Informational\)](#)**

This document summarizes the history of HTML development, and defines the "text/html" MIME type by pointing to the relevant W3C recommendations.

#### **[HTML 4.01 Specification \(Errata\)](#)**

This specification defines the HyperText Markup Language (HTML), the publishing language of the World Wide Web. This specification defines HTML 4.01, which is a subversion of HTML 4.

#### **[HTML 3.2 Reference Specification](#)**

The HyperText Markup Language (HTML) is a simple markup language used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents.

#### **[XHTML 1.1 - Module-based XHTML \(Errata\)](#)**

This Recommendation defines a new XHTML document type that is based upon the module framework and modules defined in Modularization of XHTML.

#### **[XHTML 1.0 The Extensible HyperText Markup Language \(Second Edition\) \(Errata\)](#)**

This specification defines the Second Edition of XHTML 1.0, a reformulation of HTML 4 as an XML 1.0 application, and three DTDs corresponding to the ones defined by HTML 4.



Authentication

---

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](#)**

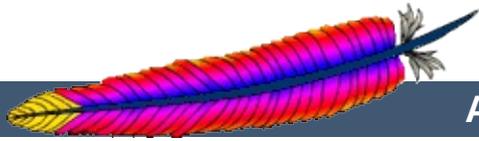
These pages document the country names (official short names in English) in alphabetical order as given in ISO 3166-1 and the corresponding ISO 3166-1-alpha-2 code elements.

### **[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.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Terms Used to Describe Modules

This document describes the terms that are used to describe each Apache [module](#).



## Description

A brief description of the purpose of the module.



---

This indicates how tightly bound into the Apache Web server the module is; in other words, you may need to recompile the server in order to gain access to the module and its functionality. Possible values for this attribute are:

### **MPM**

A module with status "MPM" is a [Multi-Processing Module](#). Unlike the other types of modules, Apache must have one and only one MPM in use at any time. This type of module is responsible for basic request handling and dispatching.

### **Base**

A module labeled as having "Base" status is compiled and loaded into the server by default, and is therefore normally available unless you have taken steps to remove the module from your configuration.

### **Extension**

A module with "Extension" status is not normally compiled and loaded into the server. To enable the module and its functionality, you may need to change the server build configuration files and re-compile Apache.

### **Experimental**

"Experimental" status indicates that the module is available as part of the Apache kit, but you are on your own if you try to use it. The module is being documented for completeness, and is not necessarily supported.

### **External**

Modules which are not included with the base Apache distribution ("third-party modules") may use the "External" status. We are not responsible for, nor do we support such modules.



**SOURCE FILE**

This quite simply lists the name of the source file which contains the code for the module. This is also the name used by the <IfModule> directive.



---

This is a string which identifies the module for use in the `LoadModule` directive when dynamically loading modules. In particular, it is the name of the external variable of type module in the source file.



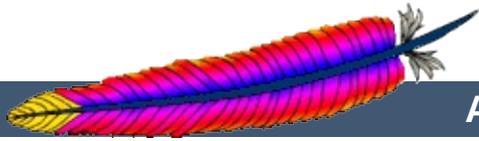
## Compatibility

If the module was not part of the original Apache version 2 distribution, the version in which it was introduced should be listed here. In addition, if the module is limited to particular platforms, the details will be listed here.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Terms Used to Describe Directives

This document describes the terms that are used to describe each Apache [configuration directive](#).

### See also

[Configuration files](#)



## Description

A brief description of the purpose of the directive.



This indicates the format of the directive as it would appear in a configuration file. This syntax is extremely directive-specific, and is described in detail in the directive's definition. Generally, the directive name is followed by a series of one or more space-separated arguments. If an argument contains a space, the argument must be enclosed in double quotes. Optional arguments are enclosed in square brackets. Where an argument can take on more than one possible value, the possible values are separated by vertical bars "|". Literal text is presented in the default font, while argument-types for which substitution is necessary are *emphasized*. Directives which can take a variable number of arguments will end in "..." indicating that the last argument is repeated.

Directives use a great number of different argument types. A few common ones are defined below.

### **URL**

A complete Uniform Resource Locator including a scheme, hostname, and optional pathname as in  
`http://www.example.com/path/to/file.html`

### **URL-path**

The part of a *url* which follows the scheme and hostname as in `/path/to/file.html`. The *url-path* represents a web-view of a resource, as opposed to a file-system view.

### **file-path**

The path to a file in the local file-system beginning with the root directory as in

`/usr/local/apache/htdocs/path/to/file.html`.

Unless otherwise specified, a *file-path* which does not begin with a slash will be treated as relative to the [ServerRoot](#).

### **directory-path**

The path to a directory in the local file-system beginning with the root directory as in

`/usr/local/apache/htdocs/path/to/`.

***filename***

The name of a file with no accompanying path information as in `file.html`.

***regex***

A Perl-compatible [regular expression](#). The directive definition will specify what the *regex* is matching against.

***extension***

In general, this is the part of the *filename* which follows the last dot. However, Apache recognizes multiple filename extensions, so if a *filename* contains more than one dot, each dot-separated part of the filename following the first dot is an *extension*. For example, the *filename* `file.html.en` contains two extensions: `.html` and `.en`. For Apache directives, you may specify *extensions* with or without the leading dot. In addition, *extensions* are not case sensitive.

***MIME-type***

A method of describing the format of a file which consists of a major format type and a minor format type, separated by a slash as in `text/html`.

***env-variable***

The name of an [environment variable](#) defined in the Apache configuration process. Note this is not necessarily the same as an operating system environment variable. See the [environment variable documentation](#) for more details.



**Default**

If the directive has a default value (*i.e.*, if you omit it from your configuration entirely, the Apache Web server will behave as though you set it to a particular value), it is described here. If there is no default value, this section should say "*None*". Note that the default listed here is not necessarily the same as the value the directive takes in the default `httpd.conf` distributed with the server.



This indicates where in the server's configuration files the directive is legal. It's a comma-separated list of one or more of the following values:

### **server config**

This means that the directive may be used in the server configuration files (e.g., `httpd.conf`), but **not** within any `<VirtualHost>` or `<Directory>` containers. It is not allowed in `.htaccess` files at all.

### **virtual host**

This context means that the directive may appear inside `<VirtualHost>` containers in the server configuration files.

### **directory**

A directive marked as being valid in this context may be used inside `<Directory>`, `<Location>`, and `<Files>` containers in the server configuration files, subject to the restrictions outlined in [How Directory, Location and Files sections work](#).

### **.htaccess**

If a directive is valid in this context, it means that it can appear inside *per-directory* `.htaccess` files. It may not be processed, though depending upon the [overrides](#) currently active.

The directive is *only* allowed within the designated context; if you try to use it elsewhere, you'll get a configuration error that will either prevent the server from handling requests in that context correctly, or will keep the server from operating at all -- *i.e.*, the server won't even start.

The valid locations for the directive are actually the result of a Boolean OR of all of the listed contexts. In other words, a directive

that is marked as being valid in "server config, .htaccess" can be used in the httpd.conf file and in .htaccess files, but not within any <Directory> or <VirtualHost> containers.



---

This directive attribute indicates which configuration override must be active in order for the directive to be processed when it appears in a `.htaccess` file. If the directive's `context` doesn't permit it to appear in `.htaccess` files, then no context will be listed.

Overrides are activated by the `AllowOverride` directive, and apply to a particular scope (such as a directory) and all descendants, unless further modified by other `AllowOverride` directives at lower levels. The documentation for that directive also lists the possible override names available.



---

This indicates how tightly bound into the Apache Web server the directive is; in other words, you may need to recompile the server with an enhanced set of modules in order to gain access to the directive and its functionality. Possible values for this attribute are:

### **Core**

If a directive is listed as having "Core" status, that means it is part of the innermost portions of the Apache Web server, and is always available.

### **MPM**

A directive labeled as having "MPM" status is provided by a [Multi-Processing Module](#). This type of directive will be available if and only if you are using one of the MPMs listed on the [Module](#) line of the directive definition.

### **Base**

A directive labeled as having "Base" status is supported by one of the standard Apache modules which is compiled into the server by default, and is therefore normally available unless you've taken steps to remove the module from your configuration.

### **Extension**

A directive with "Extension" status is provided by one of the modules included with the Apache server kit, but the module isn't normally compiled into the server. To enable the directive and its functionality, you will need to change the server build configuration files and re-compile Apache.

### **Experimental**

"Experimental" status indicates that the directive is available as part of the Apache kit, but you're on your own if you try to use it. The directive is being documented for completeness, and is not necessarily supported. The module which provides the directive may or may not be compiled in by default; check

the top of the page which describes the directive and its module to see if it remarks on the availability.



---

This quite simply lists the name of the source module which defines the directive.

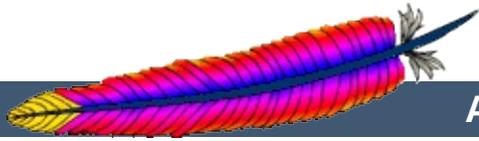


If the directive wasn't part of the original Apache version 2 distribution, the version in which it was introduced should be listed here. In addition, if the directive is available only on certain platforms, it will be noted here.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

## Apache-Kernfunktionen

Diese Übersetzung ist möglicherweise nicht mehr aktuell. Bitte prüfen Sie die englische Version auf die neuesten Änderungen.

**Beschreibung:** Ständig verfügbare Kernfunktionen des Apache HTTP Servers

**Status:** Core



<b><u>Beschreibung:</u></b>	Ressourcen lassen angehängte Pfadangaben zu
<b><u>Syntax:</u></b>	AcceptPathInfo On Off Default
<b><u>Voreinstellung:</u></b>	AcceptPathInfo Default
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Verfügbar ab Apache 2.0.30

Die Direktive steuert, ob Anfragen akzeptiert oder abgewiesen werden, bei denen nach der tatsächlichen Datei (oder einer nicht existierenden Datei in einem existierenden Verzeichnis) zusätzliche Pfadangaben folgen. Die angehängte Pfadangabe kann Skripten in der Umgebungsvariable `PATH_INFO` verfügbar gemacht werden.

Nehmen wir beispielsweise an, dass `/test/` auf ein Verzeichnis zeigt, welches lediglich eine Datei `here.html` enthält. Dann wird bei Anfragen nach `/test/here.html/more` und `/test/nothere.html/more` beides Mal `/more` als `PATH_INFO` ermittelt.

Die drei möglichen Argumente für die Direktive `AcceptPathInfo` sind:

### **off**

Eine Anfrage wird nur dann akzeptiert, wenn sie exakt auf ein existierendes Verzeichnis (oder eine Datei) abgebildet werden kann. Daher würde eine Anfrage mit einer nach dem tatsächlichen Dateinamen angehängten Pfadangabe, wie

/test/here.html/more im obigen Beispiel, den Fehler 404 NOT FOUND (*Anm.d.Ü.:* nicht gefunden) zurückgeben.

## On

Eine Anfrage wird akzeptiert, wenn eine vorangestellte Pfadangabe auf ein existierendes Verzeichnis abgebildet werden kann. Das obige Beispiel /test/here.html/more wird akzeptiert, wenn /test/here.html auf eine gültige Datei zeigt.

## Default

Die Behandlung von Anfragen mit angehängten Pfadangaben wird von dem für die Anfrage verantwortlichen [Handler](#) bestimmt. Der Core-Handler für gewöhnliche Dateien weist PATH\_INFO-Zugriffe standardmäßig zurück. Handler, die Skripte bedienen, wie z.B. [cgi-script](#) und [isapi-handler](#), sind im Allgemeinen darauf voreingestellt, PATH\_INFO zu akzeptieren.

Das eigentliche Ziel von AcceptPathInfo ist es, Ihnen das Überschreiben der Voreinstellung der Handler bezüglich der Akzeptanz oder Ablehnung von PATH\_INFO zu erlauben. Eine solche Änderung ist zum Beispiel notwendig, wenn Sie einen [Filter](#) wie [INCLUDES](#) verwenden, um Inhalte abhängig von PATH\_INFO zu generieren. Der Core-Handler würde die Anfrage normalerweise abweisen. Verwenden Sie die folgende Konfiguration, um dennoch solch ein Skript zu ermöglichen.

```
<Files "mypaths.shtml">
  Options +Includes
  SetOutputFilter INCLUDES
  AcceptPathInfo On
</Files>
```



<b>Beschreibung:</b>	Name der dezentralen Konfigurationsdateien
<b>Syntax:</b>	AccessFileName <i>Dateiname</i> [ <i>Dateiname</i> ] ...
<b>Voreinstellung:</b>	AccessFileName .htaccess
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

Aus dieser Namensliste sucht der Server während der Bearbeitung einer Anfrage in jedem Verzeichnis nach der ersten existierenden Datei, sofern im betreffenden Verzeichnis dezentrale Konfigurationsdateien [erlaubt sind](#). Beispiel:

```
AccessFileName .acl
```

Vor der Rücksendung des Dokuments /usr/local/web/index.html wird der Server /.acl, /usr/.acl, /usr/local/.acl und /usr/local/web/.acl einlesen, solange diese nicht mit

```
<Directory />
  AllowOverride None
</Directory>
```

deaktiviert wurden.

## Siehe auch

- [AllowOverride](#)
- [Konfigurationsdateien](#)
- [.htaccess-Dateien](#)



<b><u>Beschreibung:</u></b>	Standard-Charset-Parameter, der bei Antworten vom Content-Type text/plain oder text/html hinzugefügt wird
<b><u>Syntax:</u></b>	AddDefaultCharset On Off  <i>Zeichenkodierung</i>
<b><u>Voreinstellung:</u></b>	AddDefaultCharset Off
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive gibt einen Standardwert für den Charset-Parameter des Medientyps (den Namen einer Zeichencodierung) an, der einer Antwort genau dann hinzugefügt wird, wenn der Content-Type der Antwort entweder text/plain oder text/html ist. Dies sollte jedes mittels META-Element im Datenteil der Antwort angegebene Charset überschreiben. Das genaue Verhalten hängt jedoch oft von der Client-Konfiguration des Benutzers ab. Die Einstellung AddDefaultCharset Off deaktiviert diese Funktionalität. AddDefaultCharset On aktiviert die Standard-Zeichencodierung iso-8859-1. Jeder andere Wert wird als die zu verwendende *Zeichencodierung* aufgefaßt, die eines der bei [IANA registrierten Charset-Werte](#) zur Verwendung in MIME-Medientypen sein sollte. Zum Beispiel:

```
AddDefaultCharset utf-8
```

**AddDefaultCharset** sollte nur verwendet werden, wenn von allen Textressourcen, für die es gilt, bekannt ist, dass sie in dieser Zeichencodierung vorliegen, oder wenn es zu unbequem ist, ihre Zeichencodierung individuell zu benennen. Ein solches Beispiel ist

das Hinzufügen des Charset-Parameters zu Ressourcen, die generierte Inhalte enthalten. Ein Beispiel sind CGI-Skript-Altlasten, die aufgrund von in die Ausgabe integrierten Daten, die durch den Benutzer übermittelt wurden, gegen Cross-Site-Scripting-Angriffe verwundbar sind. Eine bessere Lösung wäre jedoch, diese Skripte zu korrigieren (oder zu löschen), da die Angabe einer Standard-Zeichencodierung keine Anwender schützt, die in ihrem Browser die Funktion zur automatischen Erkennung der Zeichenkodierung aktiviert haben.

## Siehe auch

- [AddCharset](#)



## AddOutputFilterByType-Direktive

<b>Beschreibung:</b>	einen Ausgabefilter einem bestimmten MIME-Type zuordnen
<b>Syntax:</b>	AddOutputFilterByType <i>Filter</i> [; <i>Filter</i> ...] <i>MIME-Type</i> [ <i>MIME-Type</i> ] ...
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	FileInfo
<b>Status:</b>	Core
<b>Modul:</b>	core
<b>Kompatibilität:</b>	Verfügbar ab Apache 2.0.33

Die Direktive aktiviert für eine Anfrage abhängig vom MIME-Type der Antwort einen bestimmten Ausgabe-[Filter](#).

Das folgende Beispiel verwendet den Filter DEFLATE, der von [mod\\_deflate](#) angeboten wird. Er komprimiert jede Ausgabe, die als text/html oder text/plain gekennzeichnet ist, (gleichgültig, ob statisch oder dynamisch) bevor sie an den Client gesendet wird.

```
AddOutputFilterByType DEFLATE text/html text/plain
```

Wenn Sie den Inhalt von mehr als einem Filter verarbeiten lassen wollen, dann müssen deren Namen durch Semikolons voneinander getrennt werden. Es ist ebenfalls möglich, eine [AddOutputFilterByType](#)-Direktive für jeden von diesen Filtern zu verwenden.

Die folgende Konfiguration sorgt dafür, dass alle Skriptausgaben, die als text/html gekennzeichnet sind, zuerst vom INCLUDES-Filter und dann vom DEFLATE-Filter verarbeitet werden.

```
<Location /cgi-bin/>
  Options Includes
  AddOutputFilterByType INCLUDES;DEFLATE text/html
</Location>
```

### Hinweis:

Die Aktivierung von Filtern mittels [AddOutputFilterByType](#) kann in einigen Fällen ganz oder teilweise fehlschlagen. Beispielsweise werden keine Filter angewendet, wenn der MIME-Type nicht bestimmt werden kann und auf die Einstellung der [DefaultType](#)-Anweisung zurückfällt, selbst wenn die [DefaultType](#)-Einstellung die gleiche ist.

Wenn Sie jedoch sicherstellen wollen, dass der Filter angewendet wird, sollten Sie den Content-Type z.B. mit [AddType](#) oder [ForceType](#) der Ressource explizit zuordnen. Das Setzen des Content-Types innerhalb eines (nicht-nph) CGI-Skriptes funktioniert ebenfalls zuverlässig.

Die Typ-gebundenen Ausgabefilter werden niemals auf Proxy-Anfragen angewendet.

### Siehe auch

- [AddOutputFilter](#)
- [SetOutputFilter](#)
- [Filter](#)



<b><u>Beschreibung:</u></b>	Legt fest, ob kodierte Pfadtrennzeichen in URLs durchgereicht werden dürfen
<b><u>Syntax:</u></b>	AllowEncodedSlashes On Off
<b><u>Voreinstellung:</u></b>	AllowEncodedSlashes Off
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Verfügbar ab Apache 2.0.46

Die `AllowEncodedSlashes`-Direktive erlaubt die Verwendung von URLs, welche kodierte Pfadtrennzeichen (%2F für / und auf entsprechenden Systemen zusätzlich %5C für \) enthalten. Normalerweise werden derartige URLs mit einem 404-Fehler (Nicht gefunden) abgewiesen.

`AllowEncodedSlashes On` ist vor allem in Verbindung mit `PATH_INFO` hilfreich.

### Anmerkung

Das Erlauben von Schrägstrichen impliziert *nicht* deren *Dekodierung*. Vorkommen von %2F oder %5C (*nur* auf entsprechenden Systemen) werden unverändert in der ansonsten dekodierten URL belassen.

### Siehe auch

- [AcceptPathInfo](#)



<b><u>Beschreibung:</u></b>	Direktiven-Typen, die in <code>.htaccess</code> -Dateien erlaubt sind.
<b><u>Syntax:</u></b>	<code>AllowOverride All None Direktiven-Typ [Direktiven-Typ] ...</code>
<b><u>Voreinstellung:</u></b>	<code>AllowOverride All</code>
<b><u>Kontext:</u></b>	Verzeichnis
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Wenn der Server eine `.htaccess`-Datei (wie durch `AccessFileName` definiert) findet, muss er wissen, welche in der Datei angegebenen Direktiven frühere Konfigurationsanweisungen überschreiben dürfen.

#### Nun in `<Directory>`-Abschnitten verfügbar

`AllowOverride` ist nur in `<Directory>`-Abschnitten gültig, die ohne reguläre Ausdrücke definiert wurden, nicht in `<Location>`-, `<DirectoryMatch>`- oder `<Files>`-Abschnitten.

Wenn diese Anweisung auf `None` gesetzt wird, dann werden `.htaccess`-Dateien komplett ignoriert. In diesem Fall wird der Server nicht einmal versuchen, die `.htaccess`-Dateien im Dateisystem zu lesen.

Wenn diese Anweisung auf `All` gesetzt wird, dann ist jede Direktive in den `.htaccess`-Dateien erlaubt, die den `Kontext` `.htaccess` besitzt.

Der `Direktiven-Typ` kann eine der folgenden Anweisungsgruppen sein.

## AuthConfig

Erlaubt die Verwendung von Autorisierungs-Anweisungen ([AuthDBMGroupFile](#), [AuthDBMUserFile](#), [AuthGroupFile](#), [AuthName](#), [AuthType](#), [AuthUserFile](#), [Require](#) usw.).

## FileInfo

Erlaubt die Verwendung von Direktiven zur Steuerung der Dokumenttypen ([DefaultType](#), [ErrorDocument](#), [ForceType](#), [LanguagePriority](#), [SetHandler](#), [SetInputFilter](#), [SetOutputFilter](#), und `mod_mime`-Direktiven `Add*` und `Remove*` usw.).

## Indexes

Erlaubt die Verwendung von Direktiven zur Steuerung von Verzeichnisindizes ([AddDescription](#), [AddIcon](#), [AddIconByEncoding](#), [AddIconByType](#), [DefaultIcon](#), [DirectoryIndex](#), [FancyIndexing](#), [HeaderName](#), [IndexIgnore](#), [IndexOptions](#), [ReadmeName](#) usw.).

## Limit

Erlaubt die Verwendung von Direktiven zur Steuerung des Zugriffs von Hosts ([Allow](#), [Deny](#) und [Order](#)).

## Options

Erlaubt die Verwendung von Direktiven zur Steuerung spezieller Verzeichniseigenschaften ([Options](#) und [XBitHack](#)).

Beispiel:

```
AllowOverride AuthConfig Indexes
```

Im obigen Beispiel erzeugen alle Direktiven einen internal server error (*Anm.d.Ü.:* (Server-interner Fehler)), die weder der Gruppe AuthConfig noch der Gruppe Indexes angehören.

## Siehe auch

- [AccessFileName](#)
- [Konfigurationsdateien](#)
- [.htaccess-Dateien](#)



<b><u>Beschreibung:</u></b>	Autorisierungsbereich zur Verwendung in der HTTP-Authentisierung
<b><u>Syntax:</u></b>	AuthName <i>auth-Bereich</i>
<b><u>Kontext:</u></b>	Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	AuthConfig
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive legt den Namen des Autorisierungsbereiches (*Anm.d.Ü.:* Der Autorisierungsbereich wird auch Realm genannt.) für ein Verzeichnis fest. Dieser Realm wird dem Client mitgeteilt, damit der Anwender weiß, welchen Benutzernamen und welches Passwort er zu übermitteln hat. **AuthName** akzeptiert ein Argument. Falls der Name des Realm Leerzeichen enthält, muss er in Anführungszeichen eingeschlossen werden. Um zu funktionieren, muss die Anweisung von den Direktiven **AuthType** und **Require** sowie von Direktiven wie **AuthUserFile** und **AuthGroupFile** begleitet werden.

Beispiel:

```
AuthName "Top Secret"
```

Die AuthName übergebene Zeichenkette ist das, was in dem von den meisten Browsern angebotenen Passwort-Dialog angezeigt wird.

## Siehe auch

- [Authentisierung, Autorisierung und Zugriffskontrolle](#)



<b>Beschreibung:</b>	Art der Authentisierung
<b>Syntax:</b>	AuthType Basic Digest
<b>Kontext:</b>	Verzeichnis, .htaccess
<b>AllowOverride:</b>	AuthConfig
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive wählt die Art der Benutzer-Authentisierung für ein Verzeichnis aus. Derzeit sind lediglich Basic und Digest implementiert. Um zu funktionieren, muss die Anweisung von den Direktiven [AuthName](#) und [Require](#) sowie von Direktiven wie [AuthUserFile](#) und [AuthGroupFile](#) begleitet werden.

### Siehe auch

- [Authentisierung, Autorisierung und Zugriffskontrolle](#)



<b><u>Beschreibung:</u></b>	Technik zur Bestimmung des Interpreters für CGI-Skripte
<b><u>Syntax:</u></b>	CGIMapExtension <i>CGI-Pfad</i> <i>.Endung</i>
<b><u>Kontext:</u></b>	Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	ausschließlich NetWare

Die Direktive wird zur Steuerung verwendet, wie Apache den Interpreter ermittelt, der zur Ausführung von CGI-Skripten verwendet wird. Beispielsweise bestimmt die Angabe von `CGIMapExtension sys:\foo.nlm .foo`, dass alle CGI-Skripte mit der Endung `.foo` an den FOO-Interpreter übergeben werden.



<b>Beschreibung:</b>	Aktiviert die Generierung von Content -MD5 HTTP-Response-Headern
<b>Syntax:</b>	ContentDigest On Off
<b>Voreinstellung:</b>	ContentDigest Off
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	Options
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive aktiviert die Generierung von Content -MD5-Headern, wie sie in RFC1864 bzw. RFC2068 definiert sind.

MD5 ist ein Algorithmus zur Berechnung eines "Datenextrakts" (zuweilen "Fingerabdruck" genannt) (*Anm.d.Ü.:* Der "Datenextrakt" wird im Englischen als "message digest" oder "fingerprint" bezeichnet.) aus beliebig langen Daten. Es gilt als zuverlässig, dass Veränderungen an den Daten sich in Veränderungen des Extrakts widerspiegeln.

Der Content -MD5-Header bietet eine End-to-End-Integritätsprüfung (MIC) (*Anm.d.Ü.:* MIC steht für "message integrity check".) des Daten-Inhalts. Ein Proxy oder Client kann diesen Header prüfen, um zufällige Veränderungen des Entity-Inhalts bei der Übertragung festzustellen. Beispielheader:

```
Content-MD5: AuLb7Dp1rqtRtxz2m9kRpA==
```

Beachten Sie bitte, dass dies Performanceprobleme auf Ihrem System verursachen kann, da der Extrakt bei jeder Anfrage berechnet wird (der Wert wird nicht zwischengespeichert).

Content -MD5 wird nur für Dokumente gesendet, die von [core](#)

bedient werden, nicht jedoch bei Modulen. SSI-Dokumente, CGI-Skript-Ausgaben und Byte-Range-Antworten besitzen diesen Header beispielsweise nicht.



## DefaultType Directive

<b><u>Beschreibung:</u></b>	MIME-Content-Type, der gesendet wird, wenn der Server den Typ nicht auf andere Weise ermitteln kann.
<b><u>Syntax:</u></b>	DefaultType <i>MIME-Type</i>
<b><u>Voreinstellung:</u></b>	DefaultType text/plain
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Es kann vorkommen, dass der Server ein Dokument ausliefern muss, dessen Typ er nicht mit Hilfe seiner MIME-Type-Zuordnungen bestimmen kann.

Der Server muss den Client über den Content-Type des Dokumentes informieren. Daher verwendet er im Falle eines unbekanntens Typs die DefaultType-Einstellung. Zum Beispiel:

```
DefaultType image/gif
```

wäre angemessen für ein Verzeichnis, das viele GIF-Bilder enthält, deren Dateinamen nicht Endung `.gif` besitzen.

Beachten Sie bitte, dass die Direktive anders als [ForceType](#) lediglich den Standard-MIME-Type bestimmt. Alle anderen MIME-Type-Definitionen, einschließlich Dateierweiterungen, die den Medien-Typ anzeigen können, überschreiben diese Voreinstellung.



## Directory Directive

<b>Beschreibung:</b>	Umschließt eine Gruppe von Direktiven, die nur auf das genannte Verzeichnis des Dateisystems und Unterverzeichnisse angewendet werden
<b>Syntax:</b>	<code>&lt;Directory Verzeichnispfad&gt; ... &lt;/Directory&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

`<Directory>` und `</Directory>` werden dazu verwendet, eine Gruppe von Direktiven zusammenzufassen, die nur für das genannte Verzeichnis und dessen Unterverzeichnisse gelten. Jede Direktive, die im Verzeichnis-Kontext erlaubt ist, kann verwendet werden. *Verzeichnispfad* ist entweder der vollständige Pfad zu einem Verzeichnis oder eine Zeichenkette mit Platzhaltern wie sie von der Unix-Shell zum Abgleich verwendet werden. In einer Zeichenkette mit Platzhaltern (*Anm.d.Ü.: sogenannte wild-cards*) entspricht `?` einem einzelnen Zeichen und `*` einer Zeichenkette beliebiger Länge. Sie können auch auch `[]`-Zeichenbereiche verwenden. Keiner der Platzhalter entspricht dem Zeichen `/`. Daher passt `<Directory /*/public_html>` nicht auf `/home/user/public_html`, `<Directory /home/*/public_html>` jedoch tut es. Beispiel:

```
<Directory /usr/local/httpd/htdocs>  
  Options Indexes FollowSymLinks  
</Directory>
```

Seien Sie vorsichtig mit den *Verzeichnispfad*-Argumenten. Sie müssen buchstäblich mit dem Dateisystempfad übereinstimmen, den der Apache für den Zugriff auf die Dateien verwendet. Direktiven, die für ein bestimmtes Verzeichnis

gelten, gelten nicht für Dateien in dem Verzeichnis, auf die über einen anderen Pfad zugegriffen wird, wie z.B. über verschiedene symbolische Links.

Erweiterte reguläre Ausdrücke können ebenfalls verwendet werden, indem das Zeichen ~ hinzugefügt wird. Beispielsweise würde

```
<Directory ~ "^/www/.*/[0-9]{3}">
```

auf Verzeichnisse in /www/ passen, die aus drei Zahlen bestehen.

Wenn mehrere `<Directory>`-Abschnitte (ohne reguläre Ausdrücke) auf ein Verzeichnis (oder ein ihm übergeordnetes Verzeichnis) passen, welches ein Dokument enthält, dann werden die Direktiven der Reihe nach, angefangen beim kürzesten passenden Muster, vermischt mit den Direktiven aus den [.htaccess](#)-Dateien, angewendet. Beispiel:

```
<Directory />
  AllowOverride None
</Directory>

<Directory /home/>
  AllowOverride FileInfo
</Directory>
```

Beim Zugriff auf das Dokument /home/web/dir/doc.html sind die einzelnen Schritte:

- Wende die Direktive AllowOverride None an (deaktiviere .htaccess-Dateien).
- Wende die Direktive AllowOverride FileInfo (auf das Verzeichnis /home) an.
- Wende jede FileInfo-Direktive aus /home/.htaccess, /home/web/.htaccess und /home/web/dir/.htaccess

der Reihe nach an.

Reguläre Ausdrücke werden solange nicht berücksichtigt, bis alle normalen Abschnitte angewendet wurden. Anschließend werden alle regulären Ausdrücke in der Reihenfolge geprüft, in der sie in der Konfigurationsdatei auftauchen. Beispielsweise wird bei

```
<Directory ~ abc$>  
  # ... hier die Direktiven ...  
</Directory>
```

der Abschnitt mit dem regulären Ausdruck nicht berücksichtigt, bis alle normalen `<Directory>`-Abschnitte und `.htaccess`-Dateien angewendet wurden. Dann erst wird der reguläre Ausdruck mit `/home/abc/public_html/abc` abgeglichen und der entsprechende `<Directory>`-Abschnitt angewendet.

**Beachten Sie bitte, dass der vom Apache voreingestellte Zugriff für `<Directory /> Allow from All` ist. Das bedeutet, dass der Apache jede Datei ausliefert, die durch eine URL abgebildet wird. Es wird empfohlen, dass Sie dies durch einen Block wie**

```
<Directory />  
  Order Deny,Allow  
  Deny from All  
</Directory>
```

**ändern und anschließend für Verzeichnisse überschreiben, die Sie verfügbar machen *wollen*. Für weitere Einzelheiten lesen Sie bitte die Seite zu den [Sicherheitshinweisen](#).**

Die Verzeichnisabschnitte erscheinen in der Datei `httpd.conf`. `<Directory>`-Direktiven dürfen nicht ineinander verschachtelt werden oder innerhalb von `<Limit>`- oder `<LimitExcept>`-Abschnitten auftauchen.

## Siehe auch

- [Wie die Abschnitte <Directory>, <Location> und <Files> arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird



## DirectoryMatch-Direktive

<b>Beschreibung:</b>	Umschließt eine Gruppe von Direktiven, die auf Verzeichnisse des Dateisystems und ihre Unterverzeichnisse abgebildet werden, welche auf einen regulären Ausdruck passen
<b>Syntax:</b>	<code>&lt;DirectoryMatch regex&gt; ...</code> <code>&lt;/DirectoryMatch&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

`<DirectoryMatch>` und `</DirectoryMatch>` werden dazu verwendet, eine Gruppe von Direktiven zusammenzufassen, die nur für das genannte Verzeichnis und dessen Unterverzeichnisse gelten, genauso wie bei `<Directory>`. Als Argument dient jedoch ein regulärer Ausdruck. Beispielsweise würde

```
<DirectoryMatch "^/www/(.+)?[0-9]{3}">
```

auf Verzeichnisse in `/www/` passen, die aus drei Zeichen bestehen.

### Siehe auch

- `<Directory>` für eine Beschreibung, wie reguläre Ausdrücke mit normalen `<Directory>`-Anweisungen vermischt werden.
- [Wie die Abschnitte <Directory>, <Location> und <Files> arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird



<b>Beschreibung:</b>	Verzeichnis, welches den Haupt-Dokumentenbaum bildet, der im Web sichtbar ist.
<b>Syntax:</b>	DocumentRoot <i>Verzeichnis</i>
<b>Voreinstellung:</b>	DocumentRoot /usr/local/apache/htdocs
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive setzt das Verzeichnis, von dem aus [httpd](#) Dateien ausliefert. Sofern nicht eine Direktive wie [Alias](#) greift, hängt der Server Pfade aus der angeforderten URL an das Wurzelverzeichnis an, um den Pfad zum Dokument zu bilden.  
Beispiel:

```
DocumentRoot /usr/web
```

Damit bezieht sich ein Zugriff auf `http://www.my.host.com/index.html` auf `/usr/web/index.html`.

**DocumentRoot** sollte ohne einen Schrägstrich am Ende angegeben werden.

## Siehe auch

- [URLs auf das Dateisystem abbilden](#)



<b><u>Beschreibung:</u></b>	Verwende Memory-Mapping, um Dateien während der Auslieferung zu lesen
<b><u>Syntax:</u></b>	EnableMMAP On Off
<b><u>Voreinstellung:</u></b>	EnableMMAP On
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive steuert, ob [httpd](#) Memory-Mapping verwenden darf, wenn er während der Auslieferung den Inhalt einer Datei lesen muss. Wenn die Bearbeitung einer Anfrage es erfordert, auf die Daten in einer Datei zuzugreifen -- zum Beispiel bei der Auslieferung einer mittels `mod_include` serverseitig analysierten Datei --, dann verwendet der Apache standardmäßig Memory-Mapping für diese Datei, sofern das Betriebssystem es unterstützt.

Memory-Mapping bedeutet zuweilen eine Performanceverbesserung. In einigen Umgebungen ist es jedoch besser, Memory-Mapping zu deaktivieren, um Problemen während des Betriebs vorzubeugen:

- Bei einigen Multiprozessorsystemen kann Memory-Mapping die Performance von [httpd](#) reduzieren.
- Bei einem per NFS eingebundenen `DocumentRoot` kann [httpd](#) mit einem Speicherzugriffsfehler (*Anm.d.Ü.:* ein so genannter "segmentation fault") abstürzen, wenn eine Datei gelöscht oder gekürzt wird, während [httpd](#) sie im Speicher abbildet.

Bei Serverkonfigurationen, die für dieses Problem anfällig sind, sollten Sie das Memory-Mapping für auszuliefernde Dateien

deaktivieren, indem Sie schreiben:

```
EnableMMAP Off
```

Bei per NFS eingebundenen Dateien kann diese Funktion explizit für die störenden Dateien deaktiviert werden, indem Sie angeben:

```
<Directory "/pfad-zu-den-nfs-dateien">  
  EnableMMAP Off  
</Directory>
```



<b><u>Beschreibung:</u></b>	Verwende die sendfile-Unterstützung des Kernels, um Dateien an den Client auszuliefern
<b><u>Syntax:</u></b>	EnableSendfile On Off
<b><u>Voreinstellung:</u></b>	EnableSendfile On
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Verfügbar ab Apache Version 2.0.44

Die Direktive steuert, ob [httpd](#) die sendfile-Unterstützung des Kernels verwenden kann, um Dateiinhalte an den Client zu übermitteln. Wenn die Bearbeitung einer Anfrage keinen Zugriff auf die Daten in der Datei erfordert -- zum Beispiel bei der Auslieferung einer statischen Datei -- und das Betriebssystem es unterstützt, verwendet der Apache standardmäßig sendfile, um den Dateiinhalt zu übertragen, ohne die Datei jemals zu lesen.

Der sendfile-Mechanismus vermeidet getrennte Lese- und Sendeoperationen sowie Puffer-Zuweisungen. Bei einigen Plattformen bzw. Dateisystemen deaktivieren Sie diese Funktion jedoch besser, um Probleme während des Betriebs zu vermeiden:

- Einige Plattformen besitzen u.U. eine fehlerhafte sendfile-Unterstützung, die das Erstellungssystem nicht erkennt, insbesondere wenn die Binärdateien auf einem anderen Rechner erstellt und auf eine solche Maschine mit fehlerhafter sendfile-Unterstützung übertragen wurden.
- Bei einem über das Netzwerk eingebundenen [DocumentRoot](#) (z.B. NFS oder SMB) ist der Kernel möglicherweise nicht in der Lage, die Netzwerkdatei über

seinen eigenen Cache zu bedienen.

- Unter Linux löst die Verwendung von `sendfile` in Verbindung mit bestimmten Netzwerkkarten und IPv6 TCP-Checksummenfehler aus.

Bei Serverkonfigurationen, die für dieses Problem anfällig sind, sollten die diese Funktion deaktivieren, indem Sie schreiben:

```
EnableSendfile Off
```

Bei per NFS oder SMB eingebundenen Dateien kann diese Funktion explizit für die störenden Dateien deaktiviert werden, indem Sie angeben:

```
<Directory "/pfad-zu-den-nfs-dateien">  
  EnableSendfile Off  
</Directory>
```



<b><u>Beschreibung:</u></b>	Das, was der Server im Fehlerfall an den Client zurückgibt
<b><u>Syntax:</u></b>	ErrorDocument <i>Fehlercode</i> <i>Dokument</i>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Die Syntax der Anführungszeichen bei Textnachrichten hat sich im Apache 2.0 geändert

Im Falle eines Problems oder Fehlers kann der Apache konfiguriert werden, eine der vier Aktionen auszuführen:

1. Ausgabe einer einfachen, hartkodierten Fehlermeldung
2. Ausgabe einer angepassten Meldung
3. Umleitung zu einem lokalen *URL-Pfad* der das Problem bzw. den Fehler behandelt
4. Umleitung zu einer externen *URL*, die das Problem bzw. den Fehler behandelt

Die erste Option ist Voreinstellung, während die Optionen 2 bis 4 über die Direktive **ErrorDocument** eingestellt werden, welcher der HTTP-Statuscode und eine URL oder Nachricht folgen. Abhängig vom Problem bzw. Fehler bietet der Apache manchmal zusätzliche Informationen an.

URLs können bei lokalen Webpfaden mit einem Schrägstrich (/) beginnen (relativ zum **DocumentRoot**-Verzeichnis) oder eine vollständige URL bilden, die der Client auflösen kann. Alternativ kann eine Nachricht für die Anzeige im Browser angeboten

werden. Beispiel:

```
ErrorDocument 500 http://foo.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/falsche_urls.pl
ErrorDocument 401 /info_zur_anmeldung.html
ErrorDocument 403 "Der Zugriff ist nicht erlaubt."
```

Außerdem kann auch der Spezialwert `default` verwendet werden, um die schlichte, im Apache hartkodierte Nachricht anzugeben. Während es normalerweise nicht benötigt wird, stellt `default` die einfache, hartkodierte Nachricht des Apache bei Konfigurationen wieder her, bei denen andernfalls eine bestehende **ErrorDocument**-Anweisung übernommen würde.

```
ErrorDocument 404 /cgi-bin/bad_urls.pl

<Directory /web/docs>
    ErrorDocument 404 default
</Directory>
```

Wenn Sie eine **ErrorDocument**-Anweisung angeben, die auf eine entfernte URL weist (d.h. irgendetwas mit der Methode `http` davor), beachten Sie bitte, dass der Apache eine Umleitung zum Client sendet, um diesem mitzuteilen, wo das Dokument zu finden ist, auch wenn das Dokument letztlich wieder zum gleichen Server führt. Das hat mehrere Auswirkungen. Die wichtigste ist, dass der Client nicht den Original-Statuscode erhält sondern statt dessen einen Umleitungs-Statuscode. Dies wiederum kann Web-Robots und andere Clients verwirren, die den Statuscode dazu verwenden, herauszufinden ob eine URL gültig ist. Wenn Sie eine entfernte URL in einer Anweisung `ErrorDocument 401` verwenden, wird der Client darüber hinaus nicht wissen, dass er den Benutzer zur Eingabe eines Passwortes auffordern muss, da er den Statuscode 401 nicht erhält. **Deshalb müssen Sie sich auf ein lokales Dokument beziehen, wenn Sie eine Anweisung `ErrorDocument 401` verwenden.**

Der Microsoft Internet Explorer (MSIE) ignoriert standardmäßig serverseitig generierte Fehlermeldungen, wenn sie "zu kurz" sind und ersetzt sie durch eigene "freundliche" Fehlermeldungen. Die Größe variiert abhängig von der Art des Fehlers, im Allgemeinen zeigt der MSIE jedoch den serverseitig generierten Fehler, anstatt ihn zu verstecken, wenn Ihr Fehlerdokument größer als 512 Bytes ist. Weitere Informationen sind im Artikel [Q294807](#) in der Microsoft Knowledgebase verfügbar.

Obwohl die meisten Fehlermeldungen überschrieben werden können, werden unter bestimmten Umständen die internen Meldungen ungeachtet der Einstellung der `ErrorDocument`-Direktive verwendet. Insbesondere bei einer fehlerhaften Anfrage werden der normale Bearbeitungsprozess sofort beendet und die interne Meldung zurückgegeben. Das ist notwendig, um Sicherheitsprobleme zu vermeiden, die auf Grund fehlerhafter Anfragen entstehen.

In Versionen vor 2.0 wurden Meldungen durch ein einzelnes vorangestelltes Anführungszeichen (") erkannt.

## Siehe auch

- [Dokumentation zu individuellen Fehlermeldungen](#)



## ErrorLog Direktive

<b>Beschreibung:</b>	Ablageort, an dem der Server Fehler protokolliert
<b>Syntax:</b>	ErrorLog <i>Dateiname</i>   syslog[: <i>facility</i> ]
<b>Voreinstellung:</b>	ErrorLog logs/error_log (Unix) ErrorLog logs/error.log (Windows and OS/2)
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive **ErrorLog** bestimmt den Namen der Datei, in welcher der Server alle auftretenden Fehler protokolliert werden. Wenn *Dateiname* nicht absolut ist, wird er relativ zu **ServerRoot** betrachtet.

### Beispiel

```
ErrorLog /var/log/httpd/error_log
```

Wenn der *Dateiname* mit einem senkrechten Strich (|, engl.: Pipe) beginnt, wird angenommen, dass es sich um einen Befehl handelt, der ausgeführt wird, um das Fehlerprotokolls zu verarbeiten.

### Beispiel

```
ErrorLog "|/usr/local/bin/httpd_errors"
```

Die Verwendung von `syslog` anstelle eines Dateinamens aktiviert die Protokollierung mittels `syslogd(8)`, sofern das System es unterstützt. Als Voreinstellung wird der `syslog`-Typ (`syslog facility`) `local7` verwendet, Sie können dies jedoch auch überschreiben, indem Sie die Syntax `syslog:facility` verwenden, wobei

*facility* einer der Namen sein kann, die üblicherweise in syslog(1) dokumentiert sind.

### Beispiel

```
ErrorLog syslog:user
```

SICHERHEITSHINWEIS: Lesen Sie das Dokument [Sicherheitshinweise](#) zu Einzelheiten darüber, warum Ihre Sicherheit gefährdet sein kann, wenn das Verzeichnis, in dem die Log-Dateien gespeichert werden, für jemand anderen, als den Benutzer, der den Server gestartet hat, beschreibbar ist.

### Anmerkung

Bei der Eingabe eines Dateipfads auf nicht-Unix-Plattformen sollte darauf geachtet werden, nur (Vorwärts-)Schrägstriche zu verwenden, auch wenn die Plattform rückwärts gerichtete Schrägstriche (Backslashes) erlaubt. Im Allgemeinen ist es eine gute Idee, innerhalb der Konfigurationsdateien immer Vorwärts-Schrägstriche zu verwenden.

### Siehe auch

- [LogLevel](#)
- [Apache-Log-Dateien](#)



## FileETag Directive

<b>Beschreibung:</b>	Dateiattribute, die zur Erstellung des HTTP-Response-Headers ETag verwendet werden
<b>Syntax:</b>	FileETag <i>Komponente</i> ...
<b>Voreinstellung:</b>	FileETag INode MTime Size
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	FileInfo
<b>Status:</b>	Core
<b>Modul:</b>	core

Wenn dem Dokument eine Datei zugrundeliegt, bestimmt die Direktive **FileETag** die Dateiattribute, die zur Erstellung des HTTP-Response-Headers ETag (Entity-Tag) verwendet werden. (Der Wert von ETag wird bei der Cache-Verwaltung zur Einsparung von Netzwerk-Bandbreite benutzt.) Im Apache 1.3.22 und früher wurde der ETag-Wert *stets* aus der I-Node, der Größe und dem Datum der letzten Änderung (*mtime*) der Datei gebildet. Die Direktive **FileETag** erlaubt es Ihnen, zu bestimmen, welche dieser Eigenschaften -- falls überhaupt -- verwendet werden sollen. Die gültigen Schlüsselworte lauten:

### **INode**

Die I-Node-Nummer wird in die Berechnung mit einbezogen

### **MTime**

Datum und Uhrzeit der letzten Änderung werden mit einbezogen

### **Size**

Die Anzahl der Bytes in der Datei wird mit einbezogen

### **All**

Alle verfügbaren Angaben werden verwendet. Die ist gleichbedeutend mit:

FileETag INode MTime Size

## None

Es wird keine ETag-Angabe in die Antwort eingefügt, wenn dem Dokument eine Datei zugrundeliegt.

Den Schlüsselwörtern INode, MTime und Size kann entweder ein + oder ein - vorangestellt werden, was die Änderung einer Vorgabe erlaubt, die von einem größeren Umfeld geerbt wurde. Jedes Schlüsselwort ohne ein solches Präfix hebt die ererbte Einstellung sofort und vollständig auf.

Wenn die Konfiguration für ein Verzeichnis FileETag INode MTime Size enthält und die eines Unterverzeichnisses FileETag -INode, dann ist die Einstellung für das Unterverzeichnis (die an jedes Unter-Unterverzeichnis weitervererbt wird, welches dies nicht überschreibt) äquivalent mit FileETag MTime Size.



<b><u>Beschreibung:</u></b>	Enthält Direktiven, die sich nur auf passende Dateinamen beziehen
<b><u>Syntax:</u></b>	<code>&lt;Files <i>Dateiname</i>&gt; ... &lt;/Files&gt;</code>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	All
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive `<Files>` begrenzt die Reichweite der enthaltenen Anweisungen auf Dateinamen. Sie ist vergleichbar mit den Direktiven `<Directory>` und `<Location>`. Sie muss eine passende `</Files>`-Anweisung besitzen. Die innerhalb dieses Abschnittes angegebenen Direktiven werden auf jedes Objekt mit einem Basisnamen (letzte Komponente des Dateinamens) angewendet, der auf die angegebenen Dateinamen passt. `<Files>`-Container werden, nachdem die `<Directory>`-Container und `.htaccess`-Dateien gelesen sind, jedoch vor den `<Location>`-Containern, in der Reihenfolge ihres Auftretens ausgeführt. Beachten Sie, dass `<Files>`-Anweisungen innerhalb von `<Directory>`-Containern auftreten können, um den Teil des Dateisystems einzuschränken, den sie betreffen.

Das Argument *Dateiname* kann einen Dateinamen oder eine Zeichenkette mit Platzhaltern enthalten, wobei `?` auf ein einzelnes Zeichen passt und `*` auf eine beliebige Folge von Zeichen. Erweiterte reguläre Ausdrücke können ebenfalls verwendet werden, indem das Zeichen `~` hinzugefügt wird. Beispielsweise würde

```
<Files ~ "\.(gif|jpe?g|png)$">
```

auf die gebräuchlichsten Grafikformate im Internet passen.  
[<FilesMatch>](#) wird jedoch bevorzugt.

Beachten Sie bitte, dass die [<Files>](#)-Container anders als [<Directory>](#)- und [<Location>](#)-Container innerhalb von .htaccess-Dateien verwendet werden können. Dies erlaubt den Anwendern auf Dateiebene die Kontrolle über ihre eigenen Dateien.

## Siehe auch

- [Wie die Abschnitte <Directory>, <Location> und <Files> arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird



<b>Beschreibung:</b>	Enthält Direktiven, die für Dateinamen gelten, die auf einen regulären Ausdruck passen
<b>Syntax:</b>	<code>&lt;FilesMatch regex&gt; ... &lt;/FilesMatch&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	All
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive `<FilesMatch>` begrenzt wie die Direktive `<Files>` die enthaltenen Anweisungen auf Dateinamen. Sie akzeptiert jedoch reguläre Ausdrücke. Beispielsweise würde

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

auf die gebräuchlichsten Grafikformate im Internet passen.

## Siehe auch

- [Wie die Abschnitte <Directory>, <Location> und <Files> arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird



## ForceType-Direktive

<b>Beschreibung:</b>	Erzwingt die Auslieferung aller passenden Dateien mit dem angegebenen MIME-Content-Type
<b>Syntax:</b>	ForceType <i>MIME-Type</i>   None
<b>Kontext:</b>	Verzeichnis, .htaccess
<b>AllowOverride:</b>	FileInfo
<b>Status:</b>	Core
<b>Modul:</b>	core
<b>Kompatibilität:</b>	Wurde im Apache 2.0 in den Core verschoben

Wenn sie innerhalb einer `.htaccess`-Datei, eines `<Directory>`-, `<Location>`- `<Files>`-Containers angegeben wird, erzwingt die Direktive die Auslieferung aller entsprechenden Dateien mit dem Content-Type, der durch *MIME-Type* definiert wurde. Wenn Sie zum Beispiel ein Verzeichnis voller GIF-Dateien haben, die Sie nicht alle durch `.gif` kennzeichnen wollen, können Sie angeben:

```
ForceType image/gif
```

Beachten Sie bitte, dass die Direktive anders als `DefaultType` alle MIME-Type-Zuordnungen überschreibt, einschließlich Dateiendungen, die einen Medientyp bezeichnen könnten.

Sie können jede `ForceType`-Angabe durch die Verwendung des Wertes `None` überschreiben:

```
# erzwinge image/gif für alle Dateien:
<Location /images>
  ForceType image/gif
</Location>

# hier jedoch normale MIME-Type-Zuordnungen:
<Location /images/mixed>
```

```
ForceType None
</Location>
```



<b><u>Beschreibung:</u></b>	Aktiviert DNS-Lookups auf Client-IP-Adressen
<b><u>Syntax:</u></b>	HostnameLookups On Off Double
<b><u>Voreinstellung:</u></b>	HostnameLookups Off
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Diese Direktive aktiviert die DNS-Abfrage (*Anm.d.Ü.: ein sogenannter DNS-Lookup*), so dass Hostnamen protokolliert (und in REMOTE\_HOST an CGIs/SSIs übergeben) werden können. Der Wert Double bezieht sich auf ein Double-Reverse-DNS-Lookup. D.h. nachdem ein Reverse-Lookup durchgeführt wurde, wird dann auf dem Ergebnis ein Forward-Lookup ausgeführt. Wenigstens eine der IP-Adressen aus dem Forward-Lookup muss der Originaladresse entsprechen. (In der "tcpwrappers"-Terminologie wird dies PARANOID genannt.)

Unabhängig von der Einstellung wird ein Double-Reverse-Lookup durchgeführt, wenn `mod_access` zur Zugriffskontrolle per Hostnamen eingesetzt wird. Dies ist aus Sicherheitsgründen notwendig. Beachten Sie, dass das Ergebnis dieses Double-Reverse-Lookups nicht generell verfügbar ist, solange Sie nicht `HostnameLookups Double` setzen. Wenn beispielsweise nur `HostnameLookups On` angegeben ist und eine Anfrage für ein Objekt erfolgt, welches durch Hostnamen-Beschränkungen geschützt ist, dann wird CGIs nur das Ergebnis des Singel-Reverse-Lookups in REMOTE\_HOST übergeben, egal ob das Doble-Reverse-Lookup fehlschlug oder nicht.

Die Voreinstellung ist Off, um Netzwerktraffic bei den Angeboten einzusparen, die nicht tatsächlich Reverse-Lookups benötigen. Es ist auch für die Endanwender besser, da sie nicht die zusätzliche

Wartezeit ertragen müssen, die ein Lookup mit sich bringt. Hoch frequentierte Angebote sollten diese Direktive auf 0f fassen. Das Hilfsprogramm [logresolve](#), das standardmäßig in das Unterverzeichnis bin Ihres Installationsverzeichnis kompiliert wird, kann dazu verwendet werden, um offline Hostnamen von protokollierten IP-Adressen nachzuschlagen.



<b><u>Beschreibung:</u></b>	Ermöglicht die Protokollierung der Identität des entfernten Anwenders nach RFC1413
<b><u>Syntax:</u></b>	IdentityCheck On Off
<b><u>Voreinstellung:</u></b>	IdentityCheck Off
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive ermöglicht die RFC1413-konforme Protokollierung des entfernten Benutzernamens für jede Verbindung, bei der auf der Client-Maschine identd oder etwas ähnliches läuft. Die Information wird im Zugriffsprotokoll festgehalten.

Der Information sollte außer für eine rudimentäre Benutzerverfolgung in keinsten Weise vertraut werden.

Beachten Sie bitte, dass dies beträchtliche Zeitprobleme beim Zugriff auf Ihren Server verursachen kann, da für jede Anfrage eine solche Rückfrage durchgeführt werden muss. Wenn Firewalls beteiligt sind, kann unter Umständen jede Rückfrage fehlschlagen und weitere 30 Sekunden Wartezeit zu jedem Hit zufügen. Daher ist dies im Allgemeinen bei öffentlichen Servern, die im Internet erreichbar sind, nicht besonders sinnvoll.



<b>Beschreibung:</b>	Schließt Direktiven ein, die nur ausgeführt werden, wenn eine Testbedingung beim Start wahr ist
<b>Syntax:</b>	<code>&lt;IfDefine [!]Parametername&gt; ... &lt;/IfDefine&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	All
<b>Status:</b>	Core
<b>Modul:</b>	core

Der Container `<IfDefine Test>...</IfDefine>` wird dazu verwendet, Direktiven als bedingt zu kennzeichnen. Die Direktiven innerhalb eines `<IfDefine>`-Abschnittes werden nur ausgeführt, wenn *Test* wahr ist. Ist *Test* falsch, wird alles zwischen der Start- und Endemarkierung ignoriert.

In der `<IfDefine>`-Anweisung kann *Test* eine von zwei Formen annehmen:

- *Parametername*
- *!Parametername*

Im ersten Fall werden die Direktiven zwischen der Start- und Endemarkierung nur ausgeführt, wenn der Parameter namens *Parametername* definiert ist. Die zweite Form kehrt den Test um und führt die Direktiven nur dann aus, wenn *Parametername* **nicht** definiert ist.

Das Argument *Parametername* ist ein sogenanntes "Define", das beim Start des Servers in der [httpd](#)-Befehlszeile durch `-DParameter` angegeben wird.

`<IfDefine>`-Container können ineinander verschachtelt werden, um einfache Multi-Parameter-Tests zu implementieren. Beispiel:

```
httpd -DReverseProxy ...

# httpd.conf
<IfDefine ReverseProxy>
  LoadModule rewrite_module modules/mod_rewrite.so
  LoadModule proxy_module modules/libproxy.so
</IfDefine>
```



<b>Beschreibung:</b>	Schließt Direktiven ein, die abhängig vom Vorhandensein oder Fehlen eines speziellen Moduls ausgeführt werden
<b>Syntax:</b>	<code>&lt;IfModule [!]Modulname&gt; ... &lt;/IfModule&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	All
<b>Status:</b>	Core
<b>Modul:</b>	core

Der Container `<IfModule Test>...</IfModule>` wird dazu verwendet, Direktiven als abhängig von dem Vorhandensein eines speziellen Moduls zu kennzeichnen. Die Direktiven innerhalb eines `<IfModule>`-Abschnitts werden nur ausgeführt, wenn *Test* wahr ist. Ist *Test* falsch, wird alles zwischen der Start- und Endemarkierung ignoriert.

In der `<IfModule>`-Anweisung kann *Test* eine von zwei Formen annehmen:

- *Modulname*
- *!Modulname*

Im ersten Fall werden die Direktiven zwischen der Start- und Endemarkierung nur ausgeführt, das Modul namens *Modulname* im Apache enthalten ist -- entweder einkompiliert oder mittels `LoadModule` dynamisch geladen. Die zweite Form dreht den Test um und führt die Direktiven nur aus, wenn *Modulname* **nicht** enthalten ist.

Das Argument *Modulname* ist der Dateiname des Moduls zum Zeitpunkt seiner Kompilierung, z.B. `mod_rewrite.c`. Wenn ein

Modul aus mehreren Quelltext-Dateien besteht, verwenden Sie den Namen der Datei, welche die Zeichenfolge STANDARD20\_MODULE\_STUFF enthält.

`<IfModule>`-Container können ineinander verschachtelt werden, um einfache Multi-Modul-Tests durchzuführen.

Dieser Container sollte verwendet werden, wenn Sie eine Konfigurationsdatei benötigen, die unabhängig davon funktioniert, ob ein bestimmtes Modul verfügbar ist oder nicht. Normalerweise ist es nicht notwendig, Direktiven in `<IfModule>`-Containern unterzubringen.



<b><u>Beschreibung:</u></b>	Fügt andere Konfigurationsdateien innerhalb der Server-Konfigurationsdatei ein
<b><u>Syntax:</u></b>	Include <i>Dateiname</i>   <i>Verzeichnis</i>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Die Platzhalter-Suche ist verfügbar seit 2.0.41

Die Direktive erlaubt das Einfügen anderer Konfigurationsdateien in die Konfigurationsdatei des Servers.

Shell-typische (fnmatch( )) Platzhalterzeichen können dazu verwendet werden, mehrere Dateien auf einmal in alphabetischer Reihenfolge einzufügen. Wenn **Include** darüber hinaus auf ein Verzeichnis anstatt auf eine Datei zeigt, liest der Apache alle Dateien in diesem Verzeichnis und allen Unterverzeichnissen ein. Das Einfügen ganzer Verzeichnisse ist jedoch nicht empfehlenswert, da temporäre Dateien sehr leicht versehentlich in einem Verzeichnis zurückgelassen werden, was [httpd](#) scheitern lassen kann.

Der angegebene Dateiname kann ein absoluter Pfad sein oder relativ zum **ServerRoot**-Verzeichnis angegeben werden.

Beispiele:

```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

Oder Sie geben Pfade relativ zu Ihrem **ServerRoot**-Verzeichnis an:

```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

---

Der Aufruf von `apachectl configtest` liefert eine Liste der Dateien, die während des Konfigurations-Tests verarbeitet werden:

```
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
```

## Siehe auch

- [apachectl](#)



<b>Beschreibung:</b>	Aktiviert persistente HTTP-Verbindungen
<b>Syntax:</b>	KeepAlive On Off
<b>Voreinstellung:</b>	KeepAlive On
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Keep-Alive-Erweiterung von HTTP/1.0 und die HTTP/1.1-Funktionalität persistenter Verbindungen unterstützt langlebige HTTP-Sitzungen, die es erlauben, mehrere Anfragen über die gleich TCP-Verbindung zu senden. In einigen Fällen wurde eine Beschleunigung der Wartezeiten von beinahe 50% für HTML-Dokumente mit vielen Bildern festgestellt. Um Keep-Alive-Verbindungen zu aktivieren, setzen Sie `KeepAlive On`.

Bei HTTP/1.0-Clients werden Keep-Alive-Verbindungen nur dann verwendet, wenn sie vom Client eigens angefordert werden. Desweiteren können Keep-Alive-Verbindungen bei einem HTTP/1.0-Client nur dann verwendet werden, wenn die Länge des Inhalts im Voraus bekannt ist. Dies impliziert, dass dynamische Inhalte wie CGI-Ausgaben, SSI-Seiten und servergenerierte Verzeichnisaufstellungen im Allgemeinen keine Keep-Alive-Verbindungen mit HTTP/1.0-Clients verwenden. Bei HTTP/1.1-Clients sind Keep-Alive-Verbindungen Voreinstellung, solange nichts anderes angegeben ist. Wenn der Client es anfordert, wird Chunked-Encoding verwendet, um Inhalte mit unbekannter Länge über persistente Verbindungen zu senden.

## Siehe auch

- [MaxKeepAliveRequests](#)



<b><u>Beschreibung:</u></b>	Zeitspanne, die der Server während persistenter Verbindungen auf nachfolgende Anfragen wartet
<b><u>Syntax:</u></b>	KeepAliveTimeout <i>Sekunden</i>
<b><u>Voreinstellung:</u></b>	KeepAliveTimeout 15
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Dies legt die Anzahl der Sekunden fest, die der Apache auf weitere Anfragen wartet, bevor er die Verbindung schließt. Nachdem einmal eine Anfrage entgegen genommen wurde, wird die durch die Direktive [Timeout](#) festgelegte Auszeit angewendet.

Auf stark belasteten Servern kann ein hoher [KeepAliveTimeout](#)-Wert zu Durchsatzminderungen führen. Je höher die Auszeit angegeben ist, desto länger ist der Apache damit beschäftigt, auf untätige Clients zu warten.



<b>Beschreibung:</b>	Beschränkt die eingeschlossenen Zugriffskontrollen auf bestimmte HTTP-Methoden
<b>Syntax:</b>	<code>&lt;Limit Methode [Methode] ... &gt; ... &lt;/Limit&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	All
<b>Status:</b>	Core
<b>Modul:</b>	core

Zugriffskontrollen gelten normalerweise für **alle** Zugriffsmethoden, was normalerweise auch das gewünschte Verhalten ist. **Im Allgemeinen sollten Zugriffskontrollen nicht in einen `<Limit>`-Container gepackt werden.**

Der Sinn der Direktive `<Limit>` ist es, den Effekt der Zugriffskontrollen auf die angegebenen HTTP-Methoden zu beschränken. Bei allen anderen Methoden haben die in der `<Limit>`-Gruppe enthaltenen Zugriffsbeschränkungen **keine Wirkung**. Im folgenden Beispiel gilt die Zugriffskontrolle nur für die Methoden POST, PUT und DELETE. Alle anderen Methoden bleiben ungeschützt:

```
<Limit POST PUT DELETE>
  Require valid-user
</Limit>
```

Sie können eine oder mehrere der folgenden Methoden angeben: GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK und UNLOCK. **Die Methodennamen unterscheiden zwischen Groß- und Kleinschreibung.** Wenn GET verwendet wird, sind HEAD-Anfragen

ebenfalls eingeschränkt. Die TRACE-Methode kann nicht limitiert werden.

Bei Zugriffsbeschränkungen ist ein `<LimitExcept>`-Abschnitt stets einem `<Limit>`-Abschnitt vorzuziehen, da ein `<LimitExcept>`-Abschnitt vor allen möglichen Methoden schützt.



<b>Beschreibung:</b>	Beschränkt Zugriffskontrollen auf alle HTTP-Methoden außer den genannten
<b>Syntax:</b>	<code>&lt;LimitExcept Methode [Methode] ... &gt; ... &lt;/LimitExcept&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	All
<b>Status:</b>	Core
<b>Modul:</b>	core

`<LimitExcept>` und `</LimitExcept>` werden dazu verwendet, eine Gruppe von Anweisungen zur Zugriffskontrolle zusammenzufassen, die dann auf jede HTTP-Methode angewendet werden, die **nicht** als Argument angegeben ist. D.h. dies ist das Gegenteil des `<Limit>`-Containers und kann zur Steuerung von Standard- und nicht-Standard-/unbekannten Methoden verwendet werden. Für weitere Einzelheiten lesen Sie bitte die Beschreibung zu `<Limit>`.

Beispiel:

```
<LimitExcept POST GET>
  Require valid-user
</LimitExcept>
```



<b><u>Beschreibung:</u></b>	Bestimmt die maximale Anzahl interner Umleitungen und verschachtelter Unteranfragen
<b><u>Syntax:</u></b>	<code>LimitInternalRecursion Zahl [Zahl]</code>
<b><u>Voreinstellung:</u></b>	<code>LimitInternalRecursion 10</code>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Verfügbar ab Apache 2.0.47

Eine interne Umleitung erfolgt beispielsweise, wenn die Direktive **Action** verwendet wird, welche die Originalanfrage intern zu einem CGI-Skript weiterleitet. Eine Unteranfrage (*Anm.d.Ü.:* engl. Subrequest) ist ein Mechanismus des Apache, um herauszufinden, was bei einer URI geschehen würde, wäre sie angefordert worden. `mod_dir` z.B. verwendet Unteranfragen, um nach den Dateien zu suchen, die in der **DirectoryIndex**-Anweisung aufgeführt sind.

**LimitInternalRecursion** bewahrt den Server vor einem Absturz, wenn er in eine Endlosschleife aus internen Umleitungen oder Unteranfragen hineinläuft. Derartige Schleifen werden gewöhnlich durch Fehlkonfiguration verursacht.

Die Direktive setzt zwei verschiedene Begrenzungen, welche je Anfrage ausgewertet werden. Die erste *Zahl* bestimmt die maximale Anzahl der Umleitungen, die aufeinander folgen dürfen. Die zweite *Zahl* legt fest, wie tief Unteranfragen ineinander verschachtelt werden dürfen. Wenn Sie lediglich eine *Zahl* angeben, wird sie beiden Begrenzungen zugewiesen.

### Beispiel

```
LimitInternalRecursion 5
```



<b><u>Beschreibung:</u></b>	Begrenzt die Gesamtgröße des vom Client gesendeten HTTP-Request-Body
<b><u>Syntax:</u></b>	LimitRequestBody <i>Bytes</i>
<b><u>Voreinstellung:</u></b>	LimitRequestBody 0
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	All
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive gibt die Anzahl der *Bytes* zwischen 0 (unbegrenzt) und 2147483647 (2GB) an, die im Request-Body (Datenteil der Anfrage) erlaubt sind.

Die Direktive **LimitRequestBody** erlaubt es dem Benutzer, die Größe des HTTP-Request-Bodys in dem Kontext zu begrenzen, in dem die Anweisung angegeben ist (Server, pro Verzeichnis, pro Datei oder pro Adresse). Wenn die Anfrage des Clients dieses Limit überschreitet, gibt der Server einen Fehler zurück anstatt die Anfrage zu bearbeiten. Die Größe des Datenteils einer Anfrage kann sehr stark variieren, abhängig von der Art der Ressource und den für diese Ressource erlaubten Methoden. CGI-Skripte verwenden den Datenteil üblicherweise zum Empfang von Formulardaten. Wird die PUT-Methode angewendet, dann muss der Wert mindestens so groß sein wie irgendeine Darstellungsform, die der Server für diese Ressource akzeptieren soll.

Die Direktive gibt dem Serveradministrator eine größere Kontrolle gegenüber abnormalem Verhalten von Clients, was bei der Vermeidung einiger Formen von Denial-of-Service-Attacks hilfreich sein kann.

Wenn Sie beispielsweise das Hochladen von Dateien zu einer bestimmten Adresse erlauben, aber die Größe der hochgeladenen Dateien auf 100K beschränken wollen, können Sie die folgende Anweisung verwenden:

```
LimitRequestBody 102400
```



**Beschreibung:** Begrenzt die Anzahl der HTTP-Request-Header, die vom Client entgegengenommen werden

**Syntax:** `LimitRequestFields Anzahl`

**Voreinstellung:** `LimitRequestFields 100`

**Kontext:** Serverkonfiguration

**Status:** Core

**Modul:** core

*Anzahl* ist ein Integer-Wert (eine positive Ganzzahl) zwischen 0 (unbegrenzt) und 32767. Die Voreinstellung wird durch die Konstante `DEFAULT_LIMIT_REQUEST_FIELDS` (100 bei der Auslieferung) zur Kompilierungszeit gesetzt.

Die Direktive `LimitRequestFields` erlaubt es dem Serveradministrator, die maximale Anzahl der in einem HTTP-Request erlaubten HTTP-Request-Header zu verändern. Für den Server muss dieser Wert größer sein als die Anzahl der Headerzeilen, die ein normaler Client senden könnte. Die Anzahl der Request-Header, die ein gewöhnlicher Client verwendet, überschreitet selten 20 Zeilen. Allerdings kann dies zwischen den verschiedenen Client-Ausführungen variieren, oft abhängig vom Ausmaß, mit dem der Anwender die genaue Content-Negotiation-Unterstützung seines Browsers konfiguriert hat. Optionale HTTP-Erweiterungen äußern sich oft in Form von HTTP-Headern.

Die Direktive gibt dem Serveradministrator eine größere Kontrolle gegenüber abnormalem Verhalten von Clients, was bei der Vermeidung einiger Formen von Denial-of-Service-Attacks hilfreich sein kann. Der Wert sollte erhöht werden, wenn normale Clients eine Fehlermeldung vom Server erhalten, die besagt, dass mit der Anfrage zu viele Headerzeilen gesendet wurden.

Beispiel:

```
LimitRequestFields 50
```



<b><u>Beschreibung:</u></b>	Begrenzt die Länge des vom Client gesendeten HTTP-Request-Headers
<b><u>Syntax:</u></b>	<code>LimitRequestFieldSize Bytes</code>
<b><u>Voreinstellung:</u></b>	<code>LimitRequestFieldSize 8190</code>
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive gibt die Anzahl der *Bytes* an, die in einem HTTP-Header erlaubt sind.

Die Direktive `LimitRequestFieldSize` erlaubt es dem Serveradministrator, die maximale Größe eines HTTP-Request-Headers zu verringern oder erhöhen. Für den Server muss der Wert groß genug sein, um eine beliebige Headerzeile einer normalen Client-Anfrage vorzuhalten. Die Größe variiert stark zwischen den verschiedenen Client-Ausführungen, oft abhängig vom Ausmaß, mit dem der Anwender die genaue Content-Negotiation-Unterstützung seines Browsers konfiguriert hat. SPNEGO-Authentisierungs-Header können bis zu 12392 Bytes lang sein.

Die Direktive gibt dem Serveradministrator eine größere Kontrolle gegenüber abnormalem Verhalten von Clients, was bei der Vermeidung einiger Formen von Denial-of-Service-Attacken hilfreich sein kann.

Beispiel:

```
LimitRequestFieldSize 4094
```

Unter normalen Umständen sollte die Voreinstellung nicht verändert werden.

---

Sie benötigen mindestens Apache 2.0.53, um das Limit über den einkompilierten Wert von `DEFAULT_LIMIT_REQUEST_FIELDSIZE` hinaus zu setzen (8190 bei der Auslieferung).



<b><u>Beschreibung:</u></b>	Begrenzt die Länge der vom Client entgegengenommenen HTTP-Anfragezeile
<b><u>Syntax:</u></b>	LimitRequestLine <i>Bytes</i>
<b><u>Voreinstellung:</u></b>	LimitRequestLine 8190
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive legt die Anzahl der *Bytes* zwischen 0 und dem Wert der zur Kompilierungszeit definierten Konstante `DEFAULT_LIMIT_REQUEST_LINE` (8190 bei der Auslieferung) fest, die in der HTTP-Anfragezeile erlaubt sind.

Die Direktive `LimitRequestLine` erlaubt es dem Serveradministrator, die maximale Größe der HTTP-Anfragezeile auf einen Wert unterhalb der normalen, im Server einkompilierten Größe des Eingabepuffers zu verringern. Da die Anfragezeile aus der HTTP-Methode, der URI und der Protokollversion besteht, bedeutet die `LimitRequestLine`-Direktive eine Beschränkung der Länge der für eine Anfrage an den Server erlaubten Anfrage-URI. Für den Server muss der Wert groß genug sein, um jeden seiner Ressourcennamen vorzuhalten, einschließlich aller Informationen, die im Query-String einer GET-Anfrage übergeben werden können.

Die Direktive gibt dem Serveradministrator eine größere Kontrolle gegenüber abnormalem Verhalten von Clients, was bei der Vermeidung einiger Formen von Denial-of-Service-Attacken hilfreich sein kann.

Beispiel:

```
LimitRequestLine 4094
```

Unter normalen Umständen sollte die Voreinstellung nicht verändert werden.



<b><u>Beschreibung:</u></b>	Begrenzt die Größe eines XML-basierten Request-Bodys
<b><u>Syntax:</u></b>	LimitXMLRequestBody <i>Bytes</i>
<b><u>Voreinstellung:</u></b>	LimitXMLRequestBody 1000000
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	All
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Dies gibt die Grenze für die maximale Größe (in Bytes) des XML-basierten Request-Bodys an. Der Wert 0 deaktiviert diese Prüfung.

Beispiel:

```
LimitXMLRequestBody 0
```



<b>Beschreibung:</b>	Wendet die enthaltenen Direktiven nur auf die entsprechenden URLs an
<b>Syntax:</b>	<code>&lt;Location URL-Pfad URL&gt; ... &lt;/Location&gt;</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive `<Location>` begrenzt die Reichweite der enthaltenen Anweisungen auf URLs. Sie ist der Direktive `<Directory>` ähnlich und startet einen Abschnitt, der mit der Anweisung `</Location>` abgeschlossen wird. `<Location>`-Container werden, nachdem die `<Directory>`-Container und `.htaccess`-Dateien gelesen wurden, und nach den `<Files>`-Containern, in der Reihenfolge ausgeführt, in der sie in der Konfigurationsdatei erscheinen.

`<Location>`-Abschnitte operieren vollständig außerhalb des Dateisystems. Dies hat mehrere Konsequenzen. Am Wichtigsten, `<Location>`-Anweisungen sollten nicht dafür verwendet werden, den Zugriff zu Teilen des Dateisystems zu steuern. Da mehrere unterschiedliche URLs auf die gleiche Stelle des Dateisystems zeigen können, könnte eine solche Zugriffskontrolle u.U. umgangen werden.

### Wann sollte `<Location>` verwendet werden

Verwenden Sie `<Location>`, um Anweisungen auf Inhalte anzuwenden, die außerhalb des Dateisystems abgelegt sind. Benutzen Sie `<Directory>` und `<Files>` für Inhalte, die innerhalb des Dateisystems abgelegt sind. Eine Ausnahme bildet `<Location />`, welches ein einfacher Weg ist, um eine Konfiguration auf den gesamten Server anzuwenden.

Für alle nicht-Proxy-Anfragen ist die entsprechende URL ein URL-Pfad in der Form `/path/`. Es dürfen weder ein Schema, noch ein Hostname, noch ein Port, noch ein Query-String einbezogen werden. Für Proxy-Anfragen hat die Vergleichs-URL die Form `schema://servername/path`. Das Präfix muss angegeben werden.

Die URL kann Platzhalter verwenden. In einer Zeichenfolge mit Platzhaltern entspricht `?` einem einzelnen Zeichen und `*` einer beliebigen Zeichenfolge.

Erweiterte reguläre Ausdrücke können ebenfalls verwendet werden, indem das Zeichen `~` hinzugefügt wird. Beispielsweise würde

```
<Location ~ "/(extra|special)/data">
```

auf URLs passen, welche die Zeichenfolge `/extra/data` oder `/special/data` enthalten. Die Direktive `<LocationMatch>` verhält sich genauso wie `<Location>` mit regulären Ausdrücken.

Die Funktionalität von `<Location>` ist insbesondere dann nützlich, wenn sie mit der `SetHandler`-Direktive kombiniert wird. Um zum Beispiel Statusabfragen zu aktivieren, sie aber nur von Browsern aus `foo.com` zuzulassen, könnten Sie schreiben:

```
<Location /status>
  SetHandler server-status
  Order Deny,Allow
  Deny from all
  Allow from .foo.com
</Location>
```

**Anmerkung zu / (Schrägstrich, Slash)**

Das Slash-Zeichen hat eine besondere Bedeutung, je nachdem, wo es in der URL erscheint. Manche werden sein Verhalten vom Dateisystem gewohnt sein, wo mehrere aufeinanderfolgende Schrägstriche häufig zu einem Schrägstrich zusammengefaßt werden (*d.h.* /home///foo ist das gleiche wie /home/foo). Im URL-Raum ist dies nicht notwendigerweise genauso. Bei der Direktive `<LocationMatch>` und der `<Location>`-Version mit regulären Ausdrücken müssen Sie explizit mehrere Schrägstriche angeben, wenn Sie genau dies beabsichtigen.

Beispielsweise würde `<LocationMatch ^/abc>` auf die angeforderte URL /abc passen, nicht aber auf //abc. Die Direktive `<Location>` (ohne reguläre Ausdrücke) verhält sich ähnlich, wenn sie für Proxy-Anfragen verwendet wird. Wenn `<Location>` (ohne reguläre Ausdrücke) jedoch für nicht-Proxy-Anfragen verwendet wird, werden stillschiegend mehrere Schrägstriche mit einem einzigen Schrägstrich gleichgesetzt. Geben Sie beispielsweise `<Location /abc/def>` an und die Anfrage lautet auf /abc//def, dann greift die Anweisung.

## Siehe auch

- [Wie die Abschnitte <Directory>, <Location> und <Files> arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird



<b><u>Beschreibung:</u></b>	Wendet die enthaltenen Direktiven nur auf URLs an, die auf reguläre Ausdrücke passen
<b><u>Syntax:</u></b>	<code>&lt;LocationMatch regex&gt; ... &lt;/LocationMatch&gt;</code>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive `<LocationMatch>` begrenzt die Reichweite der enthaltenen Anweisungen in der gleichen Weise wie `<Location>` auf URLs. Sie verwendet jedoch reguläre Ausdrücke als Argument anstelle einer einfachen Zeichenkette. Beispielsweise würde

```
<LocationMatch "/(extra|special)/data">
```

auf URLs passen, welche die Zeichenfolge `/extra/data` oder `/special/data` enthalten.

## Siehe auch

- [Wie die Abschnitte `<Directory>`, `<Location>` und `<Files>` arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird



<b>Beschreibung:</b>	Steuert die Ausführlichkeit des Fehlerprotokolls
<b>Syntax:</b>	LogLevel <i>Level</i>
<b>Voreinstellung:</b>	LogLevel warn
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

**LogLevel** stellt die Ausführlichkeit der Nachrichten ein, die im Fehlerprotokoll aufgezeichnet werden (siehe Direktive **ErrorLog**). Die folgenden, nach absteigender Aussagekraft sortierten *Level* sind verfügbar:

Level	Beschreibung	Beispiel
emerg	Notfall - das System ist unbenutzbar.	"Child cannot open lock file. Exiting" ( <i>Anm.d.Ü.:</i> "Kindprozess kann die Lock-Datei nicht öffnen. Beende Programm")
alert	Maßnahmen müssen unverzüglich ergriffen werden.	"getpwuid: couldn't determine user name from uid" ( <i>Anm.d.Ü.:</i> "getpwuid: kann keinen Benutzernamen aus der UID ermitteln")
crit	Kritischer Zustand.	"socket: Failed to get a socket, exiting child" ( <i>Anm.d.Ü.:</i> "socket: Socket-Zuweisung fehlgeschlagen, beende Kindprozess")
error	Fehlerbedingung.	"Premature end of script headers" ( <i>Anm.d.Ü.:</i> "Vorzeitiges Ende der Skript-Header")
warn	Warnung.	"child process 1234 did not exit, sending another SIGHUP" ( <i>Anm.d.Ü.:</i> "Kindprozess 1234 nicht beendet,

		sende ein weiteres SIGHUP")
notice	Normaler, aber signifikanter Zustand.	"httpd: caught SIGBUS, attempting to dump core in ..." ( <i>Anm.d.Ü.:</i> "httpd: SIGBUS empfangen, versuche Speicherabbild nach ... zu schreiben")
info	Information.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..." ( <i>Anm.d.Ü.:</i> "Server scheint beschäftigt zu sein, (möglicherweise müssen Sie StartServers oder Min/MaxSpareServers erhöhen)")
debug	Debug-Level-Nachrichten	"Opening config file ..." ( <i>Anm.d.Ü.:</i> "Öffne Konfigurationsdatei ...")

Geben Sie einen bestimmten Level an, denn werden Nachrichten von allen höheren Leveln ebenso angezeigt. *Z.B.:* Wenn `LogLevel info` eingestellt ist, dann werden Nachrichten der Log-Level `notice` und `warn` ebenso eingetragen.

Es wird empfohlen, mindestens den Level `crit` zu verwenden.

Beispiel:

```
LogLevel notice
```

### Hinweis

Beim Protokollieren in eine reguläre Datei können Nachrichten des Levels `notice` nicht unterdrückt werden und werden daher immer protokolliert. Dies trifft allerdings nicht zu wenn mittels `syslog` protokolliert wird.



## MaxKeepAliveRequests Direktive

<b>Beschreibung:</b>	Anzahl der Anfragen, die bei einer persistenten Verbindung zulässig sind
<b>Syntax:</b>	MaxKeepAliveRequests <i>Anzahl</i>
<b>Voreinstellung:</b>	MaxKeepAliveRequests 100
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive `MaxKeepAliveRequests` begrenzt die Anzahl der Anfragen, die pro Verbindung zulässig sind, wenn `KeepAlive` eingeschaltet ist. Bei der Einstellung 0 sind unbegrenzt viele Anfragen erlaubt. Wir empfehlen für diese Einstellung einen hohen Wert für eine maximale Serverleistung.

Beispiel:

```
MaxKeepAliveRequests 500
```



<b><u>Beschreibung:</u></b>	Number of ranges allowed before returning the complete resource
<b><u>Syntax:</u></b>	MaxRanges default   unlimited   none   <i>number-of-ranges</i>
<b><u>Voreinstellung:</u></b>	MaxRanges 200
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Available in Apache HTTP Server 2.0.65 and later

Die Dokumentation zu dieser Direktive wurde noch nicht übersetzt. Bitte schauen Sie in die englische Version.



<b>Beschreibung:</b>	Bestimmt eine IP-Adresse für den Betrieb namensbasierter virtueller Hosts
<b>Syntax:</b>	NameVirtualHost <i>Adresse[:Port]</i>
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive `NameVirtualHost` ist erforderlich, wenn Sie [namensbasierte virtuelle Hosts](#) konfigurieren möchten.

Obwohl *Adresse* eine Hostname sein kann, wird empfohlen, dass Sie stets eine IP-Adresse verwenden, z.B.:

```
NameVirtualHost 111.22.33.44
```

Mit der `NameVirtualHost`-Anweisung geben Sie die IP-Adresse an, unter der der Server Anfragen für namensbasierte virtuelle Hosts entgegennimmt. Das ist üblicherweise die Adresse, zu der die Namen Ihrer namensbasierten virtuellen Hosts aufgelöst werden. Falls eine Firewall oder ein anderer Proxy die Anfrage in Empfang nimmt und Sie zu einer weiteren IP-Adresse des Servers weiterleitet, müssen Sie die IP-Adresse der physikalischen Schnittstelle der Maschine angeben, welche die Anfragen bedient. Wenn Sie mehrere namensbasierte Hosts an verschiedenen Adressen betreiben, wiederholen Sie einfach die Anweisung für jede Adresse.

### Anmerkung

Beachten Sie, dass der "Hauptserver" und jeder `_default_`-Server **niemals** bei einer Anfrage an einer `NameVirtualHost`-IP-Adresse bedient wird (es sei denn, Sie geben aus irgendwelchen Gründen `NameVirtualHost` an, definieren

dann aber keine `VirtualHosts` für diese Adresse).

Optional können Sie die Nummer eines Ports angeben, an dem namensbasierte virtuelle Hosts verwendet werden sollen. Beispiel:

```
NameVirtualHost 111.22.33.44:8080
```

IPv6-Adressen müssen, wie im folgenden Beispiel angegeben, in eckige Klammern eingeschlossen werden:

```
NameVirtualHost [2001:db8::a00:20ff:fea7:ccea]:8080
```

Um an allen Schnittstellen Anfragen zu empfangen, können Sie `*` als Argument verwenden.

```
NameVirtualHost *
```

### Argument der Direktive `<VirtualHost>`

Beachten Sie, dass das Argument der `<VirtualHost>`-Anweisung exakt auf das Argument der `NameVirtualHost`-Anweisung passen muss.

```
NameVirtualHost 1.2.3.4
<VirtualHost 1.2.3.4>
# ...
</VirtualHost>
```

## Siehe auch

- [Dokumentation zu virtuellen Hosts](#)



<b>Beschreibung:</b>	Definiert, welche Eigenschaften oder Funktionen in einem bestimmten Verzeichnis verfügbar sind
<b>Syntax:</b>	Options [+ -] <i>Option</i> [[+ -] <i>Option</i> ] ...
<b>Voreinstellung:</b>	Options All
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	Options
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive `Options` steuert, welche Eigenschaften bzw. Funktionen in einem bestimmten Verzeichnis verfügbar sind.

*Option* kann auf None gesetzt werden, wobei keine der besonderen Eigenschaften verfügbar sind, oder auf eines oder mehrere der folgenden:

### All

Alle Optionen außer MultiViews. Dies ist die Voreinstellung.

### ExecCGI

Die Ausführung von CGI-Skripten, welche `mod_cgi` verwenden, ist erlaubt.

### FollowSymLinks

Der Server folgt symbolischen Links in diesem Verzeichnis.

Auch wenn der Server symbolischen Links folgt, bedeutet dies *nicht*, dass der zum Abgleich gegen `<Directory>`-Abschnitte verwendete Pfadname wechselt.

Beachten Sie auch, dass diese Option innerhalb eines

<Location>-Abschnitts ignoriert wird.

## **Includes**

Server Side Includes, die von [mod\\_include](#) bereitgestellt werden, sind erlaubt.

## **IncludesNOEXEC**

Server Side Includes sind erlaubt, `#exec cmd` und `#exec cgi` sind jedoch deaktiviert. Es ist aber noch möglich, CGI-Skripte aus [ScriptAlias](#)-Verzeichnissen mittels `#include virtual` einzubinden.

## **Indexes**

Wenn eine URL, die auf ein Verzeichnis zeigt, in dem sich keine durch [DirectoryIndex](#) definierte Indexdatei (z.B. `index.html`) befindet, dann liefert [mod\\_autoindex](#) eine formatierte Auflistung des Verzeichnisses zurück.

## **MultiViews**

"MultiViews" sind bei der Verwendung von [mod\\_negotiation](#) erlaubt (siehe [Content-Negotiation](#)).

## **SymLinksIfOwnerMatch**

Der Server folgt nur symbolischen Links, bei denen die Zieldatei bzw. das Zielverzeichnis der gleichen Benutzerkennung gehört, wie der Link.

### **Anmerkung**

Diese Option wird innerhalb eines <Location>-Abschnitts ignoriert.

Wenn mehrere [Options](#) auf ein Verzeichnis angewandt werden können, dann wird normalerweise die spezifischste (*Anm.d.Ü.: Gemeint ist die zuletzt ausgeführte Option.*) verwendet und alle anderen werden ignoriert; die Optionen werden nicht vermischt.

(Siehe auch [Wie Abschnitte zusammengeführt werden.](#)) Wenn jedoch *allen* Optionen der **Options**-Anweisung eines der Zeichen + oder - vorangestellt wird, werden die Optionen zusammengemischt. Jede Option mit vorangestelltem + wird zu den momentan gültigen Optionen hinzugefügt und jede Option mit vorangestelltem - wird aus den derzeit gültigen Optionen entfernt.

### Warnung

Die Verwendung von **Options**, bei der einzelne Optionen mit + oder - und Optionen ohne diese Präfixe vermischt werden, ist ungültig und führt häufig zu unerwarteten Ergebnissen.

So wird zum Beispiel ohne die Zeichen + und -

```
<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
  Options Includes
</Directory>
```

für das Verzeichnis /web/docs/spec wird jetzt lediglich Includes gesetzt. Wenn die zweite **Options**-Anweisung jedoch +- und --Zeichen verwenden würde,

```
<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>

<Directory /web/docs/spec>
  Options +Includes -Indexes
</Directory>
```

dann würden die Optionen FollowSymLinks und Includes für das Verzeichnis /web/docs/spec gesetzt.

### **Anmerkung**

Die Verwendung von `-IncludesNOEXEC` oder `-Includes` deaktiviert Server Side Includes unabhängig von der vorigen Einstellung vollständig.

Die Voreinstellung ist `All`, sofern keine anderen Angaben gemacht wurden.



<b>Beschreibung:</b>	Wählt die authentisierten Benutzer aus, die auf eine Ressource zugreifen können
<b>Syntax:</b>	Require <i>Name</i> [ <i>Name</i> ] ...
<b>Kontext:</b>	Verzeichnis, .htaccess
<b>AllowOverride:</b>	AuthConfig
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive wählt aus, welche authentisierten Benutzer auf eine Ressource zugreifen dürfen. Folgende Syntax ist erlaubt:

**Require user *User-ID* [*User-ID*] ...**

Nur die genannten Benutzer dürfen auf die Ressource zugreifen.

**Require group *Gruppenname* [*Gruppenname*] ...**

Nur Benutzer der genannten Gruppen dürfen auf die Ressource zugreifen.

**Require valid-user**

Alle gültigen Benutzer dürfen auf die Ressource zugreifen.

**Require** muss von den Direktiven [AuthName](#) und [AuthType](#) sowie Direktiven wie [AuthUserFile](#) und [AuthGroupFile](#) (zur Definition von Benutzern und Gruppen) begleitet werden, um korrekt zu funktionieren. Beispiel:

```
AuthType Basic
AuthName "Geschützte Ressource"
AuthUserFile /web/users
AuthGroupFile /web/groups
Require group admin
```

Zugriffskontrollen, die in dieser Form angewandt werden, gelten für **alle** Methoden. **Dies ist normalerweise gewünscht.** Wenn

Sie Zugriffskontrollen nur auf bestimmte Methoden anwenden möchten, während andere Methoden ungeschützt bleiben, dann müssen Sie die **Require**-Anweisung innerhalb eines <Limit>-Abschnitts platzieren.

## Siehe auch

- [Satisfy](#)
- [mod\\_access](#)



<b><u>Beschreibung:</u></b>	Begrenzt den CPU-Verbrauch von Prozessen, die von Apache-Kindprozessen gestartet wurden
<b><u>Syntax:</u></b>	<code>RLimitCPU <i>Sekunden</i>   max [<i>Sekunden</i>   max]</code>
<b><u>Voreinstellung:</u></b>	unbestimmt; verwendet die Voreinstellung des Systems
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	All
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Akzeptiert einen oder zwei Parameter. Der erste Parameter setzt eine weiche Ressourcenbegrenzung für alle Prozesse, der zweite Parameter setzt die Maximalgrenze für die Ressourcennutzung. Jeder der Parameter kann eine Zahl oder max sein. max zeigt dem Server an, dass das vom Betriebssystem erlaubte Maximum verwendet werden soll. Das Anheben der maximal erlaubten Ressourcennutzung erfordert, dass der Server als root läuft, zumindest in der anfänglichen Startphase.

Dies wird auf Prozesse angewendet, die von Anfragen bearbeitenden Apache-Kindprozessen abgespalten werden, nicht auf die Apache-Kindprozesse selbst. Das beinhaltet CGI-Skripte und SSI-exec-Befehle, nicht jedoch Prozesse, die vom Apache-Elternprozess abgespalten werden, wie z.B. Protokollierung.

CPU-Ressourcenbegrenzung wird in Sekunden pro Prozess ausgedrückt.

**Siehe auch**

- RLimitMEM
- RLimitNPROC



<b><u>Beschreibung:</u></b>	Begrenzt den Speicherverbrauch von Prozessen, die von Apache-Kindprozessen gestartet wurden
<b><u>Syntax:</u></b>	<code>RLimitMEM Bytes max [Bytes max]</code>
<b><u>Voreinstellung:</u></b>	unbestimmt; verwendet die Voreinstellung des Systems
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	All
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Akzeptiert einen oder zwei Parameter. Der erste Parameter setzt eine weiche Ressourcenbegrenzung für alle Prozesse, der zweite Parameter setzt die Maximalgrenze für die Ressourcennutzung. Jeder der Parameter kann eine Zahl oder max sein. max zeigt dem Server an, dass das vom Betriebssystem erlaubte Maximum verwendet werden soll. Das Anheben der maximal erlaubten Ressourcennutzung erfordert, dass der Server als root läuft, zumindest in der anfänglichen Startphase.

Dies wird auf Prozesse angewendet, die von Anfragen bearbeitenden Apache-Kindprozessen abgespalten werden, nicht auf die Apache-Kindprozesse selbst. Das beinhaltet CGI-Skripte und SSI-exec-Befehle, nicht jedoch Prozesse, die vom Apache-Elternprozess abgespalten werden, wie z.B. Protokollierung.

Die Begrenzung des Speicherverbrauchs wird in Bytes pro Prozess ausgedrückt.

## Siehe auch

- [RLimitCPU](#)

- RLimitNPROC



<b><u>Beschreibung:</u></b>	Begrenzt die Anzahl der Prozesse, die von Prozessen gestartet werden können, der ihrerseits von Apache-Kinprozessen gestartet wurden
<b><u>Syntax:</u></b>	<code>RLimitNPROC Zahl max [Zahl max]</code>
<b><u>Voreinstellung:</u></b>	unbestimmt; verwendet die Voreinstellung des Systems
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	All
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Akzeptiert einen oder zwei Parameter. Der erste Parameter setzt eine weiche Ressourcenbegrenzung für alle Prozesse, der zweite Parameter setzt die Maximalgrenze für die Ressourcennutzung. Jeder der Parameter kann eine Zahl oder max sein. max zeigt dem Server an, dass das vom Betriebssystem erlaubte Maximum verwendet werden soll. Das Anheben der maximal erlaubten Ressourcennutzung erfordert, dass der Server als root läuft, zumindest in der anfänglichen Startphase.

Dies wird auf Prozesse angewendet, die von Anfragen bearbeitenden Apache-Kindprozessen abgespalten werden, nicht auf die Apache-Kindprozesse selbst. Dies beinhaltet CGI-Skripte und SSI-exec-Befehle, nicht jedoch Prozesse, die vom Apache-Elternprozess abgespalten werden, wie z.B. Protokollierung.

Prozessbegrenzungen steuern die Anzahl der Prozesse pro Benutzer.

### Anmerkung

Wenn CGI-Prozesse nicht unter anderen Benutzerkennungen als der User-ID des Webservers laufen, dann beschränkt diese Direktive die Anzahl der Prozesse, die der Server selbst erstellen kann. Kennzeichen einer solchen Situation sind **cannot fork**-Meldungen (*Anm.d.Ü.:* kann nicht abspalten) in der Datei `error_log`.

## Siehe auch

- [RLimitMEM](#)
- [RLimitCPU](#)



<b>Beschreibung:</b>	Zusammenspiel von rechnerbasierter Zugriffskontrolle und Benutzerauthentisierung
<b>Syntax:</b>	Satisfy Any All
<b>Voreinstellung:</b>	Satisfy All
<b>Kontext:</b>	Verzeichnis, .htaccess
<b>AllowOverride:</b>	AuthConfig
<b>Status:</b>	Core
<b>Modul:</b>	core
<b>Kompatibilität:</b>	Wird seit Version 2.0.51 von <code>&lt;Limit&gt;</code> und <code>&lt;LimitExcept&gt;</code> beeinflusst

Verfahrensweise für den Zugriff, falls sowohl `Allow` als auch `Require` verwendet werden. Der Parameter kann entweder `All` oder `Any` sein. Die Direktive ist nur dann nützlich, wenn der Zugriff zu einem bestimmten Bereich durch Benutzername/Passwort *und* Clientrechner-Adressen eingeschränkt ist. In diesem Fall verlangt die Voreinstellung (`All`), dass der Client die Adressbeschränkung passiert *und* eine gültige Benutzerkennung und ein gültiges Passwort übermittelt. Mit der Auswahl `Any` wird dem Client der Zugriff erlaubt, wenn er entweder die Rechner-Beschränkung passiert oder einen gültigen Benutzernamen und ein gültiges Passwort übermittelt. Dies kann verwendet werden, um einen Bereich mit einem Passwort zu schützen, jedoch Clients von bestimmten Adressen ohne Abfrage des Passwortes zuzulassen.

Wenn Sie beispielsweise möchten, dass Personen aus Ihrem privaten Netzwerk unbeschränkten Zugriff zu Teilen Ihres Webangebots haben, jedoch verlangen, dass Personen außerhalb Ihres privaten Netzwerks ein Passwort übergeben müssen, können Sie eine Konfiguration ähnlich der folgenden verwenden:

```
Require valid-user
```

```
Allow from 192.168.1
Satisfy Any
```

Seit Version 2.0.51 können **Satisfy**-Anweisungen durch [<Limit>](#)- und [<LimitExcept>](#)-Abschnitte auf bestimmte Methoden beschränkt werden.

## Siehe auch

- [Allow](#)
- [Require](#)



<b><u>Beschreibung:</u></b>	Methode zur Ermittlung des Interpreters von CGI-Skripten
<b><u>Syntax:</u></b>	ScriptInterpreterSource Registry Registry-Strict Script
<b><u>Voreinstellung:</u></b>	ScriptInterpreterSource Script
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	ausschließlich Win32; Die Option Registry-Strict ist verfügbar seit Apache 2.0.

Die Direktive steuert, wie der Apache den Interpreter zur Ausführung von CGI-Skripten bestimmt. Die Voreinstellung ist Script. Dies veranlaßt den Apache, den Interpreter zu verwenden, auf den die Shebang-Zeile (erste Zeile, beginnt mit #!) im Skript zeigt. Auf Win32-Systemen sieht diese Zeile üblicherweise so aus:

```
#!C:/Perl/bin/perl.exe
```

oder, wenn perl im Pfad (Umgebungsvariable PATH) liegt, einfach:

```
#!perl
```

Die Einstellung ScriptInterpreterSource Registry veranlaßt eine Suche in HKEY\_CLASSES\_ROOT der Windows-Registrierungsdatenbank und verwendet die Endung der Skript-Datei (z.B. .pl) als Suchargument. Der durch den Unterschlüssel

Shell\ExecCGI\Command oder, falls dieser nicht existiert, Shell\Open\Command definierte Befehl wird zum Öffnen der Skript-Datei verwendet. Wenn die Registrierungsschlüssel nicht gefunden werden, dann verwendet der Apache die Option Script.

### **Sicherheit**

Seien Sie vorsichtig, wenn Sie ScriptInterpreterSource Registry bei Verzeichnissen verwenden, auf die eine **ScriptAlias**-Anweisung zeigt, denn der Apache wird versuchen, **jede** Datei innerhalb des Verzeichnisses auszuführen. Die Einstellung Registry kann unerwünschte Programmaufrufe bei Dateien verursachen, die üblicherweise nicht ausgeführt werden. Auf den meisten Windows-Systemen beispielsweise startet der voreingestellte Öffnen-Befehl für .htm-Dateien den Microsoft Internet Explorer, so dass jede HTTP-Anfrage nach einer existierenden .htm-Datei im Skript-Verzeichnis den Browser auf dem Server im Hintergrund starten würde. Dies ist leichte Art und Weise, Ihr System binnen etwa einer Minute zum Absturz zu bringen.

Die seit Apache 2.0 neue Option Registry-Strict macht das gleiche wie Registry, verwendet jedoch nur den Unterschlüssel Shell\ExecCGI\Command. Der Schlüssel ExecCGI ist gewöhnlich nicht voreingestellt. Er muss manuell in der Windows-Registrierungsdatenbank eingerichtet werden und verhindert dann versehentlich Programmaufrufe auf Ihrem System.



<b>Beschreibung:</b>	E-Mail-Adresse, die der Server in Fehlermeldungen einfügt, welche an den Client gesendet werden
<b>Syntax:</b>	ServerAdmin <i>E-Mail-Adresse</i>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core

`ServerAdmin` legt die E-Mail-Adresse fest, die der Server in jede Fehlermeldung einfügt, die er an den Client zurückschickt.

Es kann sich lohnen, hierfür eine reservierte Adresse anzugeben, z.B.

```
ServerAdmin www-admin@foo.example.com
```

da Anwender nicht unbedingt erwähnen, dass sie vom Server sprechen!



<b><u>Beschreibung:</u></b>	Alternativer Name für einen Host, der verwendet wird, wenn Anfragen einem namensbasierten virtuellen Host zugeordnet werden
<b><u>Syntax:</u></b>	<code>ServerAlias <i>Hostname</i> [<i>Hostname</i>] ...</code>
<b><u>Kontext:</u></b>	Virtual Host
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive `ServerAlias` bestimmt die alternativen Namen eines Hosts zur Verwendung mit [namensbasierten virtuellen Hosts](#).

```
<VirtualHost *>
ServerName server.domain.com
ServerAlias server server2.domain.com server2
# ...
</VirtualHost>
```

## Siehe auch

- [Apache-Dokumentation zu virtuellen Hosts](#)



<b>Beschreibung:</b>	Rechnername und Port, die der Server dazu verwendet, sich selbst zu identifizieren
<b>Syntax:</b>	<code>ServerName voll-qualifizierter-Domainname[:port]</code>
<b>Kontext:</b>	Serverkonfiguration, Virtual Host
<b>Status:</b>	Core
<b>Modul:</b>	core
<b>Kompatibilität:</b>	Diese Direktive löst in Version 2.0 die Funktionalität der Direktive <code>Port</code> aus Version 1.3 ab.

Die Direktive `ServerName` bestimmt den Rechnernamen und Port, den der Server dazu verwendet, sich selbst zu identifizieren. Diese werden bei der Erstellung von Umleitungs-URLs benötigt. Wenn beispielsweise der Name der Maschine, die den Webserver beherbergt, `simple.example.com` lautet, die Maschine jedoch auch einen DNS-Alias `www.example.com` besitzt und Sie den Webserver so identifizieren möchten, sollten Sie die folgende Anweisung verwenden:

```
ServerName www.example.com:80
```

Wenn kein `ServerName` angegeben wurde, dann versucht der Server den Rechnernamen mittels eines Reverse-Lookup herzuleiten. Wenn kein Port in der `ServerName`-Anweisung angegeben wurde, dann verwendet der Server den Port der eingegangenen Anfrage. Für eine optimale Zuverlässigkeit und Berechenbarkeit sollten Sie einen eindeutigen Rechnernamen und Port angeben, in dem Sie die Direktive `ServerName` verwenden.

Wenn Sie [namensbasierte virtuelle Hosts](#) verwenden, gibt `ServerName` innerhalb eines `<VirtualHost>`-Abschnitts an,

welcher Hostname im Host :-Header der Anfrage auftauchen muss, damit sie diesem virtuellen Host zugeordnet wird.

Lesen Sie bitte die Beschreibung der Direktive [UseCanonicalName](#) für Einstellungen, die bestimmen, ob selbstreferenzierende URLs (z.B. vom Modul [mod\\_dir](#)) auf den angegebenen Port zeigen oder auf die Portnummern die in der Anfrage des Clients angegeben ist.

## Siehe auch

- [Probleme bezüglich DNS und Apache](#)
- [Apache-Dokumentation zu virtuellen Hosts](#)
- [UseCanonicalName](#)
- [NameVirtualHost](#)
- [ServerAlias](#)



<b><u>Beschreibung:</u></b>	Veralteter URL-Pfad für einen namensbasierten virtuellen Host, auf den von einem inkompatiblen Browser zugegriffen wird
<b><u>Syntax:</u></b>	ServerPath <i>URL -Pfad</i>
<b><u>Kontext:</u></b>	Virtual Host
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive **ServerPath** legt den veralteten (*Anm.d.Ü.:* Gemeint ist eigentlich "Altlast" aufgrund antiquierter Clients.) URL-Pfad eines Hosts zur Verwendung mit [namensbasierten virtuellen Hosts](#) fest.

## Siehe auch

- [Apache-Dokumentation zu virtuellen Hosts](#)



<b>Beschreibung:</b>	Basisverzeichnis der Serverinstallation
<b>Syntax:</b>	ServerRoot <i>Verzeichnis</i>
<b>Voreinstellung:</b>	ServerRoot /usr/local/apache
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive **ServerRoot** bestimmt das Verzeichnis, in dem der Server installiert ist. Üblicherweise enthält es die Unterverzeichnisse `conf/` und `logs/`. Relative Pfadangaben anderer Direktiven (wie z.B. **Include** oder **LoadModule**) werden relativ zu diesem Verzeichnis betrachtet.

### Beispiel

```
ServerRoot /home/httpd
```

### Siehe auch

- [Die httpd-Option -d](#)
- [Sicherheitshinweise](#) für Informationen, wie die Rechte auf das **ServerRoot**-Verzeichnis richtig gesetzt werden



<b>Beschreibung:</b>	Konfiguriert die Fußzeile von servergenerierten Dokumenten
<b>Syntax:</b>	ServerSignature On Off EMail
<b>Voreinstellung:</b>	ServerSignature Off
<b>Kontext:</b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b>AllowOverride:</b>	All
<b>Status:</b>	Core
<b>Modul:</b>	core

Die Direktive **ServerSignature** ermöglicht die Gestaltung einer unter servergenerierten Dokumenten (z.B. Fehlerdokumente, FTP-Verzeichnislisten von **mod\_proxy**, **mod\_info**-Ausgaben, ...) angefügten Fußzeile. Ein möglicher Grund für die Aktivierung einer solchen Fußzeile ist, dass der Anwender bei einer Kette von Proxy-Servern oft keine Möglichkeit hat, zu erkennen, welcher der verketteten Server gegenwärtig die zurückgegebene Fehlermeldung produziert hat.

Die (Vor-)Einstellung **Off** unterdrückt die Fußzeile (und ist damit kompatibel zum Verhalten des Apache 1.2 und früher). Die Einstellung **On** fügt schlicht eine Zeile mit der Versionsnummer des Servers und dem Servernamen (**ServerName**) des bedienenden virtuellen Hosts an. Die Einstellung **EMail** erstellt zusätzlich einen "mailto:"-Verweis zum Serveradministrator (**ServerAdmin**) des referenzierten Dokuments.

Ab Version 2.0.44 werden die Details der angegebenen Versionsnummer des Servers von der Direktive **ServerTokens** kontrolliert.

**Siehe auch**

- ServerTokens



<b>Beschreibung:</b>	Konfiguriert den HTTP-Response-Header Server
<b>Syntax:</b>	ServerTokens Major   Minor   Min[imal]   Prod[uctOnly]
<b>Voreinstellung:</b>	ServerTokens Full
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	Core
<b>Modul:</b>	core

die Direktive steuert, ob der Response-Header Server, der an den Client zurückgesendet wird, eine Beschreibung des allgemeinen Betriebssystemtyps des Servers wie auch Informationen über einkompilierte Module enthält.

### **ServerTokens Prod[uctOnly]**

Der Server sendet (z.B.): Server : Apache

### **ServerTokens Major**

Der Server sendet (z.B.): Server : Apache/2

### **ServerTokens Minor**

Der Server sendet (z.B.): Server : Apache/2.0

### **ServerTokens Min[imal]**

Der Server sendet (z.B.): Server : Apache/2.0.41

### **ServerTokens OS**

Der Server sendet (z.B.): Server : Apache/2.0.41  
(Unix)

### **ServerTokens Full (oder nicht angegeben)**

Der Server sendet (z.B.): Server : Apache/2.0.41  
(Unix) PHP/4.2.2 MyMod/1.2

Diese Einstellung gilt für den gesamten Server und kann nicht auf Virtual-Host-Basis aktiviert oder deaktiviert werden.

Ab Version 2.0.44 steuert diese Direktive auch die Informationen, die durch die Direktive [ServerSignature](#) angeboten werden.

## Siehe auch

- [ServerSignature](#)



<b><u>Beschreibung:</u></b>	Erzwingt die Verarbeitung aller passenden Dateien durch einen Handler
<b><u>Syntax:</u></b>	SetHandler <i>Handlername</i>   None
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Seit Apache 2.0 im Core

Wenn die Direktive innerhalb einer `.htaccess`-Datei oder in einem `<Directory>`- oder `<Location>`-Abschnitt angegeben wird, erzwingt sie, dass alle entsprechenden Dateien von dem durch *Handlername* angegebenen [Handler](#) analysiert werden. Wenn Sie beispielsweise ein Verzeichnis haben, dessen Dateien unabhängig von der Endung gänzlich als Image-Maps interpretiert werden sollen, können Sie folgendes in eine `.htaccess`-Datei in dem Verzeichnis schreiben:

```
SetHandler imap-file
```

Noch ein Beispiel: wenn Sie den Server immer, wenn die URL `http://servername/status` aufgerufen wird, einen Statusbericht anzeigen lassen möchten, dann können Sie folgendes in die `httpd.conf` schreiben:

```
<Location /status>  
  SetHandler server-status  
</Location>
```

Sie können eine zuvor definierte `SetHandler`-Anweisung aufheben, indem Sie den Wert `None` verwenden.

## Siehe auch

- [AddHandler](#)



<b><u>Beschreibung:</u></b>	Bestimmt die Filter, die Client-Anfragen und POST-Eingaben verarbeiten
<b><u>Syntax:</u></b>	<code>SetInputFilter <i>Filter</i>[;<i>Filter</i>...]</code>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive `SetInputFilter` bestimmt den oder die Filter, die Client-Anfragen und POST-Eingaben verarbeiten, wenn sie vom Server empfangen werden. Diese gelten zusätzlich zu anderweitig definierten Filtern, einschließlich denen der Direktive `AddInputFilter`.

Wenn mehr als ein Filter angegeben wird, dann müssen diese durch Semikolon voneinander getrennt in der Reihenfolge angegeben werden, in der sie die Daten verarbeiten sollen.

**Siehe auch**

- [Filter](#)-Dokumentation



<b><u>Beschreibung:</u></b>	Bestimmt die Filter, die Antworten des Servers verarbeiten
<b><u>Syntax:</u></b>	<code>SetOutputFilter <i>Filter</i> [;<i>Filter</i>...]</code>
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis, .htaccess
<b><u>AllowOverride:</u></b>	FileInfo
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive `SetOutputFilter` bestimmt die Filter, die Antworten des Servers verarbeiten, bevor sie an den Client gesendet werden. Diese gelten zusätzlich zu anderweitig definierten Filtern, einschließlich denen der Direktive `AddOutputFilter`.

Die folgende Konfiguration verarbeitet zum Beispiel alle Dateien im Verzeichnis `/www/data` als Server Side Includes.

```
<Directory /www/data/>
  SetOutputFilter INCLUDES
</Directory>
```

Wenn mehr als ein Filter angegeben wird, dann müssen diese durch Semikolon voneinander getrennt in der Reihenfolge angegeben werden, in der sie die Daten verarbeiten sollen.

## Siehe auch

- [Filter](#)-Dokumentation



<b><u>Beschreibung:</u></b>	Zeitspanne, die der Server auf verschiedene Ereignisse wartet, bevor er die Anfrage abbricht
<b><u>Syntax:</u></b>	TimeOut <i>Sekunden</i>
<b><u>Voreinstellung:</u></b>	TimeOut 300
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

Die Direktive `TimeOut` definiert derzeit die Zeitspanne, die der Apache auf drei Dinge wartet:

1. Die gesamte Zeitspanne, die benötigt wird, um eine GET-Anfrage zu empfangen.
2. Die Zeitspanne zwischen dem Empfang von TCP-Paketen einer POST- oder PUT-Anfrage.
3. Die Zeitspanne zwischen ACKs bei der Übermittlung der TCP-Pakete der Antwort.

Wir haben vor, diese Zeitspannen in Zukunft separat konfigurierbar zu machen. Vor Version 1.2 war der Zeitgeber auf 1200 voreingestellt, wurde dann aber auf 300 herabgesetzt, was immer noch weit mehr ist, als in den meisten Situationen benötigt wird. Die Voreinstellung wurde nicht weiter herabgesetzt, da gelegentlich noch Stellen im Code existieren können, wo der Zeitgeber nicht zurückgesetzt wird, wenn ein Paket verschickt wird.



<b><u>Beschreibung:</u></b>	Legt das Serververhalten bei TRACE-Anfragen fest
<b><u>Syntax:</u></b>	TraceEnable [on off extended]
<b><u>Voreinstellung:</u></b>	TraceEnable on
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core
<b><u>Kompatibilität:</u></b>	Seit den Versionen 1.3.34 und 2.0.55

Diese Direktive überschreibt das Verhalten bei TRACE sowohl für den Hauptserver als auch für `mod_proxy`. Die Voreinstellung `TraceEnable on` erlaubt TRACE-Anfragen gemäß RFC 2616, welcher das Mitschicken von Request-Bodies verbietet. `TraceEnable off` veranlasst den Hauptserver und `mod_proxy`, dem Client mit dem Fehler 405 (Methode nicht erlaubt) antworten.

Mit Hilfe der nicht konformen Direktive `TraceEnable extended` können Request-Bodies zugelassen werden. Das ist allerdings ausschließlich für Test- und Diagnosezwecke gedacht. Der Hauptserver (als Ursprungsserver) (*Anm.d.Ü.:* damit ist der Normalbetrieb und nicht der Einsatz als Proxy-Server gemeint) limitiert den Request-Body auf 64K (plus 8K für die Chunk-Header, wenn `Transfer-Encoding: chunked` verwendet wird) und reflektiert den vollen Headersatz sowie alle Chunk-Header in seiner Antwort. Im Betrieb als Proxyserver wird der Request-Body nicht auf 64K begrenzt.



<b><u>Beschreibung:</u></b>	Bestimmt, wie der Server seinen eigenen Namen und Port ermittelt
<b><u>Syntax:</u></b>	UseCanonicalName On Off DNS
<b><u>Voreinstellung:</u></b>	UseCanonicalName On
<b><u>Kontext:</u></b>	Serverkonfiguration, Virtual Host, Verzeichnis
<b><u>Status:</u></b>	Core
<b><u>Modul:</u></b>	core

In vielen Situationen muss der Apache eine *selbstreferenzierende* URL -- d.h. eine URL, die auf den selben Server zurück verweist -- zusammenbauen. Bei `UseCanonicalName On` verwendet der Apache den Hostnamen und Port, der in der [ServerName](#)-Anweisung angegeben ist, um den kanonischen Namen des Servers zu erstellen. Dieser Name wird in allen selbstreferenzierenden URLs sowie in CGI-Skripten für die Werte von `SERVER_NAME` und `SERVER_PORT` verwendet.

Bei `UseCanonicalName Off` bildet der Apache selbstreferenzierende URLs, indem er den vom Client übermittelten Hostnamen und Port verwendet, sofern diese vorhanden sind (andernfalls wird der kanonische Name, wie oben beschrieben, benutzt). Die Werte sind die gleichen, die zur Anwendung von [namensbasierten virtuellen Hosts](#) verwendet werden, und sie sind mit den gleichen Clients verfügbar (*Anm.d.Ü.:* , die auch in der Lage sind, auf namensbasierte virtuelle Hosts zuzugreifen, d.h. einen Host-Header mitschicken). Die CGI-Variablen `SERVER_NAME` und `SERVER_PORT` werden ebenfalls aus den vom Client angebotenen Werten erstellt.

Ein Intranet-Server, auf den Anwender mit kurzen Namen wie `www` zugreifen, ist ein Beispiel, wo dies sinnvoll sein kann. Sie werden bemerken, dass der Apache den Benutzer auf

`http://www.domain.com/splat/` umleitet, wenn dieser einen Kurznamen und eine URL, die einem Verzeichnis entspricht, ohne abschließenden Schrägstrich eingibt, wie z.B.

`http://www/splat`. Wenn Sie Authentisierung aktiviert haben, bewirkt dies, dass der Benutzer sich zweimal identifizieren muss (einmal für `www` und noch einmal für `www.domain.com` -- lesen Sie für weitere Informationen [die FAQ zu diesem Thema](#)). Wenn `UseCanonicalName` jedoch auf `Off` gesetzt ist, denn wird der Apache zu `http://www/splat/` umleiten.

Es existiert noch eine dritte Option, `UseCanonicalName DNS`, die für den Betrieb von IP-basierten Massen-Virtual-Hosts gedacht ist, um antiquierte Clients zu unterstützen, die keinen `Host :-` Header bereit stellen. Um selbstreferenzierende URLs zu ermitteln, führt der Apache bei dieser Option ein Reverse-DNS-Lookup auf die IP-Adresse des Servers aus, zu der der Client Verbindung aufgenommen hat.

### **Warnung**

Wenn CGI-Skripte Vermutungen aufgrund des Wertes von `SERVER_NAME` anstellen, können sie durch diese Option fehlschlagen. Clients steht es im Wesentlichen frei, einen Wert für den Hostnamen anzugeben, wie er will. Wenn das CGI-Skript `SERVER_NAME` jedoch lediglich dazu verwendet, selbstreferenzierende URLs zu erstellen, sollte das gerade noch in Ordnung sein.

### **Siehe auch**

- [ServerName](#)
- [Listen](#)



**Beschreibung:** Enthält Direktiven, die nur auf bestimmte Hostnamen oder IP-Adressen angewendet werden

**Syntax:** `<VirtualHost Adresse[:Port]  
[Adresse[:Port]] ...> ...  
</VirtualHost>`

**Kontext:** Serverkonfiguration

**Status:** Core

**Modul:** core

`<VirtualHost>` und `</VirtualHost>` werden dazu verwendet, eine Gruppe von Direktiven zusammenzufassen, die nur auf einen bestimmten virtuellen Host angewendet werden. Jede Direktive, die im Virtual-Host-Kontext zulässig ist, kann verwendet werden. Wenn der Server eine Anfrage für ein bestimmtes Dokument eines bestimmten virtuellen Hosts empfängt, dann benutzt er die im `<VirtualHost>`-Container enthaltenen Konfigurationsanweisungen. *Adresse* kann sein:

- Die IP-Adresse des virtuellen Hosts.
- Ein voll qualifizierter Domainname für die IP-Adresse des virtuellen Hosts.
- Das Zeichen `*`, welches nur in Kombination mit `NameVirtualHost *` verwendet wird, um allen IP-Adressen zu entsprechen.
- Die Zeichenkette `_default_`, die nur mit IP-basierten virtuellen Hosts verwendet wird, um nicht zugewiesene IP-Adressen aufzufangen.

## Beispiel

```
<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-Adressen müssen in eckigen Klammern angegeben werden, da die optionale Portnummer sonst nicht erkannt werden kann. Hier ein IPv6-Beispiel:

```
<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>
```

Jeder virtuelle Host muss einer anderen IP-Adresse, einem anderen Port oder einem anderen Hostnamen für den Server entsprechen. Im ersten Fall muss die Servermaschine so eingerichtet sein, dass sie IP-Pakete für mehrere Adressen akzeptiert. (Wenn der Rechner nicht mehrere Netzwerkkarten besitzt, kann dies mit dem Befehl `ifconfig alias` durchgeführt werden -- sofern Ihr Betriebssystem das unterstützt).

### Anmerkung

Die Verwendung von `<VirtualHost>` beeinflusst **nicht**, an welchen Adressen der Apache lauscht. Sie müssen mit [Listen](#) sicherstellen, dass der Apache an der richtigen Adresse lauscht.

Bei der Verwendung IP-basierter virtueller Hosts kann der spezielle Name `_default_` benutzt werden. In diesem Fall weist der Apache jede IP-Adresse diesem virtuellen Host zu, die nicht explizit in einem anderen virtuellen Host angegeben ist. Falls kein virtueller Host `_default_` angegeben ist, wird die "Hauptserver"-Konfiguration, die aus allen Definitionen außerhalb der Virtual-

Host-Abschnitte besteht, für nicht passende IPs verwendet. (Beachten Sie jedoch, dass eine IP-Adresse die zu einer [NameVirtualHost](#)-Anweisung passt, weder den "Hauptserver" noch den virtuellen Host `_default_` verwendet. Lesen Sie für weitere Details die Dokumentation zu [namensbasierten virtuellen Hosts](#).)

Sie können einen speziellen `:Port` angeben, um den entsprechenden Port zu wechseln. Falls nicht angegeben, wird er auf den gleichen Port voreingestellt, wie die letzte [Listen](#)-Anweisung des Hauptservers. Sie können auch `:*` angeben, um alle Ports dieser Adresse zu akzeptieren. (Dies wird zusammen mit `_default_` empfohlen.)

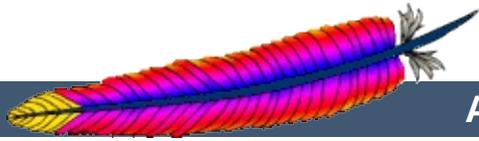
### **Sicherheit**

Lesen Sie das Dokument [Sicherheitshinweise](#) für Details, warum Ihre Sicherheit gefährdet sein kann, wenn das Verzeichnis, in dem Protokolldateien gespeichert werden, für jemanden anderes als den Benutzer beschreibbar ist, der den Server gestartet hat.

### **Siehe auch**

- [Apache-Dokumentation zu virtuellen Hosts](#)
- [Probleme bezüglich DNS und Apache](#)
- [Bestimmen, welche Adressen und Ports der Apache verwendet](#)
- [Wie die Abschnitte `<Directory>`, `<Location>` und `<Files>` arbeiten](#) für eine Erläuterung, wie diese verschiedenen Abschnitte miteinander kombiniert werden, wenn eine Anfrage empfangen wird

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

## Allgemeine Direktiven der Apache-MPMs

Diese Übersetzung ist möglicherweise nicht mehr aktuell. Bitte prüfen Sie die englische Version auf die neuesten Änderungen.

**Beschreibung:** Eine Sammlung von Direktiven, die in mehr als einem Multi-Processing-Modul (MPM) implementiert sind.

**Status:** MPM



<b>Beschreibung:</b>	Vom Apache verwendete Methode zur Serialisierung mehrerer Kindprozesse, die Anfragen an Netzwerk-Sockets entgegennehmen.
<b>Syntax:</b>	AcceptMutex Default   <i>Methode</i>
<b>Voreinstellung:</b>	AcceptMutex Default
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">leader</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Die Direktive `AcceptMutex` bestimmt die Methode, die der Apache zur Serialisierung mehrerer Kindprozesse verwendet, welche Anfragen an Netzwerk-Sockets entgegennehmen. Vor Apache 2.0 war diese Methode nur zur Kompilierungszeit einstellbar. Die optimale Methode ist sehr stark von der Architektur und Plattform abhängig. Lesen Sie bitte [Performance-Hinweise](#) für weitere Details.

Wenn die Direktive auf `Default` eingestellt ist, dann wird die zur Kompilierungszeit gewählte Voreinstellung verwendet. Weitere mögliche Methoden sind unten angegeben. Beachten Sie, dass nicht alle Methoden auf allen Plattformen verfügbar sind. Wird eine Methode angegeben, die nicht verfügbar ist, dann wird eine Nachricht in das Fehlerprotokoll geschrieben, welche die verfügbaren Methoden auflistet.

### **flock**

verwendet die Systemfunktion `flock(2)`, um die durch die [LockFile](#)-Direktive definierte Datei zu sperren.

### **fcntl**

verwendet die Systemfunktion `fcntl(2)`, um die durch die

[LockFile](#)-Direktive definierte Datei zu sperren.

**posixsem**

verwendet POSIX-kompatible Semaphore, um den Mutex zu implementieren.

**pthread**

verwendet gemäß der POSIX-Thread-Spezifikation implementierte POSIX-Mutexe.

**sysvsem**

verwendet Semaphore des SysV-Typs, um den Mutex zu implementieren.

Um die bei der Kompilierung gewählte Voreinstellung für Ihr System herauszufinden, können Sie Ihr [LogLevel](#) auf debug setzen. Dann wird der voreingestellte [AcceptMutex](#) ins [ErrorLog](#) geschrieben.



<b>Beschreibung:</b>	Bestimmt den nicht-privilegierten Account auf BS2000-Maschinen
<b>Syntax:</b>	<code>BS2000Account Account</code>
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<code>perchild</code> , <code>prefork</code>
<b>Kompatibilität:</b>	Nur für BS2000-Maschinen verfügbar

Die Direktive `BS2000Account` ist nur für BS2000-Hosts verfügbar. Sie muss dazu verwendet werden, den Account für den nicht-privilegierten Apache-Server-Benutzer (der durch die Direktive `User` eingestellt wird) zu bestimmen. Dies wird vom BS2000-POSIX-Subsystem benötigt (um die zugrundeliegende BS2000-Anwendungsumgebung mittels eines Sub-LOGONs zu wechseln), um zu verhindern, dass CGI-Skripte auf Ressourcen des privilegierten Accounts zugreifen, der den Server gestartet hat, üblicherweise SYSROOT.

### Anmerkung

Es kann nur eine `BS2000Account`-Direktive verwendet werden.

### Siehe auch

- [Apache-EBCDIC-Portierung](#)



<b>Beschreibung:</b>	Verzeichnis, in das der Apache zu wechseln versucht, bevor er einen Hauptspeicherauszug erstellt
<b>Syntax:</b>	CoreDumpDirectory <i>Verzeichnis</i>
<b>Voreinstellung:</b>	Für die Voreinstellung siehe Beschreibung
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_winnt</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Dies beeinflusst das Verzeichnis, in welches der Apache zu wechseln versucht, bevor er einen Hauptspeicherauszug (*Anm.d.Ü.:* einen so genannten Core-Dump) erstellt. Die Voreinstellung ist das [ServerRoot](#)-Verzeichnis. Da dieses jedoch nicht für den Benutzer beschreibbar sein soll, unter dem der Server läuft, werden normalerweise keine Hauptspeicherauszüge geschrieben. Wenn Sie zum Debuggen einen Hauptspeicherauszug haben möchten, können Sie ihn mit dieser Direktive an einem anderen Ort ablegen lassen.

### Hauptspeicherauszüge unter Linux

Wenn Apache als root startet und zu einem anderen Benutzer wechselt, *deaktiviert* der Linux-Kernel Hauptspeicherauszüge auch dann, wenn der Prozess in dem Verzeichnis schreiben darf. Ab Linux 2.4 reaktiviert Apache (ab 2.0.46) Hauptspeicherauszüge wieder, jedoch nur dann, wenn Sie explizit [CoreDumpDirectory](#) konfigurieren.



<b><u>Beschreibung:</u></b>	Aktiviert einen Hook, der nach einem Absturz noch Ausnahmefehler behandeln lassen kann
<b><u>Syntax:</u></b>	<code>EnableExceptionHook On Off</code>
<b><u>Voreinstellung:</u></b>	<code>EnableExceptionHook Off</code>
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<a href="#"><u>leader</u></a> , <a href="#"><u>perchild</u></a> , <a href="#"><u>prefork</u></a> , <a href="#"><u>threadpool</u></a> , <a href="#"><u>worker</u></a>
<b><u>Kompatibilität:</u></b>	Verfügbar seit Version 2.0.49

Diese Direktive ist aus Sicherheitsgründen nur verfügbar, wenn der Server mit der Option `--enable-exception-hook` konfiguriert wurde. Sie aktiviert einen Hook, der es externen Modulen erlaubt, sich dort einzuhängen und nach dem Absturz eines Kindprozesses noch Aktionen durchzuführen.

Es existieren bereits zwei Module, `mod_whatkilledus` und `mod_backtrace`, welche diesen Hook verwenden. Weitere Informationen hierzu finden Sie auf Jeff Trawicks [EnableExceptionHook-Seite](#).



## Group Directive

<b>Beschreibung:</b>	Benutzergruppe, unter welcher der Server Anfragen beantwortet
<b>Syntax:</b>	Group <i>Unix-Gruppe</i>
<b>Voreinstellung:</b>	Group #-1
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpmt_os2</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>
<b>Kompatibilität:</b>	Seit Apache 2.0 nur in der globalen Server-Konfiguration gültig

Die Direktive **Group** bestimmt die Benutzergruppe, unter welcher der Server Anfragen beantwortet. Um diese Direktive zu verwenden, muss der Server als `root` gestartet werden. Wenn Sie den Server unter einem nicht-root-Benutzer starten, wird er nicht zur angegebenen Gruppe wechseln können und statt dessen weiter mit der Gruppe des ursprünglichen Benutzers laufen. *Unix-Gruppe* kann sein:

### Ein Gruppenname

Verweist auf die durch den Namen angegebene Gruppe.

### # gefolgt von einer Gruppennummer.

Verweist auf die durch ihre Nummer angegebene Gruppe.

### Beispiel

```
Group www-group
```

Es wird empfohlen, dass Sie eine neue Gruppe speziell zum Betrieb des Servers erstellen. Einige Administratoren verwenden den Benutzer `nobody`. Dies ist jedoch nicht immer möglich oder gewünscht.

---

## Sicherheit

Setzen Sie `Group` (oder `User`) nicht auf `root`, solange Sie nicht ganz genau wissen, was Sie tun und welche Gefahren Sie eingehen.

Wichtiger Hinweis: Die Verwendung der Direktive innerhalb von `<VirtualHost>` wird nicht länger unterstützt. Benutzen Sie `SuexecUserGroup` um Ihren Server für `suexec` einzurichten.

## Anmerkung

Obwohl die Direktive `Group` in den MPMs `beos` und `mpm_os2` existiert, ist sie dort tatsächlich eine Leeranweisung und existiert nur aus Kompatibilitätsgründen.



<b><u>Beschreibung:</u></b>	IP-Adressen und Ports, an denen der Server lauscht
<b><u>Syntax:</u></b>	Listen [ <i>IP-Adresse:</i> ]Port
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<u>beos</u> , <u>leader</u> , <u>mpm_netware</u> , <u>mpm_winnt</u> , <u>mpmt_os2</u> , <u>perchild</u> , <u>prefork</u> , <u>threadpool</u> , <u>worker</u>
<b><u>Kompatibilität:</u></b>	Seit Apache 2.0 vorgeschrieben

Die Direktive **Listen** weist den Apache an, nur an den angegebenen IP-Adressen oder Ports zu lauschen. Standardmäßig antwortet er auf alle Anfragen an allen IP-Interfaces. **Listen** ist nun eine notwendige Anweisung. Wenn sie nicht in der Konfigurationsdatei enthalten ist, wird der Server-Start fehlschlagen. Dies ist eine Änderung gegenüber früheren Versionen des Apache.

Die Direktive **Listen** weist den Server an, ankommende Anfragen am angegebenen Port oder der Kombination aus Adresse und Port entgegenzunehmen. Wenn nur eine Portnummer angegeben ist, dann lauscht der Server am angegebenen Port an allen Interfaces. Wenn sowohl eine IP-Adresse als auch ein Port angegeben sind, dann lauscht der Server am angegebenen Port und Interface.

Es können mehrere **Listen**-Anweisungen verwendet werden, um eine Reihe von Adressen und Port anzugeben, an denen gelauscht werden soll. Der Server antwortet auf Anfragen von jedem der aufgeführten Adressen und Ports.

Um beispielsweise den Server Verbindungen an den beiden Ports 80 und 8000 annehmen zu lassen, verwenden Sie:

```
Listen 80
Listen 8000
```

Um den Server Verbindungen an zwei angegebenen Interfaces und Ports annehmen zu lassen, verwenden Sie:

```
Listen 192.170.2.1:80
Listen 192.170.2.5:8000
```

IPv6-Adressen müssen wie in dem folgenden Beispiel in eckige Klammern eingeschlossen werden:

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

### Fehlermöglichkeit

Mehrere `Listen`-Direktiven für gleiche IP-Adresse und Port führen zur Fehlermeldung `Address already in use` (*Anm.d.Ü.:* Adresse schon in Benutzung).

### Siehe auch

- [DNS-Probleme](#)
- [Bestimmen, welche Adressen und Ports der Apache verwendet](#)



<b><u>Beschreibung:</u></b>	Maximale Länge der Warteschlange schwebender Verbindungen
<b><u>Syntax:</u></b>	ListenBacklog <i>backlog</i>
<b><u>Voreinstellung:</u></b>	ListenBacklog 511
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<u>beos</u> , <u>leader</u> , <u>mpm_netware</u> , <u>mpm_winnt</u> , <u>mpmt_os2</u> , <u>perchild</u> , <u>prefork</u> , <u>threadpool</u> , <u>worker</u>

Die maximale Länge der Warteschlange schwebender Verbindungen. Üblicherweise ist keine Feineinstellung notwendig oder sinnvoll, auf einigen System kann es jedoch gewünscht sein, diesen Wert bei TCP-SYN-Angriffen zu erhöhen. Beachten Sie auch die Beschreibung des backlog-Parameters der Systemfunktion `listen(2)`.

Der Wert wird vom Betriebssystem oft auf eine niedrigere Einstellung begrenzt. Dies variiert von Betriebssystem zu Betriebssystem. Beachten Sie auch, dass viele Betriebssysteme nicht genau beachten, was für backlog angegeben ist, jedoch einen Wert basierend auf der Angabe (normalerweise jedoch größer als diese) verwenden.



<b><u>Beschreibung:</u></b>	Ablageort der Lock-Datei für die Serialisierung von entgegengenommenen Anfragen
<b><u>Syntax:</u></b>	LockFile <i>Dateiname</i>
<b><u>Voreinstellung:</u></b>	LockFile logs/accept.lock
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<u>leader</u> , <u>perchild</u> , <u>prefork</u> , <u>threadpool</u> , <u>worker</u>

Die Direktive **LockFile** legt den Pfad zur Lock-Datei fest, die verwendet wird, wenn der Apache mit einer der **AcceptMutex**-Einstellungen `fcntl` oder `flock` verwendet wird. Die Anweisung sollte normalerweise bei der Voreinstellung belassen werden. Der Hauptgrund, sie zu ändern, ist, wenn das `logs`-Verzeichnis auf einem per NFS-eingebundenen Laufwerk liegt, da **die Lock-Datei auf einer lokalen Platte abgelegt sein muss**. Die PID (*Anm.d.Ü.:* Prozess-ID) des Hauptserverprozesses wird automatisch an den Dateinamen angehängt.

### Sicherheit

Es ist am besten, die Ablage in einem allgemein (*Anm.d.Ü.:* für jedermann) beschreibbaren Verzeichnis wie `/var/tmp` zu vermeiden, da ein Denial-of-Service-Angriff gestartet werden könnte und der Server am Start gehindert werden könnte, indem eine Lock-Datei mit dem gleichen Namen erstellt wird, wie der Server sie zu erstellen versuchen würde.

### Siehe auch

- [AcceptMutex](#)



<b>Beschreibung:</b>	Maximale Anzahl der Kindprozesse, die zur Bedienung von Anfragen gestartet wird
<b>Syntax:</b>	<code>MaxClients <i>Anzahl</i></code>
<b>Voreinstellung:</b>	Für Details siehe Beschreibung
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<code>beos</code> , <code>leader</code> , <code>prefork</code> , <code>threadpool</code> , <code>worker</code>

Die Direktive `MaxClients` setzt die Grenze für die Anzahl gleichzeitig bedienter Anfragen. Jeder Verbindungsversuch oberhalb der `MaxClients`-Begrenzung wird üblicherweise in eine Warteschlange gestellt, bis zu einer Anzahl basierend auf der `ListenBacklog`-Anweisung. Sobald ein Kindprozess am Ende einer anderen Anfrage freigegeben wird, wird die Verbindung bedient.

Für Server ohne Thread-Unterstützung (z.B. `prefork`) wird `MaxClients` als maximale Anzahl der Kindprozesse verstanden, die zur Bedienung von Anfragen gestartet werden. Die Voreinstellung ist 256. Um diesen Wert zu erhöhen, muss auch `ServerLimit` angehoben werden.

Bei Servern mit Thread-Unterstützung und bei Hybrid-Servern (z.B. `beos` oder `worker`) begrenzt `MaxClients` die Gesamtzahl der Threads, die für die Bedienung von Anfragen verfügbar sind. Die Voreinstellung für `beos` ist 50. Bei Hybrid-MPMs ist die Voreinstellung 16 (`ServerLimit`) multipliziert mit dem Wert 25 (`ThreadsPerChild`). Um `MaxClients` auf einen Wert zu erhöhen, der mehr als 16 Prozesse erfordert, müssen Sie daher auch `ServerLimit` anheben.



<b><u>Beschreibung:</u></b>	Maximale Menge des Arbeitsspeichers, den die Haupt-Zuteilungsroutine verwalten darf, ohne <code>free()</code> aufzurufen
<b><u>Syntax:</u></b>	<code>MaxMemFree</code> <i>KBytes</i>
<b><u>Voreinstellung:</u></b>	<code>MaxMemFree</code> 0
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_netware</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a> , <a href="#">mpm_winnt</a>

Die Direktive `MaxMemFree` gibt die maximale Menge freier Kilobytes an, welche die Haupt-Zuteilungsroutine verwalten darf, ohne `free()` aufzurufen. Wenn keine Angabe gemacht wird, oder Null angegeben ist, wird dieser Wert nicht eingeschränkt.



## MaxRequestsPerChild-Direktive

<b>Beschreibung:</b>	Obergrenze für die Anzahl von Anfragen, die ein einzelner Kindprozess während seines Lebens bearbeitet
<b>Syntax:</b>	MaxRequestsPerChild <i>number</i>
<b>Voreinstellung:</b>	MaxRequestsPerChild 10000
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">leader</a> , <a href="#">mpm_netware</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Die Direktive `MaxRequestsPerChild` legt die Grenze für die Anzahl von Anfragen fest, die ein einzelner Kinprozess während seines Lebens bearbeitet. Nach `MaxRequestsPerChild` Anfragen stirbt der Kindprozess. Wenn `MaxRequestsPerChild` 0 ist, endet der Prozess niemals.

### Abweichende Voreinstellungen

Die Voreinstellung für [mpm\\_netware](#) und [mpm\\_winnt](#) ist 0.

Die Begrenzung von `MaxRequestsPerChild` auf einen Wert ungleich Null hat zwei vorteilhafte Auswirkungen:

- sie begrenzt die Menge an Arbeitsspeicher, die ein Prozess durch (versehentliche) Speicherlecks verbrauchen kann.
- das Festlegen einer endlichen Lebensdauer von Prozessen hilft, die Anzahl von Prozessen zu reduzieren, wenn die Serverlast zurückgeht.

### Anmerkung

Bei [KeepAlive](#)-Anfragen wird nur die erste Anfrage für diese

begrenzung gezählt. Eigentlich wird nur die Begrenzung für die Anzahl der *Verbindungen* pro Kindprozess geändert.



<b>Beschreibung:</b>	Maximale Anzahl unbeschäftigter Threads
<b>Syntax:</b>	MaxSpareThreads <i>Anzahl</i>
<b>Voreinstellung:</b>	Für Details siehe Beschreibung
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a> , <a href="#">perchild</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Maximale Anzahl unbeschäftigter Threads. Die verschiedenen MPMs behandeln diese Anweisung unterschiedlich.

Die Voreinstellung für [perchild](#) ist `MaxSpareThreads 10`. Das MPM überwacht die Anzahl der unbeschäftigten Threads auf der Basis einzelner Kindprozesse. Wenn zu viele unbeschäftigte Threads in einem Kindprozess existieren, beendet der Server Threads innerhalb dieses Kindprozesses.

Die Voreinstellung für [worker](#), [leader](#) und [threadpool](#) ist `MaxSpareThreads 250`. Diese MPMs behandeln Threads auf einer serverweiten Basis. Wenn zu viele unbeschäftigte Threads im Server existieren, dann werden solange Kindprozesse beendet, bis die Anzahl der unbeschäftigten Threads kleiner als der angegebene Wert ist.

Die Voreinstellung für [mpm\\_netware](#) ist `MaxSpareThreads 100`. Da dieses MPM nur einen einzigen Prozess ausführt, ist die Zählung überschüssiger Threads ebenfalls serverweit.

[beos](#) and [mpmt\\_os2](#) arbeiten ähnlich wie [mpm\\_netware](#). Die Voreinstellung für [beos](#) ist `MaxSpareThreads 50`. Die Voreinstellung für [mpmt\\_os2](#) ist 10.

## Restriktionen

Der Wertebereich von `MaxSpareThreads` ist eingeschränkt. Apache korrigiert den angegebenen Wert automatisch gemäß den folgenden Regeln:

- `perchild` verlangt, dass `MaxSpareThreads` kleiner oder gleich `ThreadLimit` ist.
- `mpm_netware` verlangt einen Wert größer als `MinSpareThreads`.
- Bei `leader`, `threadpool` und `worker` muss der Wert größer oder gleich der Summe aus `MinSpareThreads` und `ThreadsPerChild` sein.

## Siehe auch

- `MinSpareThreads`
- `StartServers`



<b>Beschreibung:</b>	Minimale Anzahl unbeschäftigter Threads, die zur Bedienung von Anfragespitzen zur Verfügung stehen
<b>Syntax:</b>	MinSpareThreads <i>Anzahl</i>
<b>Voreinstellung:</b>	Für Details siehe Beschreibung
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_netware</a> , <a href="#">mpmt_os2</a> , <a href="#">perchild</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Minimale Anzahl unbeschäftigter Threads, um Anfragespitzen zu bedienen. Die verschiedenen MPMs behandeln die Anweisung unterschiedlich.

[perchild](#) verwendet die Voreinstellung `MinSpareThreads 5` und überwacht die Anzahl der unbeschäftigten Threads auf der Basis einzelner Kindprozesse. Wenn in einem Kindprozess nicht genügend unbeschäftigte Threads vorhanden sind, erstellt der Server neue Threads innerhalb dieses Kindprozesses. Wenn Sie also [NumServers](#) auf 10 und [MinSpareThreads](#) auf einen Wert von 5 setzen, haben Sie mindestens 50 unbeschäftigte Threads auf Ihrem System.

[worker](#), [leader](#) und [threadpool](#) verwenden eine Voreinstellung von `MinSpareThreads 75` und behandeln unbeschäftigte Threads auf serverweiter Basis. Wenn nicht genügend unbeschäftigte Threads im Server vorhanden sind, dann werden solange Kindprozesse erzeugt, bis die Anzahl unbeschäftigter Threads größer als der angegebene Wert ist.

[mpm\\_netware](#) verwendet die Voreinstellung `MinSpareThreads 10` und verfolgt dies serverweit, da es ein Einzelprozess-MPM ist.

[beos](#) und [mpmt\\_os2](#) arbeiten ähnlich wie [mpm\\_netware](#). Die Voreinstellung für [beos](#) ist `MinSpareThreads 1`. Die Voreinstellung für [mpmt\\_os2](#) ist 5.

## Siehe auch

- [MaxSpareThreads](#)
- [StartServers](#)



<b>Beschreibung:</b>	Datei, in welcher der Server die Prozess-ID des Daemons ablegt
<b>Syntax:</b>	<code>PidFile <i>Dateiname</i></code>
<b>Voreinstellung:</b>	<code>PidFile logs/httpd.pid</code>
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Die Direktive `PidFile` bestimmt die Datei, in welcher der Server die Prozess-ID des Daemons ablegt. Wenn der Dateiname nicht absolut angegeben wird, wird er relativ zu `ServerRoot` interpretiert.

### Beispiel

```
PidFile /var/run/apache.pid
```

Es ist oft hilfreich, dem Server ein Signal senden zu können, damit er seine `ErrorLogs` und `TransferLogs` schließt und dann neu öffnet und seine Konfigurationsdateien neu einliest. Dies kann durch Senden eines SIGHUP-Signals (kill -1) an die Prozess-ID geschehen, die im `PidFile` eingetragen ist.

Die `PidFile`-Datei unterliegt den gleichen Warnungen über die Ablage von Protokolldateien und [Sicherheit](#).

### Anmerkung

Ab Apache 2 wird empfohlen, nur das Skript `apachectl` zum (Neu-)Starten und Stoppen des Servers zu verwenden.



<b><u>Beschreibung:</u></b>	Größe des TCP-Empfangspuffers
<b><u>Syntax:</u></b>	ReceiveBufferSize <i>Bytes</i>
<b><u>Voreinstellung:</u></b>	ReceiveBufferSize 0
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<u>beos</u> , <u>leader</u> , <u>mpm_netware</u> , <u>mpm_winnt</u> , <u>mpmt_os2</u> , <u>perchild</u> , <u>prefork</u> , <u>threadpool</u> , <u>worker</u>

Der Server setzt die Größe des TCP-Empfangspuffers auf die angegebene Anzahl Bytes.

Wird der Wert auf 0 gesetzt, dann verwendet der Server die Voreinstellung des Betriebssystems.



<b><u>Beschreibung:</u></b>	Ablageort der Datei, die zur Speicherung von Daten zur Koordinierung der Kindprozesse verwendet wird
<b><u>Syntax:</u></b>	ScoreBoardFile <i>Dateipfad</i>
<b><u>Voreinstellung:</u></b>	ScoreBoardFile logs/apache_status
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_winnt</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Apache verwendet ein Scoreboard zur Kommunikation zwischen seinen Eltern- und Kindprozessen. Einige Architekturen erfordern eine Datei zur Unterstützung der Kommunikation. Wenn die Datei undefiniert bleibt, versucht der Apache zuerst, das Scoreboard im Arbeitsspeicher zu erstellen (Verwendung von anonymem Shared-Memory), und versucht bei einem Fehlschlag anschließend die Datei auf der Festplatte zu erstellen (Verwendung von Datei-basiertem Shared-Memory). Die Angabe dieser Direktive veranlaßt den Apache stets, die Datei auf der Festplatte zu erstellen.

### Beispiel

```
ScoreBoardFile /var/run/apache_status
```

Datei-basiertes Shared-Memory ist für Applikationen von Drittanbietern hilfreich, die direkten Zugriff auf das Scoreboard benötigen.

Wenn Sie eine **ScoreBoardFile**-Anweisung verwenden, erreichen Sie eventuell eine höhere Geschwindigkeit, wenn Sie die Datei auf einer RAM-Disk ablegen. Achten Sie darauf, die gleichen Warnungen wie über die Ablage von Protokolldateien und

[Sicherheit](#) zu beherrsigen.

## Siehe auch

- [Apache beenden und neu starten](#)



<b>Beschreibung:</b>	Größe des TCP-Sendepuffers
<b>Syntax:</b>	SendBufferSize <i>Bytes</i>
<b>Voreinstellung:</b>	SendBufferSize 0
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#">beos</a> , <a href="#">leader</a> , <a href="#">mpm_netware</a> , <a href="#">mpm_winnt</a> , <a href="#">mpmt_os2</a> , <a href="#">perchild</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Der Server setzt die Größe des TCP-Sendepuffers auf die angegebene Anzahl Bytes. Dies ist sehr hilfreich, um Voreinstellungen alter Standardbetriebssysteme für Hochgeschwindigkeitsverbindungen mit hoher Latenzzeit anzuheben (*d.h.* 100ms oder so, wie bei Interkontinentalverbindungen).

Wird der Wert auf 0 gesetzt, dann verwendet der Server die Voreinstellung des Betriebssystems.



<b>Beschreibung:</b>	Obergrenze für die konfigurierbare Anzahl von Prozessen
<b>Syntax:</b>	<code>ServerLimit Anzahl</code>
<b>Voreinstellung:</b>	Für Details siehe Beschreibung
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<a href="#"><u>leader</u></a> , <a href="#"><u>perchild</u></a> , <a href="#"><u>prefork</u></a> , <a href="#"><u>threadpool</u></a> , <a href="#"><u>worker</u></a>

Bei dem MPM [prefork](#) bestimmt die Direktive den während der Lebensdauer des Apache-Prozesses maximal einstellbaren Wert für [MaxClients](#). Beim MPM [worker](#) bestimmt die Direktive in Verbindung mit [ThreadLimit](#) den Maximalwert für [MaxClients](#) für die Lebensdauer des Apache-Prozesses. Jeder Versuch, diese Anweisung während eines Neustarts zu ändern, wird ignoriert. [MaxClients](#) kann jedoch während eines Neustarts geändert werden.

Lassen Sie besondere Vorsicht bei der Verwendung dieser Direktive walten. Wenn [ServerLimit](#) auf einen Wert deutlich höher als notwendig gesetzt wird, wird zusätzliches, unbenutztes Shared-Memory belegt. Wenn sowohl [ServerLimit](#) als auch [MaxClients](#) auf Werte gesetzt werden, die größer sind, als das System sie handhaben kann, dann kann der Apache möglicherweise nicht starten, oder das System kann instabil werden.

Verwenden Sie die Direktive bei dem MPM [prefork](#) nur, wenn Sie [MaxClients](#) auf mehr als 256 (Voreinstellung) setzen müssen. Setzen Sie den Wert nicht höher als den Wert, den Sie für [MaxClients](#) angeben möchten.

Verwenden Sie die Direktive bei [worker](#), [leader](#) und [threadpool](#) nur, wenn Ihre [MaxClients](#)- und [ThreadsPerChild](#)-Einstellungen mehr als 16 Serverprozesse (Voreinstellung) erfordern. Setzen Sie den Wert dieser Direktive nicht höher, als die Anzahl der Serverprozesse, die dafür erforderlich ist, was Sie bei [MaxClients](#) und [ThreadsPerChild](#) angeben möchten.

Verwenden Sie die Direktive beim MPM [perchild](#) nur, wenn Sie [NumServers](#) auf einen Wert größer als 8 (Voreinstellung) setzen müssen.

### **Anmerkung**

Eine feste Begrenzung von `ServerLimit 20000` ist in den Server einkompiliert. Dies soll unangenehme Effekte durch Tippfehler verhindern.

### **Siehe auch**

- [Apache beenden und neu starten](#)



<b><u>Beschreibung:</u></b>	Anzahl der Kindprozesse des Servers, die beim Start erstellt werden
<b><u>Syntax:</u></b>	StartServers <i>Anzahl</i>
<b><u>Voreinstellung:</u></b>	Für Details siehe Beschreibung
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<a href="#">leader</a> , <a href="#">mpmt_os2</a> , <a href="#">prefork</a> , <a href="#">threadpool</a> , <a href="#">worker</a>

Die Direktive **StartServers** bestimmt die Anzahl der Kindprozesse des Servers, die beim Start erstellt werden. Da die Anzahl der Prozesse abhängig von der Last dynamisch kontrolliert wird, besteht normalerweise wenig Grund für eine Änderung dieses Parameters.

Die Voreinstellung unterscheidet sich von MPM zu MPM. Bei [leader](#), [threadpool](#) und [worker](#) ist die Voreinstellung StartServers 3. Die Voreinstellung bei [prefork](#) ist 5 und bei [mpmt\\_os2](#) 2.



<b><u>Beschreibung:</u></b>	Anzahl der Threads, die beim Start erstellt werden
<b><u>Syntax:</u></b>	StartThreads <i>Anzahl</i>
<b><u>Voreinstellung:</u></b>	Für Details siehe Beschreibung
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<a href="#">beos</a> , <a href="#">mpm_netware</a> , <a href="#">perchild</a>

Anzahl der Threads, die beim Start erstellt werden. Da die Anzahl der Threads abhängig von der Last dynamisch kontrolliert wird, besteht normalerweise wenig Grund für eine Änderung dieses Parameters.

Die Voreinstellung für [perchild](#) ist StartThreads 5. Die Direktive setzt während des Starts die Anzahl der Threads pro Prozess.

Die Voreinstellung bei [mpm\\_netware](#) ist StartThreads 50. Da hier lediglich ein einzelner Prozess existiert, ist dies die Gesamtzahl der Threads, die beim Start erstellt wird, um Anfragen zu bedienen.

Die Voreinstellung für [beos](#) ist StartThreads 10. Die Einstellung reflektiert ebenfalls die Gesamtzahl der Threads, die beim Start erstellt werden, um Anfragen zu bedienen.



<b>Beschreibung:</b>	Bestimmt die Obergrenze der konfigurierbaren Anzahl von Threads pro Kindprozess
<b>Syntax:</b>	ThreadLimit <i>Anzahl</i>
<b>Voreinstellung:</b>	Für Details siehe Beschreibung
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	<u>leader</u> , <u>mpm_winnt</u> , <u>perchild</u> , <u>threadpool</u> , <u>worker</u>
<b>Kompatibilität:</b>	Verfügbar für <u>mpm_winnt</u> ab Apache 2.0.41

Die Direktive bestimmt den während der Lebensdauer des Apache-Prozesses maximal einstellbaren Wert für ThreadsPerChild. Jeder Versuch, diese Direktive während eines Neustarts zu ändern, wird ignoriert. ThreadsPerChild kann jedoch während eines Neustarts modifiziert werden bis zu dem Wert dieser Anweisung.

Lassen Sie besondere Vorsicht bei der Verwendung dieser Direktive walten. Wenn ThreadLimit auf einen Wert deutlich höher als ThreadsPerChild gesetzt wird, wird zusätzliches, ungenutztes Shared-Memory belegt. Wenn sowohl ThreadLimit als auch ThreadsPerChild auf Werte gesetzt werden, die größer sind, als das System sie handhaben kann, dann kann der Apache möglicherweise nicht starten oder das System kann instabil werden. Setzen Sie den Wert dieser Direktive nicht höher als Ihre größte erwartete Einstellung für ThreadsPerChild während der aktuellen Ausführung des Apache.

Die Voreinstellung für ThreadLimit ist 1920 wenn sie zusammen mit mpm\_winnt verwendet wird, und 64 bei der Verwendung mit anderen MPMs.

### **Anmerkung**

Eine feste Begrenzung von ThreadLimit 20000 (oder ThreadLimit 15000 bei `mpm_winnt`) ist in den Server einkompiliert. Dies soll unangenehme Effekte durch Tippfehler verhindern.



<b><u>Beschreibung:</u></b>	Anzahl der Threads, die mit jedem Kindprozess gestartet werden
<b><u>Syntax:</u></b>	ThreadsPerChild <i>Anzahl</i>
<b><u>Voreinstellung:</u></b>	Für Details siehe Beschreibung
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<u>leader</u> , <u>mpm_winnt</u> , <u>threadpool</u> , <u>worker</u>

Die Direktive legt die Anzahl der Threads fest, die mit jedem Kindprozess gestartet werden. Der Kindprozess erstellt diese Threads beim Start und erstellt später keine weiteren mehr. Wenn Sie ein MPM wie mpm\_winnt verwenden, wo nur ein Kindprozess existiert, dann sollte diese Angabe hoch genug sein, die gesamte Last des Servers zu bewältigen. Wenn Sie ein MPM wie worker verwenden, wo mehrere Kindprozesse existieren, dann sollte die Gesamtzahl der Thread groß genug sein, die übliche Last auf dem Server zu bewältigen.

Die Voreinstellung für `ThreadsPerChild` ist 64, wenn mpm\_winnt verwendet wird, und 25 bei der Verwendung der anderen MPMs.



<b><u>Beschreibung:</u></b>	Die Benutzerkennung, unter welcher der Server Anfragen beantwortet
<b><u>Syntax:</u></b>	User <i>Unix-User-ID</i>
<b><u>Voreinstellung:</u></b>	User #-1
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<u>leader</u> , <u>perchild</u> , <u>prefork</u> , <u>threadpool</u> , <u>worker</u>
<b><u>Kompatibilität:</u></b>	Seit Apache 2.0 nur in der globalen Server-Konfiguration gültig

Die Direktive **User** legt die Benutzerkennung fest, mit der der Server Anfragen beantwortet. Um diese Anweisung zu verwenden, muss der Server als root gestartet werden. Wenn Sie den Server unter einem nicht-root-Benutzer starten, kann er nicht zu dem minder privilegierten Benutzer wechseln und wird statt dessen weiter mit der ursprünglichen Benutzerkennung laufen. Wenn Sie den Server als root starten, dann ist es normal, dass der Elternprozess als root weiterläuft. *Unix-User-ID* kann sein:

### **Ein Benutzername**

Verweist auf den durch Namen angegebenen Benutzer.

### **# gefolgt von einer Benutzernummer.**

Verweist auf einen durch eine Nummer angegebenen Benutzer.

Der Benutzer sollte keine Rechte besitzen, die dazu führen, dass er in der Lage ist, auf Dateien zuzugreifen, die nicht dafür bestimmt sind, für die Außenwelt sichtbar zu sein. Gleichmaßen sollte der Benutzer nicht in der Lage sein, Code auszuführen, der nicht für HTTP-Anfragen bestimmt ist. Es wird empfohlen, einen neuen Benutzer und eine neue Gruppe speziell zur Ausführung

des Servers zu erstellen. Einige Administratoren verwenden den Benutzer nobody. Dies ist jedoch nicht immer wünschenswert, da der Benutzer nobody andere Rechte auf dem System besitzen kann.

### Sicherheit

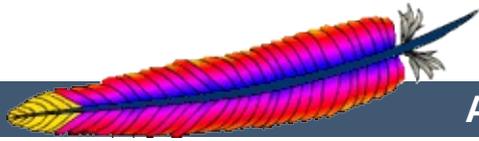
Setzen Sie `User` (oder `Group`) nicht auf `root`, solange Sie nicht genau wissen, was Sie tun, und welches die Gefahren sind.

Beim MPM `perchild`, das dafür gedacht ist, virtuelle Hosts unter verschiedenen Benutzerkennungen auszuführen, bestimmt die Direktive `User` die Benutzerkennung für den Hauptserver und bildet den Rückfallwert für `<VirtualHost>`-Abschnitte ohne eine `AssignUserID`-Anweisung.

Wichtiger Hinweis: Die Verwendung dieser Direktive innerhalb von `<VirtualHost>` wird nicht mehr unterstützt. Benutzen Sie `SuexecUserGroup`, um Ihren Server für `suexec` einzurichten.

### Anmerkung

Obwohl die Direktive `User` in den MPMs `beos` und `mpmt_os2` existiert, ist sie dort tatsächlich eine Leeraanweisung und existiert nur aus Kompatibilitätsgründen.



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

# Apache-MPM beos

<b><u>Beschreibung:</u></b>	Dieses Multi-Processing-Modul ist für BeOS optimiert.
<b><u>Status:</u></b>	MPM
<b><u>Modulbezeichner:</u></b>	mpm_beos_module
<b><u>Quelltext-Datei:</u></b>	beos.c

## Zusammenfassung

Dieses Multi-Processing-Modul (MPM) ist das Standardmodul für BeOS. Es benutzt einen einzelnen Steuerprozess welcher Threads für die Bedienung der Anfragen erzeugt.

## Siehe auch

[Adress- und Port-Einstellungen](#)



## MaxRequestsPerThread-Direktive

<b>Beschreibung:</b>	Die maximale Anzahl von Anfragen, die ein einzelner Thread während seiner Lebensdauer bedient.
<b>Syntax:</b>	MaxRequestsPerThread <i>Anzahl</i>
<b>Voreinstellung:</b>	MaxRequestsPerThread 0
<b>Kontext:</b>	Serverkonfiguration
<b>Status:</b>	MPM
<b>Modul:</b>	beos

Die Direktive `MaxRequestsPerThread` legt die Anzahl der Anfragen fest, die ein einzelner Server-Thread bedient. Nach Erreichen der angegebenen Anzahl von Anfragen wird der Thread beendet. Wird für `MaxRequestsPerThread` der Wert 0 angegeben, wird der Thread niemals beendet.

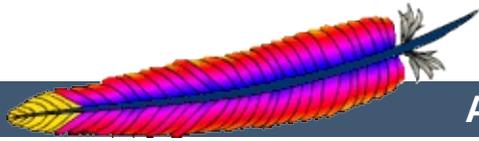
Das Setzen von `MaxRequestsPerThread` auf einen Wert ungleich null hat zwei Vorteile:

- Die Menge des von einem Thread benötigten Speicherplatzes bei (unvorhergesehenen) Speicherproblemen kann begrenzt werden;
- Threads mit begrenzter Lebensdauer reduzieren die Anzahl der Threads bei reduzierter Serverlast.

### Hinweis:

Bei `KeepAlive`-Anfragen wird nur die erste Anfrage auf das Maximum angerechnet. Das führt dazu, dass die Anzahl der *Verbindungen* pro Thread reduziert wird.

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

# Apache-MPM leader

<b>Beschreibung:</b>	Eine experimentelle Variante des Standard-MPMs <u>worker</u>
<b>Status:</b>	MPM
<b>Modulbezeichner:</b>	mpm_leader_module
<b>Quelltext-Datei:</b>	leader.c

## Zusammenfassung

### Warnung

Dieses MPM ist noch experimentell und funktioniert möglicherweise nicht wie erwartet.

Dies ist eine experimentelle Variante des Standard-MPMs worker. Das Modul verwendet ein Leader/Followers-Design-Pattern, um die Arbeit zwischen Threads zu koordinieren. Weitere Informationen finden Sie unter <http://deuce.doc.wustl.edu/doc/pspdfs/lf.pdf>.

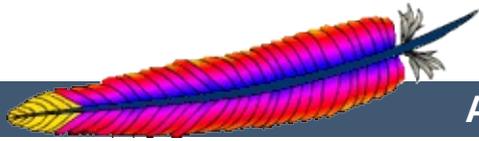
Um bei der Erstellung des [httpd](#) das MPM leader zu verwenden, fügen Sie den Argumenten des [configure](#)-Skripts `--with-mpm=leader` hinzu.

Dieses MPM baut auf den atomaren APR-Vergleichs- und -Tauschoperationen für die Thread-Synchronisation auf. Wenn Sie für einen x86-Rechner kompilieren, ohne dass 386-Unterstützung benötigt wird, oder wenn Sie für einen SPARC-Rechner kompilieren und keine pre-UltraSPARC-Chips betreiben müssen, fügen Sie den Argumenten des [configure](#)-Skripts `--enable-nonportable-atomics=yes` hinzu. Dies veranlasst die APR veranlasst dazu, atomare Operationen einzusetzen, welche effizienten Befehlscode verwenden, der älteren CPUs nicht zur Verfügung stehen.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache MPM netware

<b>Description:</b>	Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_netware_module
<b>Source File:</b>	mpm_netware.c

### Summary

This Multi-Processing Module (MPM) implements an exclusively threaded web server that has been optimized for Novell NetWare.

The main thread is responsible for launching child worker threads which listen for connections and serve them when they arrive. Apache always tries to maintain several *spare* or idle worker threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for a new child threads to be spawned before their requests can be served.

The [StartThreads](#), [MinSpareThreads](#), [MaxSpareThreads](#), and [MaxThreads](#) regulate how the main thread creates worker threads to serve requests. In general, Apache is very self-regulating, so most sites do not need to adjust these directives from their default values. Sites with limited memory may need to decrease [MaxThreads](#) to keep the server from thrashing (spawning and terminating idle threads). More information about tuning process creation is provided in the [performance hints](#) documentation.

[MaxRequestsPerChild](#) controls how frequently the server recycles processes by killing old ones and launching new ones. On the NetWare OS it is highly recommended that this directive remain set to 0. This allows worker threads to continue servicing requests

indefinitely.

## See also

[Setting which addresses and ports Apache uses](#)



## MaxThreads Directive

<b>Description:</b>	Set the maximum number of worker threads
<b>Syntax:</b>	MaxThreads <i>number</i>
<b>Default:</b>	MaxThreads 2048
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	mpm_netware

The **MaxThreads** directive sets the desired maximum number worker threads allowable. The default value is also the compiled in hard limit. Therefore it can only be lowered, for example:

```
MaxThreads 512
```



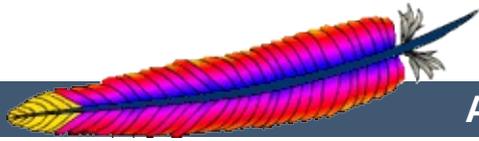
<b>Description:</b>	Determine the stack size for each thread
<b>Syntax:</b>	ThreadStackSize <i>number</i>
<b>Default:</b>	ThreadStackSize 65536
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	mpm_netware

This directive tells the server what stack size to use for each of the running threads. If you ever get a stack overflow you will need to bump this number to a higher setting.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache MPM os2

<b>Description:</b>	Hybrid multi-process, multi-threaded MPM for OS/2
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_mpmt_os2_module
<b>Source File:</b>	mpmt_os2.c

## Summary

The Server consists of a main, parent process and a small, static number of child processes.

The parent process's job is to manage the child processes. This involves spawning children as required to ensure there are always [StartServers](#) processes accepting connections.

Each child process consists of a a pool of worker threads and a main thread that accepts connections and passes them to the workers via a work queue. The worker thread pool is dynamic, managed by a maintenance thread so that the number of idle threads is kept between [MinSpareThreads](#) and [MaxSpareThreads](#).

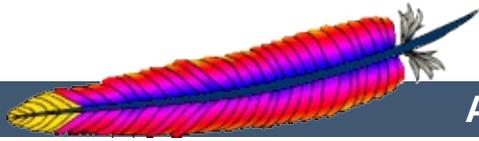
## See also

[Setting which addresses and ports Apache uses](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache MPM perchild

<b>Description:</b>	Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_perchild_module
<b>Source File:</b>	perchild.c

## Summary

This module is not functional. Development of this module is not complete and is not currently active. Do not use `perchild` unless you are a programmer willing to help fix it.

This Multi-Processing Module (MPM) implements a hybrid multi-process, multi-threaded web server. A fixed number of processes create threads to handle requests. Fluctuations in load are handled by increasing or decreasing the number of threads in each process.

## See also

[Setting which addresses and ports Apache uses](#)



A single control process launches the number of child processes indicated by the [NumServers](#) directive at server startup. Each child process creates threads as specified in the [StartThreads](#) directive. The individual threads then listen for connections and serve them when they arrive.

Apache always tries to maintain a pool of *spare* or idle server threads, which stand ready to serve incoming requests. In this way, clients do not need to wait for new threads to be created. For each child process, Apache assesses the number of idle threads and creates or destroys threads to keep this number within the boundaries specified by [MinSpareThreads](#) and [MaxSpareThreads](#). Since this process is very self-regulating, it is rarely necessary to modify these directives from their default values. The maximum number of clients that may be served simultaneously is determined by multiplying the number of server processes that will be created ([NumServers](#)) by the maximum number of threads created in each process ([MaxThreadsPerChild](#)).

While the parent process is usually started as root under Unix in order to bind to port 80, the child processes and threads are launched by Apache as a less-privileged user. The [User](#) and [Group](#) directives are used to set the privileges of the Apache child processes. The child processes must be able to read all the content that will be served, but should have as few privileges beyond that as possible. In addition, unless [suexec](#) is used, these directives also set the privileges which will be inherited by CGI scripts.

[MaxRequestsPerChild](#) controls how frequently the server recycles processes by killing old ones and launching new ones.

## Working with different user-IDs

The `perchild` MPM adds the extra ability to specify that particular processes should serve requests under different user-IDs. These user-IDs can then be associated with specific virtual hosts. You have to use one `ChildPerUserID` directive for every user/group combination you want to be run. Then you can tie particular virtual hosts to that user and group IDs.

The following example runs 7 child processes. Two of them are run under user1/group1. The next four are run under user2/group2 and the remaining process uses the `User` and `Group` of the main server:

### Global config

```
NumServers 7
ChildPerUserID user1 group1 2
ChildPerUserID user2 group2 4
```

Using unbalanced numbers of processes as above is useful, if the particular virtual hosts produce different load. The assignment to the virtual hosts is easily done as in the example below. In conclusion with the example above the following assumes, that server2 has to serve about twice of the hits of server1.

### Example

```
NameVirtualHost *

<VirtualHost *>
    ServerName fallbackhost
    # no assignment; use fallback
</VirtualHost>

<VirtualHost *>
    ServerName server1
    AssignUserID user1 group1
</VirtualHost>
```

```
<VirtualHost *>  
  ServerName server2  
  AssignUserID user2 group2  
</VirtualHost>
```



<b>Description:</b>	Tie a virtual host to a user and group ID
<b>Syntax:</b>	AssignUserID <i>user-id group-id</i>
<b>Context:</b>	virtual host
<b>Status:</b>	MPM
<b>Module:</b>	perchild

Tie a virtual host to a specific user/group combination. Requests addressed to the virtual host where this directive appears will be served by a process running with the specified user and group ID.

The user and group ID has to be assigned to a number of children in the global server config using the [ChildPerUserID](#) directive. See the section above for a [configuration example](#).



<b>Description:</b>	Specify user ID and group ID for a number of child processes
<b>Syntax:</b>	<code>ChildPerUserID <i>user-id group-id num-children</i></code>
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	perchild

Specify a user ID and group ID for a number of child processes. The third argument, *num-children*, is the number of child processes to start with the specified user and group. It does *not* represent a specific child number. In order to use this directive, the server must be run initially as root. If you start the server as a non-root user, it will fail to change to the lesser privileged user.

If the total number of child processes, found by totaling all of the third arguments to all `ChildPerUserID` directives in the config file, is less than `NumServers`, then all remaining children will inherit the `User` and `Group` settings from the main server. See the section above for a [configuration example](#).

### Security

Don't set *user-id* (or *group-id*) to root unless you know exactly what you are doing, and what the dangers are.



## MAXTHREADSPERCHILD DIRECTIVE

<b>Description:</b>	Maximum number of threads per child process
<b>Syntax:</b>	MaxThreadsPerChild <i>number</i>
<b>Default:</b>	MaxThreadsPerChild 64
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	perchild

This directive sets the maximum number of threads that will be created in each child process. To increase this value beyond its default, it is necessary to change the value of the [ThreadLimit](#) directive and stop and re-start the server.



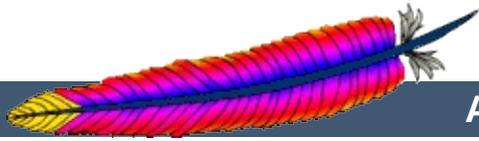
<b>Description:</b>	Total number of children alive at the same time
<b>Syntax:</b>	NumServers <i>number</i>
<b>Default:</b>	NumServers 2
<b>Context:</b>	server config
<b>Status:</b>	MPM
<b>Module:</b>	perchild

The **NumServers** directive determines the number of children alive at the same time. This number should be large enough to handle the requests for the entire site. To increase this value beyond the value of 8, it is necessary to change the value of the **ServerLimit** directive and stop and re-start the server. See the section above for a [configuration example](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

# Apache-MPM prefork

<b>Beschreibung:</b>	Implementiert einen im Voraus forkenden Webserver ohne Thread-Unterstützung
<b>Status:</b>	MPM
<b>Modulbezeichner:</b>	mpm_prefork_module
<b>Quelltext-Datei:</b>	prefork.c

## Zusammenfassung

Dieses Multi-Processing-Modul (MPM) implementiert einen im Voraus forkenden Webserver ohne Thread-Unterstützung, der Anfragen auf ähnliche Weise behandelt wie der Apache 1.3. Es ist für Angebote geeignet, die aus Kompatibilitätsgründen mit nicht-Thread-sicheren Bibliotheken Threading vermeiden müssen. Es ist außerdem das geeignetste MPM, um jede Anfrage isoliert zu bearbeiten, so dass Probleme mit einem einzelnen Prozess keinen anderen beeinflussen.

Das MPM ist stark selbstregulierend, so dass es selten notwendig ist, seine Konfigurationseinstellungen zu justieren. Das Wichtigste ist, dass **MaxClients** gross genug ist, so viele gleichzeitige Anfragen zu bedienen, wie Sie erwarten, aber klein genug, um sicherzustellen, dass genug physischer Arbeitsspeicher für alle Prozesse vorhanden ist.

## Siehe auch

[Bestimmen der vom Apache verwendeten Adressen und Ports](#)



Ein einzelner Steuerprozess ist für den Start von Kindprozessen verantwortlich, die auf Verbindungen warten und diese bedienen, sobald sie eintreffen. Der Apache versucht immer, mehrere *freie* oder unbeschäftigte Serverprozesse vorzuhalten, die zur Bedienung eingehender Anfragen bereit stehen. Auf diese Weise müssen Clients nicht darauf warten, dass neue Kindprozesse geforkt werden, bevor ihre Anfrage bearbeitet werden kann.

[StartServers](#), [MinSpareServers](#), [MaxSpareServers](#) und [MaxClients](#) regulieren, wie der Elternprozess Kindprozesse zur Bedienung von Anfragen erstellt. Im Allgemeinen ist der Apache sehr selbstregulierend, so dass die meisten Angebote die Voreinstellung dieser Direktiven nicht verändern müssen. Systeme, die mehr als 256 gleichzeitige Anfragen bedienen müssen, können [MaxClients](#) erhöhen, während Systeme mit begrenztem Arbeitsspeicher möglicherweise [MaxClients](#) heruntersetzen müssen, um den Server vor Flatterverhalten (Arbeitsspeicherinhalte auf Platte auslagern - und zurück) zu schützen. Weitere Informationen zur Feinabstimmung der Prozesserstellung sind in den [Performance-Hinweisen](#) zu finden.

Während der Elternprozess unter Unix normalerweise als `root` gestartet wird, um sich an Port 80 binden zu können, werden die Kindprozesse unter einem weniger privilegierten Benutzer gestartet. Die Direktiven [User](#) und [Group](#) werden dazu verwendet, die Privilegien der Apache-Kindprozesse festzulegen. Die Kindprozesse müssen in der Lage sein, alle Inhalte zu lesen, die sie ausliefern sollen, sollten darüber hinaus jedoch so wenig wie möglich Rechte besitzen.

[MaxRequestsPerChild](#) bestimmt, wie häufig der Server Prozesse erneuert, indem er alte beendet und neue startet.



<b><u>Beschreibung:</u></b>	Maximale Anzahl der unbeschäftigten Kindprozesse des Servers
<b><u>Syntax:</u></b>	MaxSpareServers <i>Anzahl</i>
<b><u>Voreinstellung:</u></b>	MaxSpareServers 10
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	prefork

Die Direktive [MaxSpareServers](#) bestimmt das gewünschte Maximum an *unbeschäftigten* Kindprozessen des Servers. Ein unbeschäftigter Prozess ist einer, der keine Anfrage bedient. Wenn mehr als [MaxSpareServers](#) Prozesse unbeschäftigt sind, wird der Elternprozess die überschüssigen Prozesse beenden.

Eine Feineinstellung dieses Parameters sollte nur bei sehr beschäftigten Angeboten notwendig sein. Es ist nahezu immer eine schlechte Idee, den Parameter auf einen hohen Wert zu setzen. Wenn Sie versuchen, den Wert niedriger als [MinSpareServers](#) zu setzen, wird der Apache ihn automatisch auf [MinSpareServers](#) + 1 korrigieren.

## Siehe auch

- [MinSpareServers](#)
- [StartServers](#)



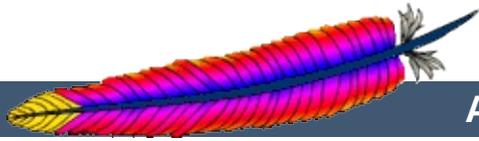
<b><u>Beschreibung:</u></b>	Minimale Anzahl der unbeschäftigten Kindprozesse des Servers
<b><u>Syntax:</u></b>	MinSpareServers <i>Anzahl</i>
<b><u>Voreinstellung:</u></b>	MinSpareServers 5
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	prefork

Die Direktive **MinSpareServers** bestimmt das gewünschte Minimum der *unbeschäftigten* Kindprozesse des Servers. Ein unbeschäftigter Prozess ist einer, der keine Anfrage bedient. Wenn weniger als **MinSpareServers** Prozesse unbeschäftigt sind, dann erstellt der Elternprozess neue mit einer maximalen Rate von 1 pro Sekunde.

Die Feineinstellung des Parameters sollte nur bei sehr beschäftigten Angeboten notwendig sein. Es ist nahezu immer eine schlechte Idee, den Parameter auf einen hohen Wert zu setzen.

## Siehe auch

- [MaxSpareServers](#)
- [StartServers](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache MPM threadpool

<b>Description:</b>	Yet another experimental variant of the standard <a href="#">worker</a> MPM
<b>Status:</b>	MPM
<b>Module Identifier:</b>	mpm_threadpool_module
<b>Source File:</b>	threadpool.c

## Summary

### Warning

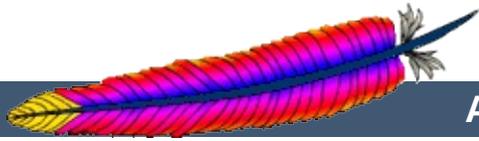
This MPM is a developer playground and highly experimental, so it may or may not work as expected.

This is an experimental variant of the standard [worker](#) MPM. Rather than queuing connections like the [worker](#) MPM, the [threadpool](#) MPM queues idle worker threads and hands each accepted connection to the next available worker.

The [threadpool](#) MPM can't match the performance of the [worker](#) MPM in benchmark testing. As of 2.0.39, some of the key load-throttling concepts from the [threadpool](#) MPM have been incorporated into the [worker](#) MPM. The [threadpool](#) code is useful primarily as a research platform. For general-purpose use and for any production environments, use [worker](#) instead.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

## Apache-MPM winnt

<b>Beschreibung:</b>	Das Multi-Processing-Modul ist optimiert für Windows NT.
<b>Status:</b>	MPM
<b>Modulbezeichner:</b>	mpm_winnt_module
<b>Quelltext-Datei:</b>	mpm_winnt.c

### Zusammenfassung

Dieses Multi-Processing-Modul (MPM) ist die Voreinstellung für das Betriebssystem Windows NT. Es verwendet einen einzelnen Steuerprozess, der einen einzelnen Kindprozess startet, welcher wiederum Threads zur Bedienung von Anfragen erstellt.

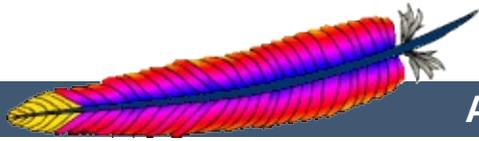


<b><u>Beschreibung:</u></b>	Für die Annahme von Netzwerkverbindungen wird <code>accept()</code> anstelle von <code>AcceptEx()</code> verwendet
<b><u>Syntax:</u></b>	<code>Win32DisableAcceptEx</code>
<b><u>Voreinstellung:</u></b>	<code>AcceptEx()</code> ist standardmäßig aktiviert. Verwenden Sie diese Direktive, um den Gebrauch von <code>AcceptEx()</code> zu deaktivieren.
<b><u>Kontext:</u></b>	Serverkonfiguration
<b><u>Status:</u></b>	MPM
<b><u>Modul:</u></b>	<code>mpm_winnt</code>
<b><u>Kompatibilität:</u></b>	Verfügbar ab Version 2.0.49

`AcceptEx()` ist eine Schnittstelle zu Microsoft Winsock v2, die unter bestimmten Umständen einige Leistungsverbesserungen gegenüber der `accept()`-API von BSD bietet. Einige beliebte Windows-Produkte, typischerweise Virens Scanner oder VPN-Pakete, besitzen jedoch Fehler, welche den einwandfreien Betrieb von `AcceptEx()` stören. Wenn Sie einen Fehler wie:

```
[error] (730038)An operation was attempted on something that is not a socket.: winnt_accept: AcceptEx failed. Attempting to recover.
```

erhalten, sollten Sie diese Direktive verwenden, um den Gebrauch von `AcceptEx()` zu unterbinden.



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

## Apache-MPM worker

<b><u>Beschreibung:</u></b>	Multi-Processing-Modul, das einen Hybrid-Webserver mit Multi-Thread und Multi-Prozess-Unterstützung implementiert
<b><u>Status:</u></b>	MPM
<b><u>Modulbezeichner:</u></b>	mpm_worker_module
<b><u>Quelltext-Datei:</u></b>	worker.c

### Zusammenfassung

Dieses Multi-Processing-Modul (MPM) implementiert einen Hybrid-Server mit Multi-Thread und Multi-Prozess-Unterstützung. Durch die Verwendung von Threads für die Bedienung von Anfragen ist er in der Lage, eine große Anzahl von Anfragen mit weniger Systemressourcen als ein Prozess-basierter Server zu bedienen. Er behält jedoch viel von der Stabilität eines Prozess-basierten Servers bei, indem er mehrere Prozesse verfügbar hält, jeden mit etlichen Threads.

Die wichtigsten Direktiven zur Steuerung des MPMs sind [ThreadsPerChild](#), welche die Anzahl der Threads beeinflusst, die von jedem Kindprozess verwendet werden, und [MaxClients](#), welche die maximale Gesamtzahl an Threads regelt, die gestartet werden können.

### Siehe auch

[Bestimmen der vom Apache verwendeten Adressen und Ports](#)



Ein einzelner Steuerprozess (der Elternprozess) ist für den Start der Kindprozesse verantwortlich. Jeder Kindprozess erstellt eine feste Anzahl von Server-Threads, wie durch die [ThreadsPerChild](#)-Direktive angegeben, sowie einen "Listener-Thread", der auf Verbindungen wartet und diese an einen Server-Thread zur Bearbeitung weiterreicht, sobald sie eintreffen.

Der Apache versucht immer, einen Vorrat von *freien* oder unbeschäftigten Threads zu verwalten, die zur Bedienung hereinkommender Anfragen bereit stehen. Auf diese Weise brauchen Clients nicht auf die Erstellung eines neuen Threads oder Prozesses zu warten, bevor ihre Anfrage bedient werden kann. Die Anzahl der Prozesse, die anfangs gestartet wird, wird mit der Direktive [StartServers](#) festgelegt. Dann, während des Betriebes, berechnet der Apache die Gesamtzahl der unbeschäftigten Threads und forkt oder beendet Prozesse, um diese Anzahl innerhalb der durch [MinSpareThreads](#) und [MaxSpareThreads](#) angegebenen Grenzen zu halten. Da dieser Prozess sehr selbstregulierend ist, ist es nur selten notwendig, die Voreinstellung dieser Direktiven zu ändern. Die maximale Anzahl Clients, die gleichzeitig bedient werden kann (d.h. die maximale Gesamtzahl der Threads in allen Prozessen), wird mit der Direktive [MaxClients](#) festgelegt. Die maximale Anzahl der aktiven Kindprozesse ergibt sich aus [MaxClients](#) dividiert durch [ThreadsPerChild](#).

Zwei Direktiven legen harte Limits für die Anzahl der aktiven Kindprozesse fest und können nur geändert werden, indem der Server komplett gestoppt und dann wieder neu gestartet wird. [ServerLimit](#) stellt die obere Grenze für die Anzahl der aktiven Kindprozesse dar und muss größer oder gleich dem Quotienten aus [MaxClients](#) und [ThreadsPerChild](#) sein. [ThreadLimit](#) ist die obere Grenze für die Anzahl der Server-Threads und muss

größer oder gleich [ThreadsPerChild](#) sein. Sofern für diese Direktiven keine Voreinstellungen verwendet werden, sollten sie vor allen anderen [worker](#)-Direktiven platziert werden.

Neben den normalen aktiven Kindprozessen gibt es möglicherweise noch zusätzliche Kindprozesse, welche gerade beendet werden, wo allerdings zumindest noch ein Server-Thread eine existierende Verbindung bearbeitet. Obwohl die tatsächlich zu erwartende Anzahl deutlich kleiner ist, können bis zu [MaxClients](#) solcher Prozesse auftreten. Dieses Verhalten können Sie vermeiden, indem Sie die Terminierung einzelner Kindprozesse wie folgt abschalten:

- setzen Sie den Wert von [MaxRequestsPerChild](#) auf Null
- setzen Sie den Wert von [MaxSpareThreads](#) auf den gleichen Wert wie [MaxClients](#)

Eine typische Konfiguration der Prozess-Thread-Steuerung für das MPM [worker](#) könnte wie folgt aussehen:

```
ServerLimit 16
StartServers 2
MaxClients 150
MinSpareThreads 25
MaxSpareThreads 75
ThreadsPerChild 25
```

Während der Elternprozess unter Unix normalerweise als `root` gestartet wird, um sich an Port 80 binden zu können, werden die Kindprozesse und Threads unter einem weniger privilegierten Benutzer gestartet. Die Direktiven [User](#) und [Group](#) werden dazu verwendet, die Privilegien der Apache-Kindprozesse festzulegen. Die Kindprozesse müssen in der Lage sein, alle Inhalte zu lesen, die sie ausliefern sollen, sollten darüber hinaus jedoch so wenig wie möglich Rechte besitzen. Zusätzlich, solange nicht [suexec](#) verwendet wird, legen diese Direktiven auch die Privilegien fest,

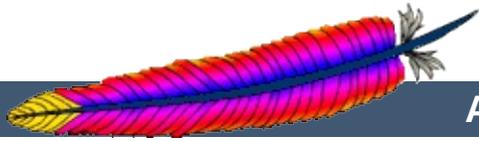
die von CGI-Skripts geerbt werden.

[MaxRequestsPerChild](#) bestimmt, wie häufig der Server Prozesse erneuert, indem er alte beendet und neue startet.

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module `mod_access`

<b>Description:</b>	Provides access control based on client hostname, IP address, or other characteristics of the client request.
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>access_module</code>
<b>Source File:</b>	<code>mod_access.c</code>
<b>Compatibility:</b>	Available only in versions prior to 2.1

### Summary

The directives provided by `mod_access` are used in `<Directory>`, `<Files>`, and `<Location>` sections as well as `.htaccess` files to control access to particular parts of the server. Access can be controlled based on the client hostname, IP address, or other characteristics of the client request, as captured in `environment variables`. The `Allow` and `Deny` directives are used to specify which clients are or are not allowed access to the server, while the `Order` directive sets the default access state, and configures how the `Allow` and `Deny` directives interact with each other.

Both host-based access restrictions and password-based authentication may be implemented simultaneously. In that case, the `Satisfy` directive is used to determine how the two sets of restrictions interact.

In general, access restriction directives apply to all access methods (GET, PUT, POST, etc). This is the desired behavior in most cases. However, it is possible to restrict some methods, while leaving other methods unrestricted, by enclosing the directives in a `<Limit>` section.

## See also

[Satisfy](#)

[Require](#)



<b>Description:</b>	Controls which hosts can access an area of the server
<b>Syntax:</b>	Allow from all  <i>host</i>  env= <i>env-variable</i> [ <i>host</i>  env= <i>env-variable</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	Limit
<b>Status:</b>	Base
<b>Module:</b>	mod_access

The **Allow** directive affects which hosts can access an area of the server. Access can be controlled by hostname, IP address, IP address range, or by other characteristics of the client request captured in environment variables.

The first argument to this directive is always `from`. The subsequent arguments can take three different forms. If `Allow from all` is specified, then all hosts are allowed access, subject to the configuration of the **Deny** and **Order** directives as discussed below. To allow only particular hosts or groups of hosts to access the server, the *host* can be specified in any of the following formats:

### A (partial) domain-name

#### Example:

```
Allow from apache.org
Allow from .net example.edu
```

Hosts whose names match, or end in, this string are allowed access. Only complete components are matched, so the above example will match `foo.apache.org` but it will not match `fooapache.org`. This configuration will cause

Apache to perform a double reverse DNS lookup on the client IP address, regardless of the setting of the [HostnameLookups](#) directive. It will do a reverse DNS lookup on the IP address to find the associated hostname, and then do a forward lookup on the hostname to assure that it matches the original IP address. Only if the forward and reverse DNS are consistent and the hostname matches will access be allowed.

## A full IP address

### Example:

```
Allow from 10.1.2.3  
Allow from 192.168.1.104 192.168.1.205
```

An IP address of a host allowed access

## A partial IP address

### Example:

```
Allow from 10.1  
Allow from 10 172.20 192.168.2
```

The first 1 to 3 bytes of an IP address, for subnet restriction.

## A network/netmask pair

### Example:

```
Allow from 10.1.0.0/255.255.0.0
```

A network a.b.c.d, and a netmask w.x.y.z. For more fine-grained subnet restriction.

## A network/nnn CIDR specification

---

### Example:

```
Allow from 10.1.0.0/16
```

Similar to the previous case, except the netmask consists of *nnn* high-order 1 bits.

Note that the last three examples above match exactly the same set of hosts.

IPv6 addresses and IPv6 subnets can be specified as shown below:

```
Allow from 2001:db8::a00:20ff:fea7:ccea  
Allow from 2001:db8::a00:20ff:fea7:ccea/10
```

The third format of the arguments to the **Allow** directive allows access to the server to be controlled based on the existence of an [environment variable](#). When `Allow from env=env-variable` is specified, then the request is allowed access if the environment variable *env-variable* exists. The server provides the ability to set environment variables in a flexible way based on characteristics of the client request using the directives provided by [mod\\_setenvif](#). Therefore, this directive can be used to allow access based on such factors as the clients User-Agent (browser type), Referer, or other HTTP request header fields.

### Example:

```
SetEnvIf User-Agent ^KnockKnock/2\.0 let_me_in  
<Directory /docroot>  
    Order Deny,Allow  
    Deny from all  
    Allow from env=let_me_in  
</Directory>
```

In this case, browsers with a user-agent string beginning with

KnockKnock/2.0 will be allowed access, and all others will be denied.



## Deny Directive

**Description:** Controls which hosts are denied access to the server

**Syntax:** Deny from *all|host|env=env-variable*  
*[host|env=env-variable] ...*

**Context:** directory, .htaccess

**Override:** Limit

**Status:** Base

**Module:** mod\_access

This directive allows access to the server to be restricted based on hostname, IP address, or environment variables. The arguments for the **Deny** directive are identical to the arguments for the **Allow** directive.



**Description:** Controls the default access state and the order in which **Allow** and **Deny** are evaluated.

**Syntax:** Order *ordering*

**Default:** Order Deny, Allow

**Context:** directory, .htaccess

**Override:** Limit

**Status:** Base

**Module:** mod\_access

The **Order** directive, along with the **Allow** and **Deny** directives, controls a three-pass access control system. The first pass processes either all **Allow** or all **Deny** directives, as specified by the **Order** directive. The second pass parses the rest of the directives (**Deny** or **Allow**). The third pass applies to all requests which do not match either of the first two.

Note that all **Allow** and **Deny** directives are processed, unlike a typical firewall, where only the first match is used. The last match is effective (also unlike a typical firewall). Additionally, the order in which lines appear in the configuration files is not significant -- all **Allow** lines are processed as one group, all **Deny** lines are considered as another, and the default state is considered by itself.

*Ordering* is one of:

### **Allow, Deny**

First, all **Allow** directives are evaluated; at least one must match, or the request is rejected. Next, all **Deny** directives are evaluated. If any matches, the request is rejected. Last, any requests which do not match an **Allow** or a **Deny** directive are denied by default.

## Deny, Allow

First, all [Deny](#) directives are evaluated; if any match, the request is denied **unless** it also matches an [Allow](#) directive. Any requests which do not match any [Allow](#) or [Deny](#) directives are permitted.

## Mutual-failure

This order has the same effect as `Order Allow, Deny` and is deprecated in its favor.

Keywords may only be separated by a comma; *no whitespace* is allowed between them.

Match	Allow,Deny result	Deny,Allow result
Match Allow only	Request allowed	Request allowed
Match Deny only	Request denied	Request denied
No match	Default to second directive: Denied	Default to second directive: Allowed
Match both Allow & Deny	Final match controls: Denied	Final match controls: Allowed

In the following example, all hosts in the `apache.org` domain are allowed access; all other hosts are denied access.

```
Order Deny,Allow
Deny from all
Allow from apache.org
```

In the next example, all hosts in the `apache.org` domain are allowed access, except for the hosts which are in the `foo.apache.org` subdomain, who are denied access. All hosts not in the `apache.org` domain are denied access because the default state is to [Deny](#) access to the server.

```
Order Allow,Deny
Allow from apache.org
Deny from foo.apache.org
```

On the other hand, if the **Order** in the last example is changed to **Deny, Allow**, all hosts will be allowed access. This happens because, regardless of the actual ordering of the directives in the configuration file, the **Allow from apache.org** will be evaluated last and will override the **Deny from foo.apache.org**. All hosts not in the **apache.org** domain will also be allowed access because the default state is **Allow**.

The presence of an **Order** directive can affect access to a part of the server even in the absence of accompanying **Allow** and **Deny** directives because of its effect on the default access state. For example,

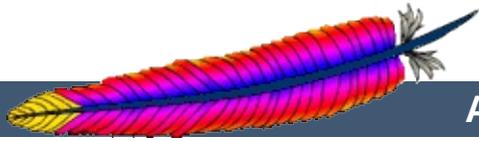
```
<Directory /www>
  Order Allow,Deny
</Directory>
```

will **Deny** all access to the **/www** directory because the default access state is set to **Deny**.

The **Order** directive controls the order of access directive processing only within each phase of the server's configuration processing. This implies, for example, that an **Allow** or **Deny** directive occurring in a **<Location>** section will always be evaluated after an **Allow** or **Deny** directive occurring in a **<Directory>** section or **.htaccess** file, regardless of the setting of the **Order** directive. For details on the merging of configuration sections, see the documentation on [How Directory, Location and Files sections work](#).

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_actions

<b>Description:</b>	This module provides for executing CGI scripts based on media type or request method.
<b>Status:</b>	Base
<b>Module Identifier:</b>	actions_module
<b>Source File:</b>	mod_actions.c

## Summary

This module has two directives. The [Action](#) directive lets you run CGI scripts whenever a file of a certain type is requested. The [Script](#) directive lets you run CGI scripts whenever a particular method is used in a request. This makes it much easier to execute scripts that process files.

## See also

[mod\\_cgi](#)

[Dynamic Content with CGI](#)

[Apache's Handler Use](#)



<b>Description:</b>	Activates a CGI script for a particular handler or content-type
<b>Syntax:</b>	Action <i>action-type cgi-script</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_actions

This directive adds an action, which will activate *cgi-script* when *action-type* is triggered by the request. The *cgi-script* is the URL-path to a resource that has been designated as a CGI script using [ScriptAlias](#) or [AddHandler](#). The *action-type* can be either a [handler](#) or a MIME content type. It sends the URL and file path of the requested document using the standard CGI PATH\_INFO and PATH\_TRANSLATED environment variables.

### Examples

```
# Requests for files of a particular type:
Action image/gif /cgi-bin/images.cgi

# Files of a particular file extension
AddHandler my-file-type .xyz
Action my-file-type /cgi-bin/program.cgi
```

In the first example, requests for files with a MIME content type of `image/gif` will instead be handled by the specified cgi script `/cgi-bin/images.cgi`.

In the second example, requests for files with a file extension of `.xyz` are handled instead by the specified cgi script `/cgi-bin/program.cgi`.

### See also

- [AddHandler](#)



**Description:** Activates a CGI script for a particular request method.

**Syntax:** `Script method cgi-script`

**Context:** server config, virtual host, directory

**Status:** Base

**Module:** mod\_actions

This directive adds an action, which will activate *cgi-script* when a file is requested using the method of *method*. The *cgi-script* is the URL-path to a resource that has been designated as a CGI script using [ScriptAlias](#) or [AddHandler](#). The URL and file path of the requested document is sent using the standard CGI `PATH_INFO` and `PATH_TRANSLATED` environment variables.

Any arbitrary method name may be used. **Method names are case-sensitive**, so `Script PUT` and `Script put` have two entirely different effects.

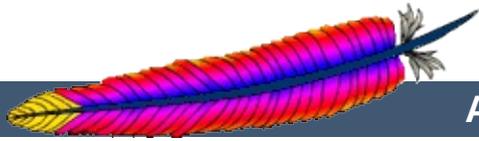
Note that the `Script` command defines default actions only. If a CGI script is called, or some other resource that is capable of handling the requested method internally, it will do so. Also note that `Script` with a method of `GET` will only be called if there are query arguments present (e.g., `foo.html?hi`). Otherwise, the request will proceed normally.

### Examples

```
# For <ISINDEX>-style searching
Script GET /cgi-bin/search
```

```
# A CGI PUT handler
Script PUT /~bob/put.cgi
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_alias`

<b>Description:</b>	Provides for mapping different parts of the host filesystem in the document tree and for URL redirection
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>alias_module</code>
<b>Source File:</b>	<code>mod_alias.c</code>

## Summary

The directives contained in this module allow for manipulation and control of URLs as requests arrive at the server. The [Alias](#) and [ScriptAlias](#) directives are used to map between URLs and filesystem paths. This allows for content which is not directly under the [DocumentRoot](#) served as part of the web document tree. The [ScriptAlias](#) directive has the additional effect of marking the target directory as containing only CGI scripts.

The [Redirect](#) directives are used to instruct clients to make a new request with a different URL. They are often used when a resource has moved to a new location.

[mod\\_alias](#) is designed to handle simple URL manipulation tasks. For more complicated tasks such as manipulating the query string, use the tools provided by [mod\\_rewrite](#).

## See also

[mod\\_rewrite](#)

[Mapping URLs to the filesystem](#)



Aliases and Redirects occurring in different contexts are processed like other directives according to standard [merging rules](#). But when multiple Aliases or Redirects occur in the same context (for example, in the same `<VirtualHost>` section) they are processed in a particular order.

First, all Redirects are processed before Aliases are processed, and therefore a request that matches a [Redirect](#) or [RedirectMatch](#) will never have Aliases applied. Second, the Aliases and Redirects are processed in the order they appear in the configuration files, with the first match taking precedence.

For this reason, when two or more of these directives apply to the same sub-path, you must list the most specific path first in order for all the directives to have an effect. For example, the following configuration will work as expected:

```
Alias /foo/bar /baz
Alias /foo /gaq
```

But if the above two directives were reversed in order, the `/foo` [Alias](#) would always match before the `/foo/bar` [Alias](#), so the latter directive would be ignored.



<b>Description:</b>	Maps URLs to filesystem locations
<b>Syntax:</b>	<code>Alias <i>URL-path</i> <i>file-path</i> <i>directory-path</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

The **Alias** directive allows documents to be stored in the local filesystem other than under the **DocumentRoot**. URLs with a (%-decoded) path beginning with *url-path* will be mapped to local files beginning with *directory-path*. The *url-path* is case-sensitive, even on case-insensitive file systems.

#### Example:

```
Alias /image /ftp/pub/image
```

A request for `http://myserver/image/foo.gif` would cause the server to return the file `/ftp/pub/image/foo.gif`.

Note that if you include a trailing `/` on the *url-path* then the server will require a trailing `/` in order to expand the alias. That is, if you use

```
Alias /icons/ /usr/local/apache/icons/
```

then the url `/icons` will not be aliased.

Note that you may need to specify additional **<Directory>** sections which cover the *destination* of aliases. Aliasing occurs before **<Directory>** sections are checked, so only the destination of aliases are affected. (Note however **<Location>** sections are run through once before aliases are performed, so they will apply.)

In particular, if you are creating an Alias to a directory outside of your [DocumentRoot](#), you may need to explicitly permit access to the target directory.

**Example:**

```
Alias /image /ftp/pub/image
<Directory /ftp/pub/image>
    Order allow,deny
    Allow from all
</Directory>
```



<b>Description:</b>	Maps URLs to filesystem locations using regular expressions
<b>Syntax:</b>	<code>AliasMatch <i>regex file-path directory-path</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

This directive is equivalent to [Alias](#), but makes use of standard regular expressions, instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to activate the `/icons` directory, one might use:

```
AliasMatch ^/icons(.*) /usr/local/apache/icons$1
```

It is also possible to construct an alias with case-insensitive matching of the url-path:

```
AliasMatch (?i)^/image(.*) /ftp/pub/image$1
```



<b>Description:</b>	Sends an external redirect asking the client to fetch a different URL
<b>Syntax:</b>	Redirect [ <i>status</i> ] <i>URL-path</i> <i>URL</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

The Redirect directive maps an old URL into a new one by asking the client to refetch the resource at the new location.

The old *URL-path* is a case-sensitive (%-decoded) path beginning with a slash. A relative path is not allowed. The new *URL* should be an absolute URL beginning with a scheme and hostname.

#### Example:

```
Redirect /service http://foo2.bar.com/service
```

If the client requests `http://myserver/service/foo.txt`, it will be told to access `http://foo2.bar.com/service/foo.txt` instead.

#### Note

Redirect directives take precedence over Alias and ScriptAlias directives, irrespective of their ordering in the configuration file. Also, *URL-path* must be a fully qualified URL, not a relative path, even when used with .htaccess files or inside of [<Directory>](#) sections.

If no *status* argument is given, the redirect will be "temporary" (HTTP status 302). This indicates to the client that the resource has moved temporarily. The *status* argument can be used to return

other HTTP status codes:

**permanent**

Returns a permanent redirect status (301) indicating that the resource has moved permanently.

**temp**

Returns a temporary redirect status (302). This is the default.

**seeother**

Returns a "See Other" status (303) indicating that the resource has been replaced.

**gone**

Returns a "Gone" status (410) indicating that the resource has been permanently removed. When this status is used the *URL* argument should be omitted.

Other status codes can be returned by giving the numeric status code as the value of *status*. If the status is between 300 and 399, the *URL* argument must be present, otherwise it must be omitted. Note that the status must be known to the Apache code (see the function `send_error_response` in `http_protocol.c`).

**Example:**

```
Redirect permanent /one http://example.com/two  
Redirect 303 /three http://example.com/other
```



<b>Description:</b>	Sends an external redirect based on a regular expression match of the current URL
<b>Syntax:</b>	<code>RedirectMatch [status] regex URL</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

This directive is equivalent to [Redirect](#), but makes use of standard regular expressions, instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to redirect all GIF files to like-named JPEG files on another server, one might use:

```
RedirectMatch (.*)\.gif$ http://www.anotherserver.com$1.jpg
```



<b>Description:</b>	Sends an external permanent redirect asking the client to fetch a different URL
<b>Syntax:</b>	RedirectPermanent <i>URL-path URL</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

This directive makes the client know that the Redirect is permanent (status 301). Exactly equivalent to Redirect permanent.



## RedirectTemp Directive

<b>Description:</b>	Sends an external temporary redirect asking the client to fetch a different URL
<b>Syntax:</b>	RedirectTemp <i>URL-path</i> URL
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

This directive makes the client know that the Redirect is only temporary (status 302). Exactly equivalent to Redirect temp.



<b>Description:</b>	Maps a URL to a filesystem location and designates the target as a CGI script
<b>Syntax:</b>	<code>ScriptAlias URL-path file-path directory-path</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

The `ScriptAlias` directive has the same behavior as the `Alias` directive, except that in addition it marks the target directory as containing CGI scripts that will be processed by `mod_cgi`'s cgi-script handler. URLs with a case-sensitive (%-decoded) path beginning with `URL-path` will be mapped to scripts beginning with the second argument, which is a full pathname in the local filesystem.

#### Example:

```
ScriptAlias /cgi-bin/ /web/cgi-bin/
```

A request for `http://myserver/cgi-bin/foo` would cause the server to run the script `/web/cgi-bin/foo`. This configuration is essentially equivalent to:

```
Alias /cgi-bin/ /web/cgi-bin/  
<Location /cgi-bin >  
    SetHandler cgi-script  
    Options +ExecCGI  
</Location>
```

It is safer to avoid placing CGI scripts under the `DocumentRoot` in order to avoid accidentally revealing their source code if the configuration is ever changed. The `ScriptAlias` makes this easy by mapping a URL and

designating CGI scripts at the same time. If you do choose to place your CGI scripts in a directory already accessible from the web, do not use `ScriptAlias`. Instead, use `<Directory>`, `SetHandler`, and `Options` as in:

```
<Directory /usr/local/apache2/htdocs/cgi-bin >  
  SetHandler cgi-script  
  Options ExecCGI  
</Directory>
```

This is necessary since multiple *URL-paths* can map to the same filesystem location, potentially bypassing the `ScriptAlias` and revealing the source code of the CGI scripts if they are not restricted by a `Directory` section.

## See also

- [CGI Tutorial](#)



<b>Description:</b>	Maps a URL to a filesystem location using a regular expression and designates the target as a CGI script
<b>Syntax:</b>	<code>ScriptAliasMatch <i>regex file-path directory-path</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_alias

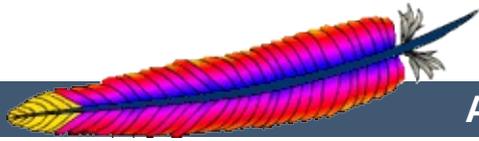
This directive is equivalent to [ScriptAlias](#), but makes use of standard regular expressions, instead of simple prefix matching. The supplied regular expression is matched against the URL-path, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename. For example, to activate the standard /cgi-bin, one might use:

```
ScriptAliasMatch ^/cgi-bin(.*) /usr/local/apache/cgi-bin$1
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_asis`

<b>Description:</b>	Sends files that contain their own HTTP headers
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>asis_module</code>
<b>Source File:</b>	<code>mod_asis.c</code>

## Summary

This module provides the handler `send-asis` which causes Apache to send the document without adding most of the usual HTTP headers.

This can be used to send any kind of data from the server, including redirects and other special HTTP responses, without requiring a cgi-script or an nph script.

For historical reasons, this module will also process any file with the mime type `httpd/send-asis`.

## See also

[mod\\_headers](#)

[mod\\_cern\\_meta](#)

[Apache's Handler Use](#)



In the server configuration file, associate files with the `send-as-is` handler e.g.

```
AddHandler send-as-is asis
```

The contents of any file with a `.asis` extension will then be sent by Apache to the client with almost no changes. In particular, HTTP headers are derived from the file itself according to [mod\\_cgi](#) rules, so an `asis` file must include valid headers, and may also use the `CGI Status:` header to determine the HTTP response code.

Here's an example of a file whose contents are sent as *is* so as to tell the client that a file has redirected.

```
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>
```

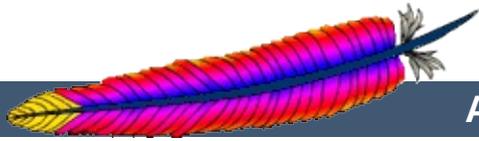
### Notes:

The server always adds a `Date:` and `Server:` header to the data returned to the client, so these should not be included in the file. The server does *not* add a `Last-Modified` header; it probably should.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_auth

<b>Description:</b>	User authentication using text files
<b>Status:</b>	Base
<b>Module Identifier:</b>	auth_module
<b>Source File:</b>	mod_auth.c
<b>Compatibility:</b>	Available only in versions prior to 2.1

## Summary

This module allows the use of HTTP Basic Authentication to restrict access by looking up users in plain text password and group files. Similar functionality and greater scalability is provided by [mod\\_auth\\_dbm](#). HTTP Digest Authentication is provided by [mod\\_auth\\_digest](#).

## See also

[Require](#)

[Satisfy](#)

[AuthName](#)

[AuthType](#)



<b>Description:</b>	Sets whether authorization and authentication are passed to lower level modules
<b>Syntax:</b>	AuthAuthoritative On Off
<b>Default:</b>	AuthAuthoritative On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_auth

Setting the `AuthAuthoritative` directive explicitly to `Off` allows for both authentication and authorization to be passed on to lower level modules (as defined in the `modules.c` files) if there is **no** `userID` or `rule` matching the supplied `userID`. If there is a `userID` and/or `rule` specified; the usual password and access checks will be applied and a failure will give an "Authentication Required" reply.

So if a `userID` appears in the database of more than one module; or if a valid `Require` directive applies to more than one module; then the first module will verify the credentials; and no access is passed on; regardless of the `AuthAuthoritative` setting.

A common use for this is in conjunction with one of the database modules; such as `mod_auth_dbm`, `mod_auth_mysql`, and `mod_auth_anon`. These modules supply the bulk of the user credential checking; but a few (administrator) related accesses fall through to a lower level with a well protected `AuthUserFile`.

By default control is not passed on and an unknown `userID` or `rule` will result in an "Authentication Required" reply. Not setting it thus keeps the system secure and forces an NCSA compliant behaviour.

---

## Security

Do consider the implications of allowing a user to allow fall-through in his .htaccess file; and verify that this is really what you want; Generally it is easier to just secure a single .htpasswd file, than it is to secure a database such as mSQL. Make sure that the [AuthUserFile](#) and the [AuthGroupFile](#) are stored outside the document tree of the web-server; do *not* put them in the directory that they protect. Otherwise, clients will be able to download the [AuthUserFile](#) and the [AuthGroupFile](#).



<b>Description:</b>	Sets the name of a text file containing the list of user groups for authentication
<b>Syntax:</b>	<code>AuthGroupFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Base
<b>Module:</b>	mod_auth

The `AuthGroupFile` directive sets the name of a textual file containing the list of user groups for user authentication. *File-path* is the path to the group file. If it is not absolute, it is treated as relative to the `ServerRoot`.

Each line of the group file contains a groupname followed by a colon, followed by the member usernames separated by spaces.

#### Example:

```
mygroup: bob joe anne
```

Note that searching large text files is *very* inefficient; `AuthDBMGroupFile` provides a much better performance.

#### Security

Make sure that the `AuthGroupFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients may be able to download the `AuthGroupFile`.



<b>Description:</b>	Sets the name of a text file containing the list of users and passwords for authentication
<b>Syntax:</b>	<code>AuthUserFile <i>file-path</i></code>
<b>Context:</b>	directory, <code>.htaccess</code>
<b>Override:</b>	<code>AuthConfig</code>
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_auth</code>

The `AuthUserFile` directive sets the name of a textual file containing the list of users and passwords for user authentication. *File-path* is the path to the user file. If it is not absolute (*i.e.*, if it doesn't begin with a slash), it is treated as relative to the `ServerRoot`.

Each line of the user file contains a username followed by a colon, followed by the encrypted password. If the same user ID is defined multiple times, `mod_auth` will use the first occurrence to verify the password.

The utility `htpasswd` which is installed as part of the binary distribution, or which can be found in `src/support`, is used to maintain this password file. See the [man page](#) for more details. In short:

Create a password file `Filename` with `username` as the initial ID. It will prompt for the password:

```
htpasswd -c Filename username
```

Add or modify `username2` in the password file `Filename`:

```
htpasswd Filename username2
```

Note that searching large text files is *very* inefficient; [AuthDBMUserFile](#) should be used instead.

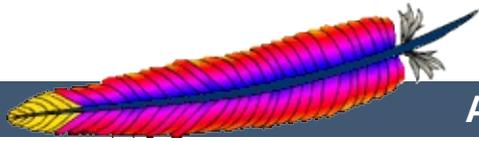
### Security

Make sure that the [AuthUserFile](#) is stored outside the document tree of the web-server. Do **not** put it in the directory that it protects. Otherwise, clients may be able to download the [AuthUserFile](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_auth\_anon

<b>Description:</b>	Allows "anonymous" user access to authenticated areas
<b>Status:</b>	Extension
<b>Module Identifier:</b>	auth_anon_module
<b>Source File:</b>	mod_auth_anon.c
<b>Compatibility:</b>	Available only in versions prior to 2.1

### Summary

This module does access control in a manner similar to anonymous-ftp sites; *i.e.* have a 'magic' user id 'anonymous' and the email address as a password. These email addresses can be logged.

Combined with other (database) access control methods, this allows for effective user tracking and customization according to a user profile while still keeping the site open for 'unregistered' users. One advantage of using Auth-based user tracking is that, unlike magic-cookies and funny URL pre/postfixes, it is completely browser independent and it allows users to share URLs.



The example below (when combined with the Auth directives of a htpasswd-file based (or GDM, mSQL etc.) base access control system allows users in as 'guests' with the following properties:

- It insists that the user enters a userID.  
(Anonymous\_NoUserID)
- It insists that the user enters a password.  
(Anonymous\_MustGiveEmail)
- The password entered must be a valid email address, ie. contain at least one '@' and a '.'.  
(Anonymous\_VerifyEmail)
- The userID must be one of anonymous guest www test welcome and comparison is **not** case sensitive.  
(Anonymous)
- And the Email addresses entered in the passwd field are logged to the error log file. (Anonymous\_LogEmail)

### Excerpt of httpd.conf:

```
Anonymous_NoUserID off
Anonymous_MustGiveEmail on
Anonymous_VerifyEmail on
Anonymous_LogEmail on
Anonymous anonymous guest www test welcome

AuthName "Use 'anonymous' & Email address for guest entry"
AuthType basic

# An AuthUserFile/AuthDBUserFile/AuthDBMUserFile
# directive must be specified, or use
# Anonymous_Authoritative for public access.
# In the .htaccess for the public directory, add:
<Files *>
    Order Deny,Allow
    Allow from all

    Require valid-user
</Files>
```



<b>Description:</b>	Specifies userIDs that are allowed access without password verification
<b>Syntax:</b>	Anonymous <i>user</i> [ <i>user</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

A list of one or more 'magic' userIDs which are allowed access without password verification. The userIDs are space separated. It is possible to use the ' and " quotes to allow a space in a userID as well as the \ escape character.

Please note that the comparison is **case-IN-sensitive**.

I strongly suggest that the magic username 'anonymous' is always one of the allowed userIDs.

#### Example:

```
Anonymous anonymous "Not Registered" "I don't know"
```

This would allow the user to enter without password verification by using the userIDs "anonymous", "AnonyMous", "Not Registered" and "I Don't Know".



## AuthConfig: Anonymous\_Authoritative Directive

<b>Description:</b>	Configures if authorization will fall-through to other methods
<b>Syntax:</b>	Anonymous_Authoritative On Off
<b>Default:</b>	Anonymous_Authoritative Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

When set On, there is no fall-through to other authentication methods. So if a userID does not match the values specified in the [Anonymous](#) directive, access is denied.

Be sure you know what you are doing when you decide to switch it on. And remember that the order in which the Authentication modules are queried is defined in the modules.c files at compile time.



## Anonymous\_LogEmail

<b>Description:</b>	Sets whether the password entered will be logged in the error log
<b>Syntax:</b>	Anonymous_LogEmail On Off
<b>Default:</b>	Anonymous_LogEmail On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

When set On, the default, the 'password' entered (which hopefully contains a sensible email address) is logged in the error log.



<b>Description:</b>	Specifies whether blank passwords are allowed
<b>Syntax:</b>	Anonymous_MustGiveEmail On Off
<b>Default:</b>	Anonymous_MustGiveEmail On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

Specifies whether the user must specify an email address as the password. This prohibits blank passwords.



<b>Description:</b>	Sets whether the userID field may be empty
<b>Syntax:</b>	Anonymous_NoUserID On Off
<b>Default:</b>	Anonymous_NoUserID Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

When set On, users can leave the userID (and perhaps the password field) empty. This can be very convenient for MS-Explorer users who can just hit return or click directly on the OK button; which seems a natural reaction.



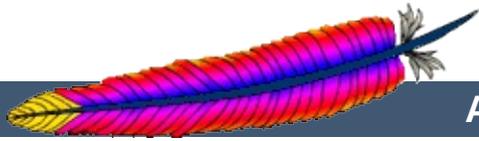
<b>Description:</b>	Sets whether to check the password field for a correctly formatted email address
<b>Syntax:</b>	Anonymous_VerifyEmail On Off
<b>Default:</b>	Anonymous_VerifyEmail Off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_anon

When set On the 'password' entered is checked for at least one '@' and a '.' to encourage users to enter valid email addresses (see the above [Anonymous\\_LogEmail](#)).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_auth\_dbm

<b>Description:</b>	Provides for user authentication using DBM files
<b>Status:</b>	Extension
<b>Module Identifier:</b>	auth_dbm_module
<b>Source File:</b>	mod_auth_dbm.c
<b>Compatibility:</b>	Available only in versions prior to 2.1

## Summary

This module provides for HTTP Basic Authentication, where the usernames and passwords are stored in DBM type database files. It is an alternative to the plain text password files provided by [mod\\_auth](#).

## See also

[AuthName](#)  
[AuthType](#)  
[Require](#)  
[Satisfy](#)



<b>Description:</b>	Sets whether authentication and authorization will be passed on to lower level modules
<b>Syntax:</b>	AuthDBMAuthoritative On Off
<b>Default:</b>	AuthDBMAuthoritative On
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_dbm

Setting the `AuthDBMAuthoritative` directive explicitly to `Off` allows for both authentication and authorization to be passed on to lower level modules (as defined in the `modules.c` files) if there is **no** `userID` or `rule` matching the supplied `userID`. If there is a `userID` and/or `rule` specified; the usual password and access checks will be applied and a failure will give an "Authentication Required" reply.

So if a `userID` appears in the database of more than one module; or if a valid `Require` directive applies to more than one module; then the first module will verify the credentials; and no access is passed on; regardless of the `AuthDBMAuthoritative` setting.

A common use for this is in conjunction with one of the basic auth modules; such as `mod_auth`. Whereas this DBM module supplies the bulk of the user credential checking; a few (administrator) related accesses fall through to a lower level with a well protected `.htpasswd` file.

By default, control is not passed on and an unknown `userID` or `rule` will result in an "Authentication Required" reply. Not setting it thus keeps the system secure and forces an NCSA compliant behaviour.

---

**Security:**

Do consider the implications of allowing a user to allow fall-through in his `.htaccess` file; and verify that this is really what you want; Generally it is easier to just secure a single `.htpasswd` file, than it is to secure a database which might have more access interfaces.



<b>Description:</b>	Sets the name of the database file containing the list of user groups for authentication
<b>Syntax:</b>	<code>AuthDBMGroupFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_dbm

The `AuthDBMGroupFile` directive sets the name of a DBM file containing the list of user groups for user authentication. *File-path* is the absolute path to the group file.

The group file is keyed on the username. The value for a user is a comma-separated list of the groups to which the users belongs. There must be no whitespace within the value, and it must never contain any colons.

Security: make sure that the `AuthDBMGroupFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients will be able to download the `AuthDBMGroupFile` unless otherwise protected.

Combining Group and Password DBM files: In some cases it is easier to manage a single database which contains both the password and group details for each user. This simplifies any support programs that need to be written: they now only have to deal with writing to and locking a single DBM file. This can be accomplished by first setting the group and password files to point to the same DBM:

```
AuthDBMGroupFile /www/userbase  
AuthDBMUserFile /www/userbase
```

The key for the single DBM is the username. The value consists of

```
Unix Crypt-ed Password:List of Groups[:(ignored)]
```

The password section contains the encrypted password as before. This is followed by a colon and the comma separated list of groups. Other data may optionally be left in the DBM file after another colon; it is ignored by the authentication module. This is what [www.telescope.org](http://www.telescope.org) uses for its combined password and group database.



## AuthDBMType Directive

<b>Description:</b>	Sets the type of database file that is used to store passwords
<b>Syntax:</b>	AuthDBMType default   SDBM   GDBM   NDBM   DB
<b>Default:</b>	AuthDBMType default
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_dbm
<b>Compatibility:</b>	Available in version 2.0.30 and later.

Sets the type of database file that is used to store the passwords. The default database type is determined at compile time. The availability of other types of database files also depends on [compile-time settings](#).

It is crucial that whatever program you use to create your password files is configured to use the same type of database.



<b>Description:</b>	Sets the name of a database file containing the list of users and passwords for authentication
<b>Syntax:</b>	<code>AuthDBMUserFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_auth_dbm

The `AuthDBMUserFile` directive sets the name of a DBM file containing the list of users and passwords for user authentication. *File-path* is the absolute path to the user file.

The user file is keyed on the username. The value for a user is the encrypted password, optionally followed by a colon and arbitrary data. The colon and the data following it will be ignored by the server.

#### **Security:**

Make sure that the `AuthDBMUserFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients will be able to download the `AuthDBMUserFile`.

Important compatibility note: The implementation of "dbmopen" in the apache modules reads the string length of the hashed values from the DBM data structures, rather than relying upon the string being NULL-appended. Some applications, such as the Netscape web server, rely upon the string being NULL-appended, so if you are having trouble using DBM files interchangeably between applications this may be a part of the problem.

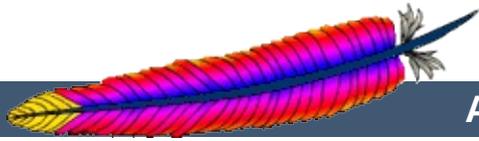
A perl script called `dbmmanage` is included with Apache. This

program can be used to create and update DBM format password files for use with this module.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_auth\_digest

<b>Description:</b>	User authentication using MD5 Digest Authentication.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	auth_digest_module
<b>Source File:</b>	mod_auth_digest.c

## Summary

This module implements HTTP Digest Authentication. However, it has not been extensively tested and is therefore marked experimental.

## See also

[AuthName](#)  
[AuthType](#)  
[Require](#)  
[Satisfy](#)



## Using Digest Authentication

Using MD5 Digest authentication is very simple. Simply set up authentication normally, using `AuthType Digest` and [AuthDigestFile](#) instead of the normal `AuthType Basic` and [AuthUserFile](#); also, replace any [AuthGroupFile](#) with [AuthDigestGroupFile](#). Then add a [AuthDigestDomain](#) directive containing at least the root URI(s) for this protection space.

Appropriate user (text) files can be created using the [htdigest](#) tool.

### Example:

```
<Location /private/>
  AuthType Digest
  AuthName "private area"
  AuthDigestDomain /private/ http://mirror.my.dom/private2/
  AuthDigestFile /web/auth/.digest_pw
  Require valid-user
</Location>
```

### Note

Digest authentication provides a more secure password system than Basic authentication, but only works with supporting browsers. As of November 2002, the major browsers that support digest authentication are [Opera](#), [MS Internet Explorer](#) (fails when used with a query string - see "[Working with MS Internet Explorer](#)" below for a workaround), [Amaya](#), [Mozilla](#) and [Netscape](#) since version 7. Since digest authentication is not as widely implemented as basic authentication, you should use it only in controlled environments.



The Digest authentication implementation in previous Internet Explorer for Windows versions (5 and 6) had issues, namely that GET requests with a query string were not RFC compliant. There are a few ways to work around this issue.

The first way is to use POST requests instead of GET requests to pass data to your program. This method is the simplest approach if your application can work with this limitation.

Since version 2.0.51 Apache also provides a workaround in the `AuthDigestEnableQueryStringHack` environment variable. If `AuthDigestEnableQueryStringHack` is set for the request, Apache will take steps to work around the MSIE bug and remove the query string from the digest comparison. Using this method would look similar to the following.

### Using Digest Authentication with MSIE:

```
BrowserMatch "MSIE" AuthDigestEnableQueryStringHack=On
```

This workaround is not necessary for MSIE 7, though enabling it does not cause any compatibility issues or significant overhead.

See the [BrowserMatch](#) directive for more details on conditionally setting environment variables



## AuthDigestAlgorithm Directive

<b>Description:</b>	Selects the algorithm used to calculate the challenge and response hashes in digest authentication
<b>Syntax:</b>	AuthDigestAlgorithm MD5 MD5-sess
<b>Default:</b>	AuthDigestAlgorithm MD5
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestAlgorithm` directive selects the algorithm used to calculate the challenge and response hashes.

MD5-sess is not correctly implemented yet.



<b>Description:</b>	URIs that are in the same protection space for digest authentication
<b>Syntax:</b>	AuthDigestDomain <i>URI</i> [ <i>URI</i> ] ...
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestDomain` directive allows you to specify one or more URIs which are in the same protection space (*i.e.* use the same realm and username/password info). The specified URIs are prefixes, *i.e.* the client will assume that all URIs "below" these are also protected by the same username/password. The URIs may be either absolute URIs (*i.e.* including a scheme, host, port, etc) or relative URIs.

This directive *should* always be specified and contain at least the (set of) root URI(s) for this space. Omitting to do so will cause the client to send the Authorization header for *every request* sent to this server. Apart from increasing the size of the request, it may also have a detrimental effect on performance if `AuthDigestNcCheck` is on.

The URIs specified can also point to different servers, in which case clients (which understand this) will then share username/password info across multiple servers without prompting the user each time.



<b>Description:</b>	Location of the text file containing the list of users and encoded passwords for digest authentication
<b>Syntax:</b>	<code>AuthDigestFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestFile` directive sets the name of a textual file containing the list of users and encoded passwords for digest authentication. *File-path* is the absolute path to the user file.

The digest file uses a special format. Files in this format can be created using the `htdigest` utility found in the `support/` subdirectory of the Apache distribution.



## AuthDigestGroupFile Directive

<b>Description:</b>	Name of the text file containing the list of groups for digest authentication
<b>Syntax:</b>	<code>AuthDigestGroupFile <i>file-path</i></code>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestGroupFile` directive sets the name of a textual file containing the list of groups and their members (user names). *File-path* is the absolute path to the group file.

Each line of the group file contains a groupname followed by a colon, followed by the member usernames separated by spaces. Example:

```
mygroup: bob joe anne
```

Note that searching large text files is *very* inefficient.

**Security:**

Make sure that the `AuthGroupFile` is stored outside the document tree of the web-server; do *not* put it in the directory that it protects. Otherwise, clients may be able to download the `AuthGroupFile`.



## AuthDigestNcCheck Directive

**Description:** Enables or disables checking of the nonce-count sent by the server

**Syntax:** AuthDigestNcCheck On|Off

**Default:** AuthDigestNcCheck Off

**Context:** server config

**Status:** Experimental

**Module:** mod\_auth\_digest

Not implemented yet.



<b>Description:</b>	Determines how the nonce is generated
<b>Syntax:</b>	AuthDigestNonceFormat <i>format</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

Not implemented yet.



<b>Description:</b>	How long the server nonce is valid
<b>Syntax:</b>	<code>AuthDigestNonceLifetime</code> <i>seconds</i>
<b>Default:</b>	<code>AuthDigestNonceLifetime</code> 300
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestNonceLifetime` directive controls how long the server nonce is valid. When the client contacts the server using an expired nonce the server will send back a 401 with `stale=true`. If *seconds* is greater than 0 then it specifies the amount of time for which the nonce is valid; this should probably never be set to less than 10 seconds. If *seconds* is less than 0 then the nonce never expires.



<b>Description:</b>	Determines the quality-of-protection to use in digest authentication
<b>Syntax:</b>	AuthDigestQop none auth auth-int [auth auth-int]
<b>Default:</b>	AuthDigestQop auth
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_digest

The `AuthDigestQop` directive determines the *quality-of-protection* to use. `auth` will only do authentication (username/password); `auth-int` is authentication plus integrity checking (an MD5 hash of the entity is also computed and checked); `none` will cause the module to use the old RFC-2069 digest algorithm (which does not include integrity checking). Both `auth` and `auth-int` may be specified, in which the case the browser will choose which of these to use. `none` should only be used if the browser for some reason does not like the challenge it receives otherwise.

`auth-int` is not implemented yet.



**Description:** The amount of shared memory to allocate for keeping track of clients

**Syntax:** AuthDigestShmemSize *size*

**Default:** AuthDigestShmemSize 1000

**Context:** server config

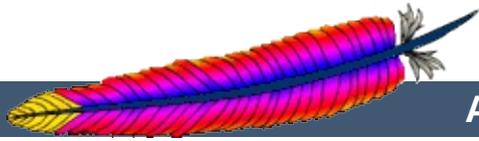
**Status:** Experimental

**Module:** mod\_auth\_digest

The `AuthDigestShmemSize` directive defines the amount of shared memory, that will be allocated at the server startup for keeping track of clients. Note that the shared memory segment cannot be set less than the space that is necessary for tracking at least *one* client. This value is dependant on your system. If you want to find out the exact value, you may simply set `AuthDigestShmemSize` to the value of 0 and read the error message after trying to start the server.

The *size* is normally expressed in Bytes, but you may let the number follow a K or an M to express your value as KBytes or MBytes. For example, the following directives are all equivalent:

```
AuthDigestShmemSize 1048576
AuthDigestShmemSize 1024K
AuthDigestShmemSize 1M
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_auth_ldap`

<b>Description:</b>	Allows an LDAP directory to be used to store the database for HTTP Basic authentication.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>auth_ldap_module</code>
<b>Source File:</b>	<code>mod_auth_ldap.c</code>
<b>Compatibility:</b>	Available in version 2.0.41 and later

## Summary

`mod_auth_ldap` supports the following features:

- Known to support the [OpenLDAP SDK](#) (both 1.x and 2.x), [Novell LDAP SDK](#) and the [iPlanet \(Netscape\) SDK](#).
- Complex authorization policies can be implemented by representing the policy with LDAP filters.
- Support for Microsoft FrontPage allows FrontPage users to control access to their webs, while retaining LDAP for user authentication.
- Uses extensive caching of LDAP operations via [mod\\_ldap](#).
- Support for LDAP over SSL (requires the Netscape SDK) or TLS (requires the OpenLDAP 2.x SDK or Novell LDAP SDK).

## See also

[mod\\_ldap](#)



- 
- [Operation](#)
    - [The Authentication Phase](#)
    - [The Authorization Phase](#)
  - [The Require Directives](#)
    - [Require valid-user](#)
    - [Require user](#)
    - [Require group](#)
    - [Require dn](#)
    - [Require ldap-attribute](#)
  - [Examples](#)
  - [Using TLS](#)
  - [Using SSL](#)
  - [Using Microsoft FrontPage with mod\\_auth\\_ldap](#)
    - [How It Works](#)
    - [Caveats](#)



There are two phases in granting access to a user. The first phase is authentication, in which [mod\\_auth\\_ldap](#) verifies that the user's credentials are valid. This is also called the *search/bind* phase. The second phase is authorization, in which [mod\\_auth\\_ldap](#) determines if the authenticated user is allowed access to the resource in question. This is also known as the *compare* phase.

## The Authentication Phase

During the authentication phase, [mod\\_auth\\_ldap](#) searches for an entry in the directory that matches the username that the HTTP client passes. If a single unique match is found, then [mod\\_auth\\_ldap](#) attempts to bind to the directory server using the DN of the entry plus the password provided by the HTTP client. Because it does a search, then a bind, it is often referred to as the search/bind phase. Here are the steps taken during the search/bind phase.

1. Generate a search filter by combining the attribute and filter provided in the [AuthLDAPURL](#) directive with the username passed by the HTTP client.
2. Search the directory using the generated filter. If the search does not return exactly one entry, deny or decline access.
3. Fetch the distinguished name of the entry retrieved from the search and attempt to bind to the LDAP server using the DN and the password passed by the HTTP client. If the bind is unsuccessful, deny or decline access.

The following directives are used during the search/bind phase

[AuthLDAPURL](#)

Specifies the LDAP server, the base DN, the attribute to use in the search, as well as the extra search filter to

	use.
<a href="#">AuthLDAPBindDN</a>	An optional DN to bind with during the search phase.
<a href="#">AuthLDAPBindPassword</a>	An optional password to bind with during the search phase.

## The Authorization Phase

During the authorization phase, [mod\\_auth\\_ldap](#) attempts to determine if the user is authorized to access the resource. Many of these checks require [mod\\_auth\\_ldap](#) to do a compare operation on the LDAP server. This is why this phase is often referred to as the compare phase. [mod\\_auth\\_ldap](#) accepts the following [Require](#) directives to determine if the credentials are acceptable:

- Grant access if there is a [Require valid-user](#) directive.
- Grant access if there is a [Require user](#) directive, and the username in the directive matches the username passed by the client.
- Grant access if there is a [Require dn](#) directive, and the DN in the directive matches the DN fetched from the LDAP directory.
- Grant access if there is a [Require group](#) directive, and the DN fetched from the LDAP directory (or the username passed by the client) occurs in the LDAP group.
- Grant access if there is a [Require ldap-attribute](#) directive, and the attribute fetched from the LDAP directory matches the given value.
- otherwise, deny or decline access

[mod\\_auth\\_ldap](#) uses the following directives during the compare phase:

<a href="#">AuthLDAPURL</a>	The attribute specified in the
-----------------------------	--------------------------------

URL is used in compare operations for the Require user operation.

AuthLDAPCompareDNOnServer

Determines the behavior of the Require dn directive.

AuthLDAPGroupAttribute

Determines the attribute to use for comparisons in the Require group directive.

AuthLDAPGroupAttributeIsDN

Specifies whether to use the user DN or the username when doing comparisons for the Require group directive.



Apache's [Require](#) directives are used during the authorization phase to ensure that a user is allowed to access a resource.

## Require valid-user

If this directive exists, [mod\\_auth\\_ldap](#) grants access to any user that has successfully authenticated during the search/bind phase.

## Require user

The `Require user` directive specifies what usernames can access the resource. Once [mod\\_auth\\_ldap](#) has retrieved a unique DN from the directory, it does an LDAP compare operation using the username specified in the `Require user` to see if that username is part of the just-fetched LDAP entry. Multiple users can be granted access by putting multiple usernames on the line, separated with spaces. If a username has a space in it, then it must be surrounded with double quotes. Multiple users can also be granted access by using multiple `Require user` directives, with one user per line. For example, with a [AuthLDAPURL](#) of `ldap://ldap/o=Airius?cn` (i.e., `cn` is used for searches), the following `Require` directives could be used to restrict access:

```
Require user "Barbara Jenson"  
Require user "Fred User"  
Require user "Joe Manager"
```

Because of the way that [mod\\_auth\\_ldap](#) handles this directive, Barbara Jenson could sign on as *Barbara Jenson*, *Babs Jenson* or any other `cn` that she has in her LDAP entry. Only the single `Require user` line is needed to support all values of the attribute in the user's entry.

If the `uid` attribute was used instead of the `cn` attribute in the URL

above, the above three lines could be condensed to

```
Require user bjenson fuser jmanager
```

## Require group

This directive specifies an LDAP group whose members are allowed access. It takes the distinguished name of the LDAP group. Note: Do not surround the group name with quotes. For example, assume that the following entry existed in the LDAP directory:

```
dn: cn=Administrators, o=Airius
objectClass: groupOfUniqueNames
uniqueMember: cn=Barbara Jenson, o=Airius
uniqueMember: cn=Fred User, o=Airius
```

The following directive would grant access to both Fred and Barbara:

```
Require group cn=Administrators, o=Airius
```

Behavior of this directive is modified by the [AuthLDAPGroupAttribute](#) and [AuthLDAPGroupAttributeIsDN](#) directives.

## Require dn

The `Require dn` directive allows the administrator to grant access based on distinguished names. It specifies a DN that must match for access to be granted. If the distinguished name that was retrieved from the directory server matches the distinguished name in the `Require dn`, then authorization is granted. Note: do not surround the distinguished name with quotes.

The following directive would grant access to a specific DN:

```
Require dn cn=Barbara Jenson, o=Airius
```

Behavior of this directive is modified by the [AuthLDAPCompareDNOnServer](#) directive.

## Require ldap-attribute

The `Require ldap-attribute` directive allows the administrator to grant access based on attributes of the authenticated user in the LDAP directory. If the attribute in the directory matches the value given in the configuration, access is granted.

The following directive would grant access to anyone with the attribute `employeeType = active`

```
Require ldap-attribute employeeType=active
```

Multiple attribute/value pairs can be specified on the same line separated by spaces or they can be specified in multiple `Require ldap-attribute` directives. The effect of listing multiple attribute/values pairs is an OR operation. Access will be granted if any of the listed attribute values match the value of a corresponding attribute in the user object. If the value of the attribute contains a space, only the value must be within double quotes.

The following directive would grant access to anyone with the city attribute equal to "San Jose" or status equal to "Active"

```
Require ldap-attribute city="San Jose" status=active
```



- Grant access to anyone who exists in the LDAP directory, using their UID for searches.

```
AuthLDAPURL "ldap://ldap1.airius.com:389/ou=People,
o=Airius?uid?sub?(objectClass=*)"
Require valid-user
```

- The next example is the same as above; but with the fields that have useful defaults omitted. Also, note the use of a redundant LDAP server.

```
AuthLDAPURL "ldap://ldap1.airius.com
ldap2.airius.com/ou=People, o=Airius"
Require valid-user
```

- The next example is similar to the previous one, but it uses the common name instead of the UID. Note that this could be problematical if multiple people in the directory share the same cn, because a search on cn **must** return exactly one entry. That's why this approach is not recommended: it's a better idea to choose an attribute that is guaranteed unique in your directory, such as uid.

```
AuthLDAPURL "ldap://ldap.airius.com/ou=People, o=Airius?
cn"
Require valid-user
```

- Grant access to anybody in the Administrators group. The users must authenticate using their UID.

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid
Require group cn=Administrators, o=Airius
```

- The next example assumes that everyone at Airius who carries an alphanumeric pager will have an LDAP attribute of

qpagePagerID. The example will grant access only to people (authenticated via their UID) who have alphanumeric pagers:

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid??  
(qpagePagerID=*)  
Require valid-user
```

- The next example demonstrates the power of using filters to accomplish complicated administrative requirements. Without filters, it would have been necessary to create a new LDAP group and ensure that the group's members remain synchronized with the pager users. This becomes trivial with filters. The goal is to grant access to anyone who has a filter, plus grant access to Joe Manager, who doesn't have a pager, but does need to access the same resource:

```
AuthLDAPURL ldap://ldap.airius.com/o=Airius?uid??(|  
(qpagePagerID=*)(uid=jmanager))  
Require valid-user
```

This last may look confusing at first, so it helps to evaluate what the search filter will look like based on who connects, as shown below. The text in blue is the part that is filled in using the attribute specified in the URL. The text in red is the part that is filled in using the filter specified in the URL. The text in green is filled in using the information that is retrieved from the HTTP client. If Fred User connects as *fuser*, the filter would look like

```
(&( |(qpagePagerID=*)(uid=jmanager))(uid=fuser))
```

The above search will only succeed if *fuser* has a pager. When Joe Manager connects as *jmanager*, the filter looks like

```
(&( |(qpagePagerID=*)(uid=jmanager))(uid=jmanager))
```

The above search will succeed whether *jmanager* has a pager or not.



To use TLS, see the [mod\\_ldap](#) directives [LDAPTrustedCA](#) and [LDAPTrustedCAType](#).



To use SSL, see the [mod\\_ldap](#) directives [LDAPTrustedCA](#) and [LDAPTrustedCAType](#).

To specify a secure LDAP server, use *ldaps://* in the [AuthLDAPURL](#) directive, instead of *ldap://*.



Normally, FrontPage uses FrontPage-web-specific user/group files (i.e., the `mod_auth` module) to handle all authentication.

Unfortunately, it is not possible to just change to LDAP authentication by adding the proper directives, because it will break the *Permissions* forms in the FrontPage client, which attempt to modify the standard text-based authorization files.

Once a FrontPage web has been created, adding LDAP authentication to it is a matter of adding the following directives to every `.htaccess` file that gets created in the web

```
AuthLDAPURL          "the url"  
AuthLDAPAuthoritative off  
AuthLDAPFrontPageHack on
```

`AuthLDAPAuthoritative` must be off to allow `mod_auth_ldap` to decline group authentication so that Apache will fall back to file authentication for checking group membership. This allows the FrontPage-managed group file to be used.

## How It Works

FrontPage restricts access to a web by adding the `Require valid-user` directive to the `.htaccess` files. If `AuthLDAPFrontPageHack` is not on, the `Require valid-user` directive will succeed for any user who is valid as *far as LDAP is concerned*. This means that anybody who has an entry in the LDAP directory is considered a valid user, whereas FrontPage considers only those people in the local user file to be valid. The purpose of the hack is to force Apache to consult the local user file (which is managed by FrontPage) - instead of LDAP - when handling the `Require valid-user` directive.

Once directives have been added as specified above, FrontPage

users will be able to perform all management operations from the FrontPage client.

## Caveats

- When choosing the LDAP URL, the attribute to use for authentication should be something that will also be valid for putting into a `mod_auth` user file. The user ID is ideal for this.
- When adding users via FrontPage, FrontPage administrators should choose usernames that already exist in the LDAP directory (for obvious reasons). Also, the password that the administrator enters into the form is ignored, since Apache will actually be authenticating against the password in the LDAP database, and not against the password in the local user file. This could cause confusion for web administrators.
- Apache must be compiled with `mod_auth` in order to use FrontPage support. This is because Apache will still use the `mod_auth` group file for determine the extent of a user's access to the FrontPage web.
- The directives must be put in the `.htaccess` files. Attempting to put them inside `<Location>` or `<Directory>` directives won't work. This is because `mod_auth_ldap` has to be able to grab the `AuthUserFile` directive that is found in FrontPage `.htaccess` files so that it knows where to look for the valid user list. If the `mod_auth_ldap` directives aren't in the same `.htaccess` file as the FrontPage directives, then the hack won't work, because `mod_auth_ldap` will never get a chance to process the `.htaccess` file, and won't be able to find the FrontPage-managed user file.



<b>Description:</b>	Prevent other authentication modules from authenticating the user if this one fails
<b>Syntax:</b>	AuthLDAPAuthoritative on off
<b>Default:</b>	AuthLDAPAuthoritative on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

Set to off if this module should let other authentication modules attempt to authenticate the user, should authentication with this module fail. Control is only passed on to lower modules if there is no DN or rule that matches the supplied user name (as passed by the client).



<b>Description:</b>	Optional DN to use in binding to the LDAP server
<b>Syntax:</b>	AuthLDAPBindDN <i>distinguished-name</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

An optional DN used to bind to the server when searching for entries. If not provided, [mod\\_auth\\_ldap](#) will use an anonymous bind.



<b>Description:</b>	Password used in conjunction with the bind DN
<b>Syntax:</b>	AuthLDAPBindPassword <i>password</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

A bind password to use in conjunction with the bind DN. Note that the bind password is probably sensitive data, and should be properly protected. You should only use the [AuthLDAPBindDN](#) and [AuthLDAPBindPassword](#) if you absolutely need them to search the directory.



<b>Description:</b>	Language to charset conversion configuration file
<b>Syntax:</b>	<code>AuthLDAPCharsetConfig</code> <i>file-path</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_auth_ldap</code>

The `AuthLDAPCharsetConfig` directive sets the location of the language to charset conversion configuration file. *File-path* is relative to the `ServerRoot`. This file specifies the list of language extensions to character sets. Most administrators use the provided `charset.conv` file, which associates common language extensions to character sets.

The file contains lines in the following format:

```
Language-Extension charset [Language-String] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (#) are ignored.



<b>Description:</b>	Use the LDAP server to compare the DN's
<b>Syntax:</b>	AuthLDAPCompareDNOnServer on off
<b>Default:</b>	AuthLDAPCompareDNOnServer on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

When set, [mod\\_auth\\_ldap](#) will use the LDAP server to compare the DN's. This is the only foolproof way to compare DN's. [mod\\_auth\\_ldap](#) will search the directory for the DN specified with the [Require dn](#) directive, then, retrieve the DN and compare it with the DN retrieved from the user entry. If this directive is not set, [mod\\_auth\\_ldap](#) simply does a string comparison. It is possible to get false negatives with this approach, but it is much faster. Note the [mod\\_ldap](#) cache can speed up DN comparison in most situations.



<b>Description:</b>	When will the module de-reference aliases
<b>Syntax:</b>	AuthLDAPDereferenceAliases never   searching   finding   always
<b>Default:</b>	AuthLDAPDereferenceAliases Always
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

This directive specifies when `mod_auth_ldap` will de-reference aliases during LDAP operations. The default is `always`.



<b>Description:</b>	Turn on or off LDAP authentication
<b>Syntax:</b>	AuthLDAPEnabled on off
<b>Default:</b>	AuthLDAPEnabled on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

Set to off to disable [mod\\_auth\\_ldap](#) in certain directories. This is useful if you have [mod\\_auth\\_ldap](#) enabled at or near the top of your tree, but want to disable it completely in certain locations.



<b>Description:</b>	Allow LDAP authentication to work with MS FrontPage
<b>Syntax:</b>	AuthLDAPFrontPageHack on off
<b>Default:</b>	AuthLDAPFrontPageHack off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

See the section on [using Microsoft FrontPage](#) with [mod\\_auth\\_ldap](#).



## AuthLDAPGroupAttribute Directive

<b>Description:</b>	LDAP attributes used to check for group membership
<b>Syntax:</b>	AuthLDAPGroupAttribute <i>attribute</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

This directive specifies which LDAP attributes are used to check for group membership. Multiple attributes can be used by specifying this directive multiple times. If not specified, then [mod\\_auth\\_ldap](#) uses the member and uniquemember attributes.



<b>Description:</b>	Use the DN of the client username when checking for group membership
<b>Syntax:</b>	AuthLDAPGroupAttributeIsDN on off
<b>Default:</b>	AuthLDAPGroupAttributeIsDN on
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

When set on, this directive says to use the distinguished name of the client username when checking for group membership. Otherwise, the username will be used. For example, assume that the client sent the username bjenson, which corresponds to the LDAP DN cn=Babs Jenson, o=Airius. If this directive is set, [mod\\_auth\\_ldap](#) will check if the group has cn=Babs Jenson, o=Airius as a member. If this directive is not set, then [mod\\_auth\\_ldap](#) will check if the group has bjenson as a member.



<b>Description:</b>	Use the DN of the client username to set the REMOTE_USER environment variable
<b>Syntax:</b>	AuthLDAPRemoteUserIsDN on off
<b>Default:</b>	AuthLDAPRemoteUserIsDN off
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

If this directive is set to on, the value of the REMOTE\_USER environment variable will be set to the full distinguished name of the authenticated user, rather than just the username that was passed by the client. It is turned off by default.



<b>Description:</b>	URL specifying the LDAP search parameters
<b>Syntax:</b>	AuthLDAPUrl <i>url</i>
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Experimental
<b>Module:</b>	mod_auth_ldap

An RFC 2255 URL which specifies the LDAP search parameters to use. The syntax of the URL is

```
ldap://host:port/basedn?attribute?scope?filter
```

## ldap

For regular ldap, use the string ldap. For secure LDAP, use ldaps instead. Secure LDAP is only available if Apache was linked to an LDAP library with SSL support.

## host:port

The name/port of the ldap server (defaults to localhost:389 for ldap, and localhost:636 for ldaps). To specify multiple, redundant LDAP servers, just list all servers, separated by spaces. `mod_auth_ldap` will try connecting to each server in turn, until it makes a successful connection.

Once a connection has been made to a server, that connection remains active for the life of the httpd process, or until the LDAP server goes down.

If the LDAP server goes down and breaks an existing connection, `mod_auth_ldap` will attempt to re-connect, starting with the primary server, and trying each redundant server in turn. Note that this is different than a true round-

robin search.

### **basedn**

The DN of the branch of the directory where all searches should start from. At the very least, this must be the top of your directory tree, but could also specify a subtree in the directory.

### **attribute**

The attribute to search for. Although RFC 2255 allows a comma-separated list of attributes, only the first attribute will be used, no matter how many are provided. If no attributes are provided, the default is to use `uid`. It's a good idea to choose an attribute that will be unique across all entries in the subtree you will be using.

### **scope**

The scope of the search. Can be either `one` or `sub`. Note that a scope of `base` is also supported by RFC 2255, but is not supported by this module. If the scope is not provided, or if `base` scope is specified, the default is to use a scope of `sub`.

### **filter**

A valid LDAP search filter. If not provided, defaults to `(objectClass=*)`, which will search for all objects in the tree. Filters are limited to approximately 8000 characters (the definition of `MAX_STRING_LEN` in the Apache source code). This should be than sufficient for any application.

When doing searches, the attribute, filter and username passed by the HTTP client are combined to create a search filter that looks like `(&(filter)(attribute=username))`.

For example, consider an URL of `ldap://ldap.airius.com/o=Airius?cn?sub?(posixid=*)`. When a client attempts to connect using a

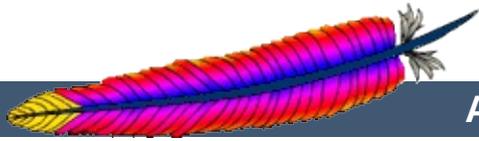
username of Babs Jenson, the resulting search filter will be (& (posixid=\*)(cn=Babs Jenson)).

See above for examples of [AuthLDAPURL](#) URLs.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_autoindex`

<b>Description:</b>	Generates directory indexes, automatically, similar to the Unix <code>ls</code> command or the Win32 <code>dir</code> shell command
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>autoindex_module</code>
<b>Source File:</b>	<code>mod_autoindex.c</code>

## Summary

The index of a directory can come from one of two sources:

- A file written by the user, typically called `index.html`. The [DirectoryIndex](#) directive sets the name of this file. This is controlled by [mod\\_dir](#).
- Otherwise, a listing generated by the server. The other directives control the format of this listing. The [AddIcon](#), [AddIconByEncoding](#) and [AddIconByType](#) are used to set a list of icons to display for various file types; for each file listed, the first icon listed that matches the file is displayed. These are controlled by [mod\\_autoindex](#).

The two functions are separated so that you can completely remove (or replace) automatic index generation should you want to.

Automatic index generation is enabled with using `Options +Indexes`. See the [Options](#) directive for more details.

If the [FancyIndexing](#) option is given with the [IndexOptions](#) directive, the column headers are links that control the order of the display. If you select a header link, the listing will be regenerated, sorted by the values in that column. Selecting the same header repeatedly toggles between ascending and descending order. These

column header links are suppressed with [IndexOptions](#) directive's SuppressColumnSorting option.

Note that when the display is sorted by "Size", it's the *actual* size of the files that's used, not the displayed value - so a 1010-byte file will always be displayed before a 1011-byte file (if in ascending order) even though they both are shown as "1K".



## Autoindex Request Query Arguments

Apache 2.0.23 reorganized the Query Arguments for Column Sorting, and introduced an entire group of new query options. To effectively eliminate all client control over the output, the [IndexOptions IgnoreClient](#) option was introduced.

The column sorting headers themselves are self-referencing hyperlinks that add the sort query options shown below. Any option below may be added to any request for the directory resource.

- C=N sorts the directory by file name
- C=M sorts the directory by last-modified date, then file name
- C=S sorts the directory by size, then file name
- C=D sorts the directory by description, then file name
  
- O=A sorts the listing in Ascending Order
- O=D sorts the listing in Descending Order
  
- F=0 formats the listing as a simple list (not FancyIndexed)
- F=1 formats the listing as a FancyIndexed list
- F=2 formats the listing as an HTMLTable FancyIndexed list
  
- V=0 disables version sorting
- V=1 enables version sorting
  
- P=*pattern* lists only files matching the given *pattern*

Note that the 'P'attern query argument is tested *after* the usual [IndexIgnore](#) directives are processed, and all file names are still subjected to the same criteria as any other autoindex listing. The Query Arguments parser in [mod\\_autoindex](#) will stop abruptly when an unrecognized option is encountered. The Query Arguments must be well formed, according to the table above.

The simple example below, which can be clipped and saved in a header.html file, illustrates these query options. Note that the unknown "X" argument, for the submit button, is listed last to assure the arguments are all parsed before mod\_autoindex encounters the X=Go input.

```
<form action="" method="get">
  Show me a <select name="F">
    <option value="0"> Plain list</option>
    <option value="1" selected="selected"> Fancy list</option>
    <option value="2"> Table list</option>
  </select>
  Sorted by <select name="C">
    <option value="N" selected="selected"> Name</option>
    <option value="M"> Date Modified</option>
    <option value="S"> Size</option>
    <option value="D"> Description</option>
  </select>
  <select name="O">
    <option value="A" selected="selected"> Ascending</option>
    <option value="D"> Descending</option>
  </select>
  <select name="V">
    <option value="0" selected="selected"> in Normal
    order</option>
    <option value="1"> in Version order</option>
  </select>
  Matching <input type="text" name="P" value="*" />
  <input type="submit" name="X" value="Go" />
</form>
```



<b>Description:</b>	Alternate text to display for a file, instead of an icon selected by filename
<b>Syntax:</b>	<code>AddAlt <i>string file</i> [<i>file</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

**AddAlt** provides the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *File* is a file extension, partial filename, wild-card expression or full filename for files to describe. If *String* contains any whitespace, you have to enclose it in quotes (" or '). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

### Examples

```
AddAlt "PDF file" *.pdf
AddAlt Compressed *.gz *.zip *.Z
```



<b>Description:</b>	Alternate text to display for a file instead of an icon selected by MIME-encoding
<b>Syntax:</b>	<code>AddAltByEncoding <i>string</i> <i>MIME-encoding</i> [<i>MIME-encoding</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

`AddAltByEncoding` provides the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *MIME-encoding* is a valid content-encoding, such as x-compress. If *String* contains any whitespace, you have to enclose it in quotes (" or '). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

### Example

```
AddAltByEncoding gzip x-gzip
```



## AddAltByType Directive

<b>Description:</b>	Alternate text to display for a file, instead of an icon selected by MIME content-type
<b>Syntax:</b>	<code>AddAltByType <i>string</i> <i>MIME-type</i> [<i>MIME-type</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

`AddAltByType` sets the alternate text to display for a file, instead of an icon, for [FancyIndexing](#). *MIME-type* is a valid content-type, such as `text/html`. If *String* contains any whitespace, you have to enclose it in quotes (" or '). This alternate text is displayed if the client is image-incapable, has image loading disabled, or fails to retrieve the icon.

### Example

```
AddAltByType 'plain text' text/plain
```



<b>Description:</b>	Description to display for a file
<b>Syntax:</b>	AddDescription <i>string file</i> [ <i>file</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

This sets the description to display for a file, for [FancyIndexing](#). *File* is a file extension, partial filename, wild-card expression or full filename for files to describe. *String* is enclosed in double quotes (").

### Example

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

The typical, default description field is 23 bytes wide. 6 more bytes are added by the [IndexOptions SuppressIcon](#) option, 7 bytes are added by the [IndexOptions SuppressSize](#) option, and 19 bytes are added by the [IndexOptions SuppressLastModified](#) option. Therefore, the widest default the description column is ever assigned is 55 bytes.

See the [DescriptionWidth](#) [IndexOptions](#) keyword for details on overriding the size of this column, or allowing descriptions of unlimited length.

### Caution

Descriptive text defined with [AddDescription](#) may contain HTML markup, such as tags and character entities. If the width of the description column should happen to truncate a tagged element (such as cutting off the end of a bolded phrase), the

results may affect the rest of the directory listing.



<b>Description:</b>	Icon to display for a file selected by name
<b>Syntax:</b>	AddIcon <i>icon name</i> [ <i>name</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

This sets the icon to display next to a file ending in *name* for [FancyIndexing](#). *Icon* is either a (%-escaped) relative URL to the icon, a fully qualified remote URL, or of the format (*alt text*, *url*) where *alt text* is the text tag given for an icon for non-graphical browsers.

*Name* is either ^^DIRECTORY^^ for directories, ^^BLANKICON^^ for blank lines (to format the list correctly), a file extension, a wildcard expression, a partial filename or a complete filename.

^^BLANKICON^^ is only used for formatting, and so is unnecessary if you're using `IndexOptions HTMLTable`.

### Examples

```
AddIcon (IMG,/icons/image.xbm) .gif .jpg .xbm
AddIcon /icons/dir.xbm ^^DIRECTORY^^
AddIcon /icons/backup.xbm *~
```

[AddIconByType](#) should be used in preference to [AddIcon](#), when possible.



<b>Description:</b>	Icon to display next to files selected by MIME content-encoding
<b>Syntax:</b>	<code>AddIconByEncoding <i>icon</i> <i>MIME-encoding</i> [<i>MIME-encoding</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

This sets the icon to display next to files with [FancyIndexing](#). *Icon* is either a (%-escaped) relative URL to the icon, a fully qualified remote URL, or of the format (*alttext*, *url*) where *alttext* is the text tag given for an icon for non-graphical browsers.

*MIME-encoding* is a wildcard expression matching required the content-encoding.

### Example

```
AddIconByEncoding /icons/compress.xbm x-compress
```



## AddIconByType Directive

<b>Description:</b>	Icon to display next to files selected by MIME content-type
<b>Syntax:</b>	AddIconByType <i>icon</i> <i>MIME-type</i> [ <i>MIME-type</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

This sets the icon to display next to files of type *MIME-type* for [FancyIndexing](#). *Icon* is either a (%-escaped) relative URL to the icon, a fully qualified remote URL, or of the format (*alt text, url*) where *alt text* is the text tag given for an icon for non-graphical browsers.

*MIME-type* is a wildcard expression matching required the mime types.

### Example

```
AddIconByType (IMG,/icons/image.xbm) image/*
```



<b>Description:</b>	Icon to display for files when no specific icon is configured
<b>Syntax:</b>	<code>DefaultIcon <i>url-path</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

The `DefaultIcon` directive sets the icon to display for files when no specific icon is known, for [FancyIndexing](#). *Url-path* is a (%-escaped) relative URL to the icon, or a fully qualified remote URL.

### Example

```
DefaultIcon /icon/unknown.xbm
```



<b>Description:</b>	Name of the file that will be inserted at the top of the index listing
<b>Syntax:</b>	HeaderName <i>filename</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

The **HeaderName** directive sets the name of the file that will be inserted at the top of the index listing. *Filename* is the name of the file to include.

### Example

```
HeaderName HEADER.html
```

Both **HeaderName** and **ReadmeName** now treat *Filename* as a URI path relative to the one used to access the directory being indexed. If *Filename* begins with a slash, it will be taken to be relative to the **DocumentRoot**.

### Example

```
HeaderName /include/HEADER.html
```

*Filename* must resolve to a document with a major content type of `text/*` (e.g., `text/html`, `text/plain`, etc.). This means that *filename* may refer to a CGI script if the script's actual file type (as opposed to its output) is marked as `text/html` such as with a directive like:

```
AddType text/html .cgi
```

[Content negotiation](#) will be performed if [Options MultiViews](#) is in effect. If *filename* resolves to a static text/html document (not a CGI script) and either one of the [options Includes](#) or [IncludesNOEXEC](#) is enabled, the file will be processed for server-side includes (see the [mod\\_include](#) documentation).

If the file specified by [HeaderName](#) contains the beginnings of an HTML document (<html>, <head>, etc.) then you will probably want to set [IndexOptions +SuppressHTMLPreamble](#), so that these tags are not repeated.



## IndexIgnore Directive

<b>Description:</b>	Adds to the list of files to hide when listing a directory
<b>Syntax:</b>	IndexIgnore <i>file</i> [ <i>file</i> ] ...
<b>Default:</b>	IndexIgnore "."
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

The **IndexIgnore** directive adds to the list of files to hide when listing a directory. *File* is a shell-style wildcard expression or full filename. Multiple IndexIgnore directives add to the list, rather than replacing the list of ignored files. By default, the list contains . (the current directory).

```
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
```



<b>Description:</b>	Various configuration settings for directory indexing
<b>Syntax:</b>	<code>IndexOptions [+ -]option [[+ -]option] ...</code>
<b>Default:</b>	By default, no options are enabled.
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

The `IndexOptions` directive specifies the behavior of the directory indexing. *Option* can be one of

#### **Charset=*character-set* (Apache 2.0.61 and later)**

The `Charset` keyword allows you to specify the character set of the generated page. The default is either *ISO-8859-1* or *UTF-8*, depending on whether the underlying file system is unicode or not.

##### **Example:**

```
IndexOptions Charset=UTF-8
```

#### **Type=*MIME content-type* (Apache 2.0.61 and later)**

The `Type` keyword allows you to specify the MIME content-type of the generated page. The default is *text/html*.

##### **Example:**

```
IndexOptions Type=text/plain
```

#### **DescriptionWidth=[*n* | \*] (Apache 2.0.23 and later)**

The `DescriptionWidth` keyword allows you to specify the width of the description column in characters.

-DescriptionWidth (or unset) allows [mod\\_autoindex](#) to calculate the best width.

DescriptionWidth=*n* fixes the column width to *n* bytes wide.

DescriptionWidth=\* grows the column to the width necessary to accommodate the longest description string.

**See the section on [AddDescription](#) for dangers inherent in truncating descriptions.**

### **FancyIndexing**

This turns on fancy indexing of directories.

### **FoldersFirst (*Apache 2.0.23 and later*)**

If this option is enabled, subdirectory listings will *always* appear first, followed by normal files in the directory. The listing is basically broken into two components, the files and the subdirectories, and each is sorted separately and then displayed subdirectories-first. For instance, if the sort order is descending by name, and FoldersFirst is enabled, subdirectory Zed will be listed before subdirectory Beta, which will be listed before normal files Gamma and Alpha.

**This option only has an effect if [FancyIndexing](#) is also enabled.**

### **HTMLTable (*Experimental, Apache 2.0.23 and later*)**

This experimental option with FancyIndexing constructs a simple table for the fancy directory listing. Note this will confuse older browsers. It is particularly necessary if file names or description text will alternate between left-to-right and right-to-left reading order, as can happen on WinNT or other utf-8 enabled platforms.

### **IconsAreLinks**

This makes the icons part of the anchor for the filename, for fancy indexing.

### **IconHeight[=*pixels*]**

Presence of this option, when used with `IconWidth`, will cause the server to include `height` and `width` attributes in the `img` tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard height of the icons supplied with the Apache software.

### **IconWidth[=*pixels*]**

Presence of this option, when used with `IconHeight`, will cause the server to include `height` and `width` attributes in the `img` tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard width of the icons supplied with the Apache software.

### **IgnoreCase**

If this option is enabled, names are sorted in a case-insensitive manner. For instance, if the sort order is ascending by name, and `IgnoreCase` is enabled, file Zeta will be listed after file alfa (Note: file GAMMA will always be listed before file gamma).

### **IgnoreClient**

This option causes `mod_autoindex` to ignore all query variables from the client, including sort order (implies [SuppressColumnSorting](#).)

### **NameWidth=[*n* | \*]**

The `NameWidth` keyword allows you to specify the width of the filename column in bytes.

-`NameWidth` (or `unset`) allows `mod_autoindex` to calculate the best width.

`NameWidth=n` fixes the column width to *n* bytes wide.

`NameWidth=*` grows the column to the necessary width.

## ScanHTMLTitles

This enables the extraction of the title from HTML documents for fancy indexing. If the file does not have a description given by [AddDescription](#) then httpd will read the document for the value of the `title` element. This is CPU and disk intensive.

## SuppressColumnSorting

If specified, Apache will not make the column headings in a FancyIndexed directory listing into links for sorting. The default behavior is for them to be links; selecting the column heading will sort the directory listing by the values in that column. **Prior to Apache 2.0.23, this also disabled parsing the Query Arguments for the sort string.** That behavior is now controlled by [IndexOptions IgnoreClient](#) in Apache 2.0.23.

## SuppressDescription

This will suppress the file description in fancy indexing listings. By default, no file descriptions are defined, and so the use of this option will regain 23 characters of screen space to use for something else. See [AddDescription](#) for information about setting the file description. See also the [DescriptionWidth](#) index option to limit the size of the description column.

## SuppressHTMLPreamble

If the directory actually contains a file specified by the [HeaderName](#) directive, the module usually includes the contents of the file after a standard HTML preamble (`<html>`, `<head>`, *et cetera*). The `SuppressHTMLPreamble` option disables this behaviour, causing the module to start the display with the header file contents. The header file must contain appropriate HTML instructions in this case. If there is no header file, the preamble is generated as usual.

### **SuppressIcon (*Apache 2.0.23 and later*)**

This will suppress the icon in fancy indexing listings.

Combining both SuppressIcon and SuppressRules yields proper HTML 3.2 output, which by the final specification prohibits `img` and `hr` elements from the `pre` block (used to format FancyIndexed listings.)

### **SuppressLastModified**

This will suppress the display of the last modification date, in fancy indexing listings.

### **SuppressRules (*Apache 2.0.23 and later*)**

This will suppress the horizontal rule lines (`hr` elements) in directory listings. Combining both SuppressIcon and SuppressRules yields proper HTML 3.2 output, which by the final specification prohibits `img` and `hr` elements from the `pre` block (used to format FancyIndexed listings.)

### **SuppressSize**

This will suppress the file size in fancy indexing listings.

### **TrackModified (*Apache 2.0.23 and later*)**

This returns the Last-Modified and ETag values for the listed directory in the HTTP header. It is only valid if the operating system and file system return appropriate `stat()` results. Some Unix systems do so, as do OS2's JFS and Win32's NTFS volumes. OS2 and Win32 FAT volumes, for example, do not. Once this feature is enabled, the client or proxy can track changes to the list of files when they perform a HEAD request. Note some operating systems correctly track new and removed files, but do not track changes for sizes or dates of the files within the directory. **Changes to the size or date stamp of an existing file will not update the Last-Modified header on all Unix platforms.** If this is a concern, leave this option disabled.

## VersionSort (*Apache 2.0a3 and later*)

The `VersionSort` keyword causes files containing version numbers to sort in a natural way. Strings are sorted as usual, except that substrings of digits in the name and description are compared according to their numeric value.

### Example:

```
foo-1.7
foo-1.7.2
foo-1.7.12
foo-1.8.2
foo-1.8.2a
foo-1.12
```

If the number starts with a zero, then it is considered to be a fraction:

```
foo-1.001
foo-1.002
foo-1.030
foo-1.04
```

## XHTML (*Apache 2.0.49 and later*)

The `XHTML` keyword forces `mod_autoindex` to emit XHTML 1.0 code instead of HTML 3.2.

## Incremental IndexOptions

Apache 1.3.3 introduced some significant changes in the handling of `IndexOptions` directives. In particular:

- Multiple `IndexOptions` directives for a single directory are now merged together. The result of:

```
<Directory /foo>
  IndexOptions HTMLTable
  IndexOptions SuppressColumnsorting
</Directory>
```

will be the equivalent of

```
IndexOptions HTMLTable SuppressColumnsorting
```

- The addition of the incremental syntax (*i.e.*, prefixing keywords with + or -).

Whenever a '+' or '-' prefixed keyword is encountered, it is applied to the current `IndexOptions` settings (which may have been inherited from an upper-level directory). However, whenever an unprefixed keyword is processed, it clears all inherited options and any incremental settings encountered so far. Consider the following example:

```
IndexOptions +ScanHTMLTitles -IconsAreLinks FancyIndexing  
IndexOptions +SuppressSize
```

The net effect is equivalent to `IndexOptions FancyIndexing +SuppressSize`, because the unprefixed `FancyIndexing` discarded the incremental keywords before it, but allowed them to start accumulating again afterward.

To unconditionally set the `IndexOptions` for a particular directory, clearing the inherited settings, specify keywords without any + or - prefixes.



<b>Description:</b>	Sets the default ordering of the directory index
<b>Syntax:</b>	IndexOrderDefault Ascending Descending Name Date Size Description
<b>Default:</b>	IndexOrderDefault Ascending Name
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

The `IndexOrderDefault` directive is used in combination with the `FancyIndexing` index option. By default, fancyindexed directory listings are displayed in ascending order by filename; the `IndexOrderDefault` allows you to change this initial display order.

`IndexOrderDefault` takes two arguments. The first must be either `Ascending` or `Descending`, indicating the direction of the sort. The second argument must be one of the keywords `Name`, `Date`, `Size`, or `Description`, and identifies the primary key. The secondary key is *always* the ascending filename.

You can, if desired, prevent the client from reordering the list by also adding the `SuppressColumnSorting` index option to remove the sort link from the top of the column, along with the `IgnoreClient` index option to prevent them from manually adding sort options to the query string in order to override your ordering preferences.



<b>Description:</b>	Name of the file that will be inserted at the end of the index listing
<b>Syntax:</b>	ReadmeName <i>filename</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_autoindex

The **ReadmeName** directive sets the name of the file that will be appended to the end of the index listing. *Filename* is the name of the file to include, and is taken to be relative to the location being indexed. If *Filename* begins with a slash, it will be taken to be relative to the **DocumentRoot**.

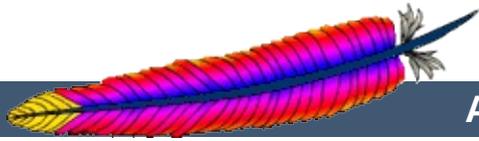
### Example

```
ReadmeName FOOTER.html
```

### Example 2

```
ReadmeName /include/FOOTER.html
```

See also **HeaderName**, where this behavior is described in greater detail.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_cache`

<b>Description:</b>	Content cache keyed to URIs.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>cache_module</code>
<b>Source File:</b>	<code>mod_cache.c</code>

## Summary

This module is experimental. Documentation is still under development...

`mod_cache` implements an [RFC 2616](#) compliant HTTP content cache that can be used to cache either local or proxied content. `mod_cache` requires the services of one or more storage management modules. Two storage management modules are included in the base Apache distribution:

### `mod_disk_cache`

implements a disk based storage manager.

### `mod_mem_cache`

implements a memory based storage manager. `mod_mem_cache` can be configured to operate in two modes: caching open file descriptors or caching objects in heap storage. `mod_mem_cache` can be used to cache locally generated content or to cache backend server content for `mod_proxy` when configured using [ProxyPass](#) (aka *reverse proxy*)

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.



Related Modules	Related Directives
<a href="#">mod disk cache</a>	<a href="#">CacheRoot</a>
<a href="#">mod mem cache</a>	<a href="#">CacheSize</a>
	<a href="#">CacheGcInterval</a>
	<a href="#">CacheDirLevels</a>
	<a href="#">CacheDirLength</a>
	<a href="#">CacheExpiryCheck</a>
	<a href="#">CacheMinFileSize</a>
	<a href="#">CacheMaxFileSize</a>
	<a href="#">CacheTimeMargin</a>
	<a href="#">CacheGcDaily</a>
	<a href="#">CacheGcUnused</a>
	<a href="#">CacheGcClean</a>
	<a href="#">CacheGcMemUsage</a>
	<a href="#">MCacheSize</a>
	<a href="#">MCacheMaxObjectCount</a>
	<a href="#">MCacheMinObjectSize</a>
	<a href="#">MCacheMaxObjectSize</a>
	<a href="#">MCacheRemovalAlgorithm</a>
	<a href="#">MCacheMaxStreamingBuffer</a>



## Sample httpd.conf

```
#
# Sample Cache Configuration
#
LoadModule cache_module modules/mod_cache.so

<IfModule mod_cache.c>
  #LoadModule disk_cache_module modules/mod_disk_cache.so
  <IfModule mod_disk_cache.c>
    CacheRoot c:/cacheroot
    CacheSize 256
    CacheEnable disk /
    CacheDirLevels 5
    CacheDirLength 3
  </IfModule>

  LoadModule mem_cache_module modules/mod_mem_cache.so
  <IfModule mod_mem_cache.c>
    CacheEnable mem /
    MCacheSize 4096
    MCacheMaxObjectCount 100
    MCacheMinObjectSize 1
    MCacheMaxObjectSize 2048
  </IfModule>
</IfModule>
```



## CacheDefaultExpire Directive

<b>Description:</b>	The default duration to cache a document when no expiry date is specified.
<b>Syntax:</b>	CacheDefaultExpire <i>seconds</i>
<b>Default:</b>	CacheDefaultExpire 3600 (one hour)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

The `CacheDefaultExpire` directive specifies a default time, in seconds, to cache a document if neither an expiry date nor last-modified date are provided with the document. The value specified with the `CacheMaxExpire` directive does *not* override this setting.

```
CacheDefaultExpire 86400
```



## CacheDisable Directive

<b>Description:</b>	Disable caching of specified URLs
<b>Syntax:</b>	CacheDisable <i>url-string</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

The `CacheDisable` directive instructs `mod_cache` to *not* cache urls at or below *url-string*.

### Example

```
CacheDisable /local_files
```



<b>Description:</b>	Enable caching of specified URLs using a specified storage manager
<b>Syntax:</b>	<code>CacheEnable <i>cache_type url-string</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_cache</code>

The `CacheEnable` directive instructs `mod_cache` to cache urls at or below `url-string`. The cache storage manager is specified with the `cache_type` argument. `cache_type mem` instructs `mod_cache` to use the memory based storage manager implemented by `mod_mem_cache`. `cache_type disk` instructs `mod_cache` to use the disk based storage manager implemented by `mod_disk_cache`. `cache_type fd` instructs `mod_cache` to use the file descriptor cache implemented by `mod_mem_cache`.

In the event that the URL space overlaps between different `CacheEnable` directives (as in the example below), each possible storage manager will be run until the first one that actually processes the request. The order in which the storage managers are run is determined by the order of the `CacheEnable` directives in the configuration file.

```
CacheEnable mem /manual
CacheEnable fd /images
CacheEnable disk /
```



**Description:** Percentage of document served, after which the server will complete caching the file even if the request is cancelled.

**Syntax:** `CacheForceCompletion Percentage`

**Default:** `CacheForceCompletion 60`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_cache`

Ordinarily, if a request is cancelled while the response is being cached and delivered to the client the processing of the response will stop and the cache entry will be removed. The `CacheForceCompletion` directive specifies a threshold beyond which the document will continue to be cached to completion, even if the request is cancelled.

The threshold is a percentage specified as a value between 1 and 100. A value of 0 specifies that the default be used. A value of 100 will only cache documents that are served in their entirety. A value between 60 and 90 is recommended.

```
CacheForceCompletion 80
```

**Note:**

This feature is currently *not* implemented.



<b>Description:</b>	Ignore the fact that the client requested the content not be cached.
<b>Syntax:</b>	CacheIgnoreCacheControl On Off
<b>Default:</b>	CacheIgnoreCacheControl Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

Ordinarily, documents with no-cache or no-store header values will not be stored in the cache. The `CacheIgnoreCacheControl` directive allows this behavior to be overridden.

`CacheIgnoreCacheControl On` tells the server to attempt to cache the document even if it contains no-cache or no-store header values. Documents requiring authorization will *never* be cached.

```
CacheIgnoreCacheControl On
```



## CacheIgnoreHeaders Directive

<b>Description:</b>	Do not store the given HTTP header(s) in the cache.
<b>Syntax:</b>	CacheIgnoreHeaders <i>header-string</i> [ <i>header-string</i> ] ...
<b>Default:</b>	CacheIgnoreHeaders None
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

According to RFC 2616, hop-by-hop HTTP headers are not stored in the cache. The following HTTP headers are hop-by-hop headers and thus do not get stored in the cache in *any* case regardless of the setting of **CacheIgnoreHeaders**:

- Connection
- Keep-Alive
- Proxy-Authenticate
- Proxy-Authorization
- TE
- Trailers
- Transfer-Encoding
- Upgrade

**CacheIgnoreHeaders** specifies additional HTTP headers that should not to be stored in the cache. For example, it makes sense in some cases to prevent cookies from being stored in the cache.

**CacheIgnoreHeaders** takes a space separated list of HTTP headers that should not be stored in the cache. If only hop-by-hop headers not should be stored in the cache (the RFC 2616 compliant behaviour), **CacheIgnoreHeaders** can be set to None.

## Example 1

CacheIgnoreHeaders Set-Cookie

## Example 2

CacheIgnoreHeaders None

### Warning:

If headers like Expires which are needed for proper cache management are not stored due to a `CacheIgnoreHeaders` setting, the behaviour of `mod_cache` is undefined.



## CacheIgnoreNoLastMod Directive

<b>Description:</b>	Ignore the fact that a response has no Last Modified header.
<b>Syntax:</b>	CacheIgnoreNoLastMod On Off
<b>Default:</b>	CacheIgnoreNoLastMod Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

Ordinarily, documents without a last-modified date are not cached. Under some circumstances the last-modified date is removed (during `mod_include` processing for example) or not provided at all. The `CacheIgnoreNoLastMod` directive provides a way to specify that documents without last-modified dates should be considered for caching, even without a last-modified date. If neither a last-modified date nor an expiry date are provided with the document then the value specified by the `CacheDefaultExpire` directive will be used to generate an expiration date.

```
CacheIgnoreNoLastMod On
```



**Description:** The factor used to compute an expiry date based on the LastModified date.

**Syntax:** CacheLastModifiedFactor *float*

**Default:** CacheLastModifiedFactor 0.1

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_cache

In the event that a document does not provide an expiry date but does provide a last-modified date, an expiry date can be calculated based on the time since the document was last modified. The `CacheLastModifiedFactor` directive specifies a *factor* to be used in the generation of this expiry date according to the following formula:  $\text{expiry-period} = \text{time-since-last-modified-date} * \text{factor}$   $\text{expiry-date} = \text{current-date} + \text{expiry-period}$  For example, if the document was last modified 10 hours ago, and *factor* is 0.1 then the expiry-period will be set to  $10 * 0.1 = 1$  hour. If the current time was 3:00pm then the computed expiry-date would be 3:00pm + 1hour = 4:00pm. If the expiry-period would be longer than that set by `CacheMaxExpire`, then the latter takes precedence.

```
CacheLastModifiedFactor 0.5
```

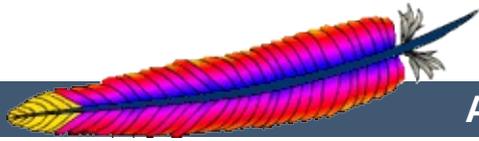


## CacheMaxExpire Directive

<b>Description:</b>	The maximum time in seconds to cache a document
<b>Syntax:</b>	CacheMaxExpire <i>seconds</i>
<b>Default:</b>	CacheMaxExpire 86400 (one day)
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_cache

The `CacheMaxExpire` directive specifies the maximum number of seconds for which cachable HTTP documents will be retained without checking the origin server. Thus, documents will be out of date at most this number of seconds. This maximum value is enforced even if an expiry date was supplied with the document.

```
CacheMaxExpire 604800
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_cern\_meta

<b>Description:</b>	CERN httpd metafile semantics
<b>Status:</b>	Extension
<b>Module Identifier:</b>	cern_meta_module
<b>Source File:</b>	mod_cern_meta.c

## Summary

Emulate the CERN HTTPD Meta file semantics. Meta files are HTTP headers that can be output in addition to the normal range of headers for each file accessed. They appear rather like the Apache .asis files, and are able to provide a crude way of influencing the Expires: header, as well as providing other curiosities. There are many ways to manage meta information, this one was chosen because there is already a large number of CERN users who can exploit this module.

More information on the [CERN metafile semantics](#) is available.

## See also

[mod\\_headers](#)

[mod\\_asis](#)



<b>Description:</b>	Name of the directory to find CERN-style meta information files
<b>Syntax:</b>	MetaDir <i>directory</i>
<b>Default:</b>	MetaDir .web
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_cern_meta

Specifies the name of the directory in which Apache can find meta information files. The directory is usually a 'hidden' subdirectory of the directory that contains the file being accessed. Set to "." to look in the same directory as the file:

```
MetaDir .
```

Or, to set it to a subdirectory of the directory containing the files:

```
MetaDir .meta
```



<b>Description:</b>	Activates CERN meta-file processing
<b>Syntax:</b>	MetaFiles on off
<b>Default:</b>	MetaFiles off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_cern_meta

Turns on/off Meta file processing on a per-directory basis.



<b>Description:</b>	File name suffix for the file containing CERN-style meta information
<b>Syntax:</b>	MetaSuffix <i>suffix</i>
<b>Default:</b>	MetaSuffix .meta
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_cern_meta

Specifies the file name suffix for the file containing the meta information. For example, the default values for the two directives will cause a request to `DOCUMENT_ROOT/somedir/index.html` to look in `DOCUMENT_ROOT/somedir/.web/index.html.meta` and will use its contents to generate additional MIME header information.

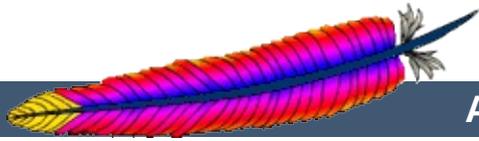
**Example:**

```
MetaSuffix .meta
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_cgi

<b>Description:</b>	Execution of CGI scripts
<b>Status:</b>	Base
<b>Module Identifier:</b>	cgi_module
<b>Source File:</b>	mod_cgi.c

## Summary

Any file that has the mime type `application/x-httpd-cgi` or handler `cgi-script` (Apache 1.1 or later) will be treated as a CGI script, and run by the server, with its output being returned to the client. Files acquire this type either by having a name containing an extension defined by the [AddType](#) directive, or by being in a [ScriptAlias](#) directory.

When the server invokes a CGI script, it will add a variable called `DOCUMENT_ROOT` to the environment. This variable will contain the value of the [DocumentRoot](#) configuration variable.

For an introduction to using CGI scripts with Apache, see our tutorial on [Dynamic Content With CGI](#).

When using a multi-threaded MPM under unix, the module [mod\\_cgid](#) should be used in place of this module. At the user level, the two modules are essentially identical.

## See also

[AcceptPathInfo](#)

[Options](#)

[ScriptAlias](#)

[AddHandler](#)

[Running CGI programs under different user IDs](#)  
[CGI Specification](#)



---

The server will set the CGI environment variables as described in the [CGI specification](#), with the following provisions:

### **PATH\_INFO**

This will not be available if the [AcceptPathInfo](#) directive is explicitly set to off. The default behavior, if [AcceptPathInfo](#) is not given, is that [mod\\_cgi](#) will accept path info (trailing /more/path/info following the script filename in the URI), while the core server will return a 404 NOT FOUND error for requests with additional path info. Omitting the [AcceptPathInfo](#) directive has the same effect as setting it On for [mod\\_cgi](#) requests.

### **REMOTE\_HOST**

This will only be set if [HostnameLookups](#) is set to on (it is off by default), and if a reverse DNS lookup of the accessing host's address indeed finds a host name.

### **REMOTE\_IDENT**

This will only be set if [IdentityCheck](#) is set to on and the accessing host supports the ident protocol. Note that the contents of this variable cannot be relied upon because it can easily be faked, and if there is a proxy between the client and the server, it is usually totally useless.

### **REMOTE\_USER**

This will only be set if the CGI script is subject to authentication.



Debugging CGI scripts has traditionally been difficult, mainly because it has not been possible to study the output (standard output and error) for scripts which are failing to run properly. These directives, included in Apache 1.2 and later, provide more detailed logging of errors when they occur.

## CGI Logfile Format

When configured, the CGI error log logs any CGI which does not execute properly. Each CGI script which fails to operate causes several lines of information to be logged. The first two lines are always of the format:

```
%% [time] request-line
%% HTTP-status CGI-script-filename
```

If the error is that CGI script cannot be run, the log file will contain an extra two lines:

```
%%error
error-message
```

Alternatively, if the error is the result of the script returning incorrect header information (often due to a bug in the script), the following information is logged:

```
%request
All HTTP request headers received
POST or PUT entity (if any)
%response
All headers output by the CGI script
%stdout
CGI standard output
%stderr
CGI standard error
```

(The %stdout and %stderr parts may be missing if the script did

not output anything on standard output or standard error).



<b>Description:</b>	Location of the CGI script error logfile
<b>Syntax:</b>	ScriptLog <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	<a href="#">mod_cgi</a> , <a href="#">mod_cgid</a>

The **ScriptLog** directive sets the CGI script error logfile. If no **ScriptLog** is given, no error log is created. If given, any CGI errors are logged into the filename given as argument. If this is a relative file or path it is taken relative to the **ServerRoot**.

**Example**

```
ScriptLog logs/cgi_log
```

This log will be opened as the user the child processes run as, *i.e.* the user specified in the main **User** directive. This means that either the directory the script log is in needs to be writable by that user or the file needs to be manually created and set to be writable by that user. If you place the script log in your main logs directory, do **NOT** change the directory permissions to make it writable by the user the child processes run as.

Note that script logging is meant to be a debugging feature when writing CGI scripts, and is not meant to be activated continuously on running servers. It is not optimized for speed or efficiency, and may have security problems if used in a manner other than that for which it was designed.



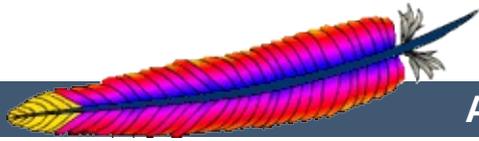
<b>Description:</b>	Maximum amount of PUT or POST requests that will be recorded in the scriptlog
<b>Syntax:</b>	ScriptLogBuffer <i>bytes</i>
<b>Default:</b>	ScriptLogBuffer 1024
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	<a href="#">mod_cgi</a> , <a href="#">mod_cgid</a>

The size of any PUT or POST entity body that is logged to the file is limited, to prevent the log file growing too big too quickly if large bodies are being received. By default, up to 1024 bytes are logged, but this can be changed with this directive.



<b>Description:</b>	Size limit of the CGI script logfile
<b>Syntax:</b>	ScriptLogLength <i>bytes</i>
<b>Default:</b>	ScriptLogLength 10385760
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	<a href="#">mod_cgi</a> , <a href="#">mod_cgid</a>

**ScriptLogLength** can be used to limit the size of the CGI script logfile. Since the logfile logs a lot of information per CGI error (all request headers, all script output) it can grow to be a big file. To prevent problems due to unbounded growth, this directive can be used to set an maximum file-size for the CGI logfile. If the file exceeds this size, no more information will be written to it.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module `mod_cgid`

<b>Description:</b>	Execution of CGI scripts using an external CGI daemon
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>cgid_module</code>
<b>Source File:</b>	<code>mod_cgid.c</code>
<b>Compatibility:</b>	Unix threaded MPMs only

### Summary

Except for the optimizations and the additional [ScriptSock](#) directive noted below, `mod_cgid` behaves similarly to `mod_cgi`. See the [mod\\_cgi](#) summary for additional details about Apache and CGI.

On certain unix operating systems, forking a process from a multi-threaded server is a very expensive operation because the new process will replicate all the threads of the parent process. In order to avoid incurring this expense on each CGI invocation, `mod_cgid` creates an external daemon that is responsible for forking child processes to run CGI scripts. The main server communicates with this daemon using a unix domain socket.

This module is used by default instead of `mod_cgi` whenever a multi-threaded MPM is selected during the compilation process. At the user level, this module is identical in configuration and operation to `mod_cgi`. The only exception is the additional directive `ScriptSock` which gives the name of the socket to use for communication with the `cgid` daemon.

### See also

[mod\\_cgi](#)

## Running CGI programs under different user IDs



<b>Description:</b>	The name of the socket to use for communication with the cgi daemon
<b>Syntax:</b>	ScriptSock <i>file-path</i>
<b>Default:</b>	ScriptSock logs/cgisock
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_cgid

This directive sets the name of the socket to use for communication with the CGI daemon. The socket will be opened using the permissions of the user who starts Apache (usually root). To maintain the security of communications with CGI scripts, it is important that no other user has permission to write in the directory where the socket is located.

### Example

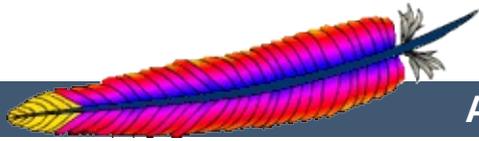
```
ScriptSock /var/run/cgid.sock
```

---

Copyright 2013 The Apache Software Foundation.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module `mod_charset_lite`

<b>Description:</b>	Specify character set translation or recoding
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>charset_lite_module</code>
<b>Source File:</b>	<code>mod_charset_lite.c</code>

### Summary

This is an **experimental** module and should be used with care. Experiment with your `mod_charset_lite` configuration to ensure that it performs the desired function.

`mod_charset_lite` allows the administrator to specify the source character set of objects as well as the character set they should be translated into before sending to the client. `mod_charset_lite` does not translate the data itself but instead tells Apache what translation to perform. `mod_charset_lite` is applicable to EBCDIC and ASCII host environments. In an EBCDIC environment, Apache normally translates text content from the code page of the Apache process locale to ISO-8859-1. `mod_charset_lite` can be used to specify that a different translation is to be performed. In an ASCII environment, Apache normally performs no translation, so `mod_charset_lite` is needed in order for any translation to take place.

This module provides a small subset of configuration mechanisms implemented by Russian Apache and its associated `mod_charset`.



## Invalid character set names

The character set name parameters of [CharsetSourceEnc](#) and [CharsetDefault](#) must be acceptable to the translation mechanism used by APR on the system where [mod\\_charset\\_lite](#) is deployed. These character set names are not standardized and are usually not the same as the corresponding values used in http headers. Currently, APR can only use `iconv(3)`, so you can easily test your character set names using the `iconv(1)` program, as follows:

```
iconv -f charsetsourceenc-value -t charsetdefault-value
```

## Mismatch between character set of content and translation rules

If the translation rules don't make sense for the content, translation can fail in various ways, including:

- The translation mechanism may return a bad return code, and the connection will be aborted.
- The translation mechanism may silently place special characters (e.g., question marks) in the output buffer when it cannot translate the input buffer.



<b>Description:</b>	Charset to translate into
<b>Syntax:</b>	CharsetDefault <i>charset</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Experimental
<b>Module:</b>	mod_charset_lite

The `CharsetDefault` directive specifies the charset that content in the associated container should be translated to.

The value of the *charset* argument must be accepted as a valid character set name by the character set support in APR. Generally, this means that it must be supported by iconv.

### Example

```
<Directory /export/home/trawick/apacheinst/htdocs/convert>  
  CharsetSourceEnc UTF-16BE  
  CharsetDefault ISO-8859-1  
</Directory>
```



<b>Description:</b>	Configures charset translation behavior
<b>Syntax:</b>	CharsetOptions <i>option</i> [ <i>option</i> ] ...
<b>Default:</b>	CharsetOptions DebugLevel=0 NoImplicitAdd
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Experimental
<b>Module:</b>	mod_charset_lite

The `CharsetOptions` directive configures certain behaviors of `mod_charset_lite`. *Option* can be one of

### **DebugLevel=*n***

The `DebugLevel` keyword allows you to specify the level of debug messages generated by `mod_charset_lite`. By default, no messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the `DBGLVL_` constants near the beginning of `mod_charset_lite.c`.

### **ImplicitAdd | NoImplicitAdd**

The `ImplicitAdd` keyword specifies that `mod_charset_lite` should implicitly insert its filter when the configuration specifies that the character set of content should be translated. If the filter chain is explicitly configured using the `AddOutputFilter` directive, `NoImplicitAdd` should be specified so that `mod_charset_lite` doesn't add its filter.



<b>Description:</b>	Source charset of files
<b>Syntax:</b>	CharsetSourceEnc <i>charset</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Experimental
<b>Module:</b>	mod_charset_lite

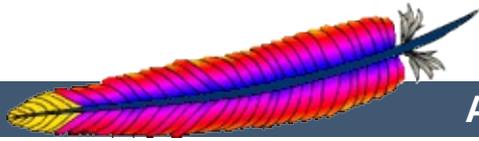
The **CharsetSourceEnc** directive specifies the source charset of files in the associated container.

The value of the *charset* argument must be accepted as a valid character set name by the character set support in APR. Generally, this means that it must be supported by iconv.

### Example

```
<Directory /export/home/trawick/apacheinst/htdocs/convert>  
  CharsetSourceEnc UTF-16BE  
  CharsetDefault ISO-8859-1  
</Directory>
```

The character set names in this example work with the iconv translation support in Solaris 8.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_dav

<b>Description:</b>	Distributed Authoring and Versioning ( <a href="#">WebDAV</a> ) functionality
<b>Status:</b>	Extension
<b>Module Identifier:</b>	dav_module
<b>Source File:</b>	mod_dav.c

## Summary

This module provides class 1 and class 2 [WebDAV](#) ('Web-based Distributed Authoring and Versioning') functionality for Apache. This extension to the HTTP protocol allows creating, moving, copying, and deleting resources and collections on a remote web server.

## See also

[DavLockDB](#)

[LimitXMLRequestBody](#)

[WebDAV Resources](#)



To enable `mod_dav`, add the following to a container in your `httpd.conf` file:

```
Dav On
```

This enables the DAV file system provider, which is implemented by the `mod_dav_fs` module. Therefore, that module must be compiled into the server or loaded at runtime using the `LoadModule` directive.

In addition, a location for the DAV lock database must be specified in the global section of your `httpd.conf` file using the `DavLockDB` directive:

```
DavLockDB /usr/local/apache2/var/DavLock
```

The directory containing the lock database file must be writable by the `User` and `Group` under which Apache is running.

You may wish to add a `<Limit>` clause inside the `<Location>` directive to limit access to DAV-enabled locations. If you want to set the maximum amount of bytes that a DAV client can send at one request, you have to use the `LimitXMLRequestBody` directive. The "normal" `LimitRequestBody` directive has no effect on DAV requests.

### Full Example

```
DavLockDB /usr/local/apache2/var/DavLock
```

```
<Location /foo>
```

```
  Dav On
```

```
  AuthType Basic
```

```
  AuthName DAV
```

```
  AuthUserFile user.passwd
```

```
<LimitExcept GET OPTIONS>
  require user admin
</LimitExcept>
</Location>
```

[mod\\_dav](#) is a descendent of Greg Stein's [mod\\_dav for Apache 1.3](#). More information about the module is available from that site.



Since DAV access methods allow remote clients to manipulate files on the server, you must take particular care to assure that your server is secure before enabling [mod\\_dav](#).

Any location on the server where DAV is enabled should be protected by authentication. The use of HTTP Basic Authentication is not recommended. You should use at least HTTP Digest Authentication, which is provided by the [mod\\_auth\\_digest](#) module. Nearly all WebDAV clients support this authentication method. An alternative is Basic Authentication over an [SSL](#) enabled connection.

In order for [mod\\_dav](#) to manage files, it must be able to write to the directories and files under its control using the [User](#) and [Group](#) under which Apache is running. New files created will also be owned by this [User](#) and [Group](#). For this reason, it is important to control access to this account. The DAV repository is considered private to Apache; modifying files outside of Apache (for example using FTP or filesystem-level tools) should not be allowed.

[mod\\_dav](#) may be subject to various kinds of denial-of-service attacks. The [LimitXMLRequestBody](#) directive can be used to limit the amount of memory consumed in parsing large DAV requests. The [DavDepthInfinity](#) directive can be used to prevent PROPFIND requests on a very large repository from consuming large amounts of memory. Another possible denial-of-service attack involves a client simply filling up all available disk space with many large files. There is no direct way to prevent this in Apache, so you should avoid giving DAV access to untrusted users.



## Complex Configurations

One common request is to use `mod_dav` to manipulate dynamic files (PHP scripts, CGI scripts, etc). This is difficult because a GET request will always run the script, rather than downloading its contents. One way to avoid this is to map two different URLs to the content, one of which will run the script, and one of which will allow it to be downloaded and manipulated with DAV.

```
Alias /phparea /home/gstein/php_files
Alias /php-source /home/gstein/php_files
<Location /php-source>
    DAV On
    ForceType text/plain
</Location>
```

With this setup, `http://example.com/phparea` can be used to access the output of the PHP scripts, and `http://example.com/php-source` can be used with a DAV client to manipulate them.



<b>Description:</b>	Enable WebDAV HTTP methods
<b>Syntax:</b>	Dav On Off  <i>provider-name</i>
<b>Default:</b>	Dav Off
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_dav

Use the **Dav** directive to enable the WebDAV HTTP methods for the given container:

```
<Location /foo>  
  Dav On  
</Location>
```

The value On is actually an alias for the default provider filesystem which is served by the **mod\_dav\_fs** module. Note, that once you have DAV enabled for some location, it *cannot* be disabled for sublocations. For a complete configuration example have a look at the [section above](#).

Do not enable WebDAV until you have secured your server. Otherwise everyone will be able to distribute files on your system.



<b>Description:</b>	Allow PROPFIND, Depth: Infinity requests
<b>Syntax:</b>	DavDepthInfinity on off
<b>Default:</b>	DavDepthInfinity off
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_dav

Use the `DavDepthInfinity` directive to allow the processing of PROPFIND requests containing the header 'Depth: Infinity'. Because this type of request could constitute a denial-of-service attack, by default it is not allowed.



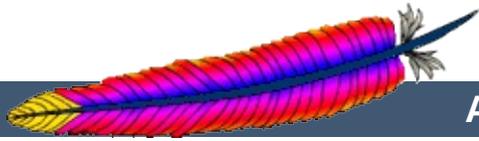
<b>Description:</b>	Minimum amount of time the server holds a lock on a DAV resource
<b>Syntax:</b>	DavMinTimeout <i>seconds</i>
<b>Default:</b>	DavMinTimeout 0
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_dav

When a client requests a DAV resource lock, it can also specify a time when the lock will be automatically removed by the server. This value is only a request, and the server can ignore it or inform the client of an arbitrary value.

Use the `DavMinTimeout` directive to specify, in seconds, the minimum lock timeout to return to a client. Microsoft Web Folders defaults to a timeout of 120 seconds; the `DavMinTimeout` can override this to a higher value (like 600 seconds) to reduce the chance of the client losing the lock due to network latency.

### Example

```
<Location /MSWord>
  DavMinTimeout 600
</Location>
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_dav_fs`

<b>Description:</b>	filesystem provider for <code>mod_dav</code>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>dav_fs_module</code>
<b>Source File:</b>	<code>mod_dav_fs.c</code>

## Summary

This module *requires* the service of `mod_dav`. It acts as a support module for `mod_dav` and provides access to resources located in the server's file system. The formal name of this provider is `filesystem`. `mod_dav` backend providers will be invoked by using the `Dav` directive:

### Example

```
Dav filesystem
```

Since `filesystem` is the default provider for `mod_dav`, you may simply use the value `On` instead.

## See also

[mod\\_dav](#)



<b>Description:</b>	Location of the DAV lock database
<b>Syntax:</b>	DavLockDB <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_dav_fs

Use the [DavLockDB](#) directive to specify the full path to the lock database, excluding an extension. If the path is not absolute, it will be taken relative to [ServerRoot](#). The implementation of [mod\\_dav\\_fs](#) uses a SDBM database to track user locks.

### Example

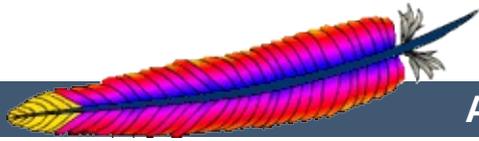
```
DavLockDB var/DavLock
```

The directory containing the lock database file must be writable by the [User](#) and [Group](#) under which Apache is running. For security reasons, you should create a directory for this purpose rather than changing the permissions on an existing directory. In the above example, Apache will create files in the `var/` directory under the [ServerRoot](#) with the base filename `DavLock` and extension name chosen by the server.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_deflate

<b>Description:</b>	Compress content before it is delivered to the client
<b>Status:</b>	Extension
<b>Module Identifier:</b>	deflate_module
<b>Source File:</b>	mod_deflate.c

## Summary

The [mod\\_deflate](#) module provides the DEFLATE output filter that allows output from your server to be compressed before being sent to the client over the network.

## See also

[Filters](#)



This is a simple sample configuration for the impatient.

### Compress only a few types

```
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

The following configuration, while resulting in more compressed content, is also much more complicated. Do not use this unless you fully understand all the configuration details.

### Compress everything except images

```
<Location />
# Insert filter
SetOutputFilter DEFLATE

# Netscape 4.x has some problems...
BrowserMatch ^Mozilla/4 gzip-only-text/html

# Netscape 4.06-4.08 have some more problems
BrowserMatch ^Mozilla/4\.0[678] no-gzip

# MSIE masquerades as Netscape, but it is fine
# BrowserMatch \bMSIE !no-gzip !gzip-only-text/html

# NOTE: Due to a bug in mod_setenvif up to Apache 2.0.48
# the above regex won't work. You can use the following
# workaround to get the desired effect:
BrowserMatch \bMSI[E] !no-gzip !gzip-only-text/html

# Don't compress images
SetEnvIfNoCase Request_URI \
  \.(?:gif|jpe?g|png)$ no-gzip dont-vary

# Make sure proxies don't deliver the wrong content
Header append Vary User-Agent env=!dont-vary
</Location>
```



## Output Compression

Compression is implemented by the DEFLATE [filter](#). The following directive will enable compression for documents in the container where it is placed:

```
SetOutputFilter DEFLATE
```

Some popular browsers cannot handle compression of all content so you may want to set the `gzip-only-text/html` note to `1` to only allow html files to be compressed (see below). If you set this to *anything but 1* it will be ignored.

If you want to restrict the compression to particular MIME types in general, you may use the [AddOutputFilterByType](#) directive. Here is an example of enabling compression only for the html files of the Apache documentation:

```
<Directory "/your-server-root/manual">  
  AddOutputFilterByType DEFLATE text/html  
</Directory>
```

For browsers that have problems even with compression of all file types, use the [BrowserMatch](#) directive to set the `no-gzip` note for that particular browser so that no compression will be performed. You may combine `no-gzip` with `gzip-only-text/html` to get the best results. In that case the former overrides the latter. Take a look at the following excerpt from the [configuration example](#) defined in the section above:

```
BrowserMatch ^Mozilla/4 gzip-only-text/html  
BrowserMatch ^Mozilla/4\.0[678] no-gzip  
BrowserMatch \bMSIE !no-gzip !gzip-only-text/html
```

At first we probe for a User -Agent string that indicates a Netscape Navigator version of 4.x. These versions cannot handle compression of types other than text/html. The versions 4.06, 4.07 and 4.08 also have problems with decompressing html files. Thus, we completely turn off the deflate filter for them.

The third [BrowserMatch](#) directive fixes the guessed identity of the user agent, because the Microsoft Internet Explorer identifies itself also as "Mozilla/4" but is actually able to handle requested compression. Therefore we match against the additional string "MSIE" (\b means "word boundary") in the User -Agent Header and turn off the restrictions defined before.

### Note

The DEFLATE filter is always inserted after RESOURCE filters like PHP or SSI. It never touches internal subrequests.

## Input Decompression

The [mod\\_deflate](#) module also provides a filter for decompressing a gzip compressed request body . In order to activate this feature you have to insert the DEFLATE filter into the input filter chain using [SetInputFilter](#) or [AddInputFilter](#), for example:

```
<Location /dav-area>  
    SetInputFilter DEFLATE  
</Location>
```

Now if a request contains a Content -Encoding: gzip header, the body will be automatically decompressed. Few browsers have the ability to gzip request bodies. However, some special applications actually do support request compression, for instance some [WebDAV](#) clients.

### **Note on Content-Length**

If you evaluate the request body yourself, *don't trust the Content-Length header!* The Content-Length header reflects the length of the incoming data from the client and *not* the byte count of the decompressed data stream.



## Working with proxy servers

The `mod_deflate` module sends a `Vary: Accept-Encoding` HTTP response header to alert proxies that a cached response should be sent only to clients that send the appropriate `Accept-Encoding` request header. This prevents compressed content from being sent to a client that will not understand it.

If you use some special exclusions dependent on, for example, the `User-Agent` header, you must manually configure an addition to the `Vary` header to alert proxies of the additional restrictions. For example, in a typical configuration where the addition of the DEFLATE filter depends on the `User-Agent`, you should add:

```
Header append Vary User-Agent
```

If your decision about compression depends on other information than request headers (e.g. HTTP version), you have to set the `Vary` header to the value `*`. This prevents compliant proxies from caching entirely.

### Example

```
Header set Vary *
```



<b>Description:</b>	Fragment size to be compressed at one time by zlib
<b>Syntax:</b>	DeflateBufferSize <i>value</i>
<b>Default:</b>	DeflateBufferSize 8096
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_deflate

The `DeflateBufferSize` directive specifies the size in bytes of the fragments that zlib should compress at one time.



<b>Description:</b>	How much compression do we apply to the output
<b>Syntax:</b>	<code>DeflateCompressionLevel</code> <i>value</i>
<b>Default:</b>	Zlib's default
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_deflate
<b>Compatibility:</b>	This directive is available since Apache 2.0.45

The `DeflateCompressionLevel` directive specifies what level of compression should be used, the higher the value, the better the compression, but the more CPU time is required to achieve this.

The value must be between 1 (less compression) and 9 (more compression).



<b>Description:</b>	Places the compression ratio in a note for logging
<b>Syntax:</b>	<code>DeflateFilterNote [type] notename</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_deflate
<b>Compatibility:</b>	<i>type</i> is available since Apache 2.0.45

The `DeflateFilterNote` directive specifies that a note about compression ratios should be attached to the request. The name of the note is the value specified for the directive. You can use that note for statistical purposes by adding the value to your [access log](#).

### Example

```
DeflateFilterNote ratio

LogFormat "%r" %b (%{ratio}n) "%{User-agent}i" deflate
CustomLog logs/deflate_log deflate
```

If you want to extract more accurate values from your logs, you can use the *type* argument to specify the type of data left as note for logging. *type* can be one of:

### Input

Store the byte count of the filter's input stream in the note.

### Output

Store the byte count of the filter's output stream in the note.

### Ratio

Store the compression ratio ( $\text{output}/\text{input} * 100$ ) in the note. This is the default, if the *type* argument is omitted.

Thus you may log it this way:

## Accurate Logging

```
DeflateFilterNote Input instream  
DeflateFilterNote Output outstream  
DeflateFilterNote Ratio ratio
```

```
LogFormat "%r" %{outstream}n/%{instream}n (%{ratio}n%)'  
deflate  
CustomLog logs/deflate_log deflate
```

## See also

- [mod\\_log\\_config](#)



<b>Description:</b>	How much memory should be used by zlib for compression
<b>Syntax:</b>	<code>DeflateMemLevel</code> <i>value</i>
<b>Default:</b>	<code>DeflateMemLevel</code> 9
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_deflate</code>

The `DeflateMemLevel` directive specifies how much memory should be used by zlib for compression (a value between 1 and 9).



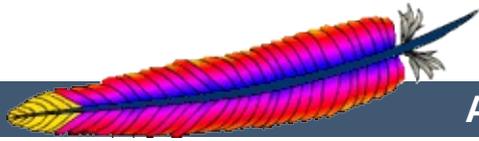
<b>Description:</b>	Zlib compression window size
<b>Syntax:</b>	<code>DeflateWindowSize</code> <i>value</i>
<b>Default:</b>	<code>DeflateWindowSize</code> 15
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_deflate</code>

The `DeflateWindowSize` directive specifies the zlib compression window size (a value between 1 and 15). Generally, the higher the window size, the higher can the compression ratio be expected.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module `mod_dir`

<b>Description:</b>	Provides for "trailing slash" redirects and serving directory index files
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>dir_module</code>
<b>Source File:</b>	<code>mod_dir.c</code>

### Summary

The index of a directory can come from one of two sources:

- A file written by the user, typically called `index.html`. The `DirectoryIndex` directive sets the name of this file. This is controlled by `mod_dir`.
- Otherwise, a listing generated by the server. This is provided by `mod_autoindex`.

The two functions are separated so that you can completely remove (or replace) automatic index generation should you want to.

A "trailing slash" redirect is issued when the server receives a request for a URL `http://servername/foo/dirname` where `dirname` is a directory. Directories require a trailing slash, so `mod_dir` issues a redirect to `http://servername/foo/dirname/`.



<b>Description:</b>	List of resources to look for when the client requests a directory
<b>Syntax:</b>	DirectoryIndex <i>local-url</i> [ <i>local-url</i> ] ...
<b>Default:</b>	DirectoryIndex index.html
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_dir

The **DirectoryIndex** directive sets the list of resources to look for, when the client requests an index of the directory by specifying a / at the end of the directory name. *Local-url* is the (%-encoded) URL of a document on the server relative to the requested directory; it is usually the name of a file in the directory. Several URLs may be given, in which case the server will return the first one that it finds. If none of the resources exist and the Indexes option is set, the server will generate its own listing of the directory.

### Example

```
DirectoryIndex index.html
```

then a request for `http://myserver/docs/` would return `http://myserver/docs/index.html` if it exists, or would list the directory if it did not.

Note that the documents do not need to be relative to the directory;

```
DirectoryIndex index.html index.txt /cgi-bin/index.pl
```

would cause the CGI script `/cgi-bin/index.pl` to be executed if neither `index.html` or `index.txt` existed in a directory.



<b>Description:</b>	Toggle trailing slash redirects on or off
<b>Syntax:</b>	DirectorySlash On Off
<b>Default:</b>	DirectorySlash On
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_dir
<b>Compatibility:</b>	Available in version 2.0.51 and later

The `DirectorySlash` directive determines, whether `mod_dir` should fixup URLs pointing to a directory or not.

Typically if a user requests a resource without a trailing slash, which points to a directory, `mod_dir` redirects him to the same resource, but *with* trailing slash for some good reasons:

- The user is finally requesting the canonical URL of the resource
- `mod_autoindex` works correctly. Since it doesn't emit the path in the link, it would point to the wrong path.
- `DirectoryIndex` will be evaluated *only* for directories requested with trailing slash.
- Relative URL references inside html pages will work correctly.

Well, if you don't want this effect *and* the reasons above don't apply to you, you can turn off the redirect with:

```
# see security warning below!  
<Location /some/path>  
    DirectorySlash Off  
    SetHandler some-handler  
</Location>
```

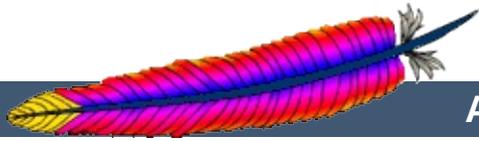
## Security Warning

Turning off the trailing slash redirect may result in an information disclosure. Consider a situation where `mod_autoindex` is active (`Options +Indexes`) and `DirectoryIndex` is set to a valid resource (say, `index.html`) and there's no other special handler defined for that URL. In this case a request with a trailing slash would show the `index.html` file. **But a request without trailing slash would list the directory contents.**

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_disk_cache`

<b>Description:</b>	Content cache storage manager keyed to URIs
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>disk_cache_module</code>
<b>Source File:</b>	<code>mod_disk_cache.c</code>

## Summary

This module is experimental. Documentation is still under development...

`mod_disk_cache` implements a disk based storage manager. It is primarily of use in conjunction with `mod_proxy`.

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.

### Note:

`mod_disk_cache` requires the services of `mod_cache`.



## CacheDirLength Directive

<b>Description:</b>	The number of characters in subdirectory names
<b>Syntax:</b>	CacheDirLength <i>length</i>
<b>Default:</b>	CacheDirLength 2
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

The `CacheDirLength` directive sets the number of characters for each subdirectory name in the cache hierarchy.

The result of `CacheDirLevels`\* `CacheDirLength` must not be higher than 20.

```
CacheDirLength 4
```



**Description:** The number of levels of subdirectories in the cache.

**Syntax:** `CacheDirLevels levels`

**Default:** `CacheDirLevels 3`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_disk_cache`

The `CacheDirLevels` directive sets the number of subdirectory levels in the cache. Cached data will be saved this many directory levels below the `CacheRoot` directory.

The result of `CacheDirLevels* CacheDirLength` must not be higher than 20.

```
CacheDirLevels 5
```



## CacheExpiryCheck Directive

<b>Description:</b>	Indicates if the cache observes Expires dates when seeking files
<b>Syntax:</b>	CacheExpiryCheck On Off
<b>Default:</b>	CacheExpiryCheck On
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

```
CacheExpiryCheck Off
```

The `CacheExpiryCheck` directive is currently *not* implemented.



**Description:** The time to retain unchanged cached files that match a URL

**Syntax:** `CacheGcClean hours url-string`

**Default:** `CacheGcClean ?`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_disk_cache`

More detail will be added here, when the function is implemented.

```
CacheGcClean 12 /daily_scripts
```

The `CacheGcClean` directive is currently *not* implemented.



## CacheGcDaily Directive

<b>Description:</b>	The recurring time each day for garbage collection to be run. (24 hour clock)
<b>Syntax:</b>	CacheGcDaily <i>time</i>
<b>Default:</b>	CacheGcDaily ?
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

```
CacheGcDaily 23:59
```

The `CacheGcDaily` directive is currently *not* implemented.



<b>Description:</b>	The interval between garbage collection attempts.
<b>Syntax:</b>	<code>CacheGcInterval</code> <i>hours</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_disk_cache</code>

The `CacheGcInterval` directive specifies the number of hours to wait between attempts to free up disk space.

More detail will be added here, when the function is implemented.

```
CacheGcInterval 24
```

The `CacheGcInterval` directive is currently *not* implemented.



## CacheGcMemUsage Directive

<b>Description:</b>	The maximum kilobytes of memory used for garbage collection
<b>Syntax:</b>	CacheGcMemUsage <i>KBytes</i>
<b>Default:</b>	CacheGcMemUsage ?
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

```
CacheGcMemUsage 16
```

The `CacheGcMemUsage` directive is currently *not* implemented.



**Description:** The time to retain unreferenced cached files that match a URL.

**Syntax:** `CacheGcUnused hours url-string`

**Default:** `CacheGcUnused ?`

**Context:** server config, virtual host

**Status:** Experimental

**Module:** `mod_disk_cache`

More detail will be added here, when the function is implemented.

```
CacheGcUnused 12 /local_images
```

The `CacheGcUnused` directive is currently *not* implemented.



<b>Description:</b>	The maximum size (in bytes) of a document to be placed in the cache
<b>Syntax:</b>	CacheMaxFileSize <i>bytes</i>
<b>Default:</b>	CacheMaxFileSize 1000000
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

The `CacheMaxFileSize` directive sets the maximum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMaxFileSize 64000
```



**Description:** The minimum size (in bytes) of a document to be placed in the cache

**Syntax:** CacheMinFileSize *bytes*

**Default:** CacheMinFileSize 1

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_disk\_cache

The `CacheMinFileSize` directive sets the minimum size, in bytes, for a document to be considered for storage in the cache.

```
CacheMinFileSize 64
```



**Description:** The directory root under which cache files are stored

**Syntax:** CacheRoot *directory*

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_disk\_cache

The **CacheRoot** directive defines the name of the directory on the disk to contain cache files. If the **mod\_disk\_cache** module has been loaded or compiled in to the Apache server, this directive *must* be defined. Failing to provide a value for **CacheRoot** will result in a configuration file processing error. The **CacheDirLevels** and **CacheDirLength** directives define the structure of the directories under the specified root directory.

```
CacheRoot c:/cacheroot
```



**Description:** The maximum amount of disk space that will be used by the cache in KBytes

**Syntax:** CacheSize *KBytes*

**Default:** CacheSize 1000000

**Context:** server config, virtual host

**Status:** Experimental

**Module:** mod\_disk\_cache

The `CacheSize` directive sets the desired disk space usage of the cache, in KBytes (1024-byte units). This directive does not put a hard limit on the size of the cache. The garbage collector will delete files until the usage is at or below the settings. Always use a value that is lower than the available disk space.

```
CacheSize 5000000
```



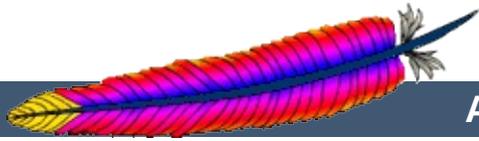
## CacheTimeMargin Directive

<b>Description:</b>	The minimum time margin to cache a document
<b>Syntax:</b>	CacheTimeMargin ?
<b>Default:</b>	CacheTimeMargin ?
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_disk_cache

More detail will be added here, when the function is implemented.

CacheTimeMargin X

The `CacheTimeMargin` directive is currently *not* implemented.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_dumpio

<b>Description:</b>	Dumps all I/O to error log as desired.
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	dumpio_module
<b>Source File:</b>	mod_dumpio.c

### Summary

mod\_dumpio allows for the logging of all input received by Apache and/or all output sent by Apache to be logged (dumped) to the error.log file.

The data logging is done right after SSL decoding (for input) and right before SSL encoding (for output). As can be expected, this can produce extreme volumes of data, and should only be used when debugging problems.



## Enabling sample support

To enable the module, it should be compiled and loaded in to your running Apache configuration. Logging can then be enabled or disabled via the below directives.

In order for dumping to work [LogLevel](#) must be set to debug.



## DumpIOInput Directive

<b>Description:</b>	Dump all input data to the error log
<b>Syntax:</b>	DumpIOInput On Off
<b>Default:</b>	DumpIOInput Off
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOInput is only available in Apache 2.0.53 and later.

Enable dumping of all input.

```
Example  
DumpIOInput On
```



## DumpIOOutput Directive

<b>Description:</b>	Dump all output data to the error log
<b>Syntax:</b>	DumpIOOutput On Off
<b>Default:</b>	DumpIOOutput Off
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_dumpio
<b>Compatibility:</b>	DumpIOOutput is only available in Apache 2.0.53 and later.

Enable dumping of all output.

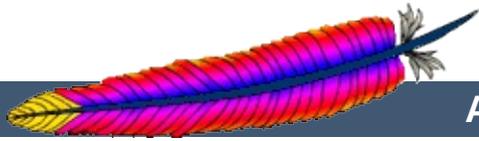
### Example

```
DumpIOOutput On
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_echo

<b>Description:</b>	A simple echo server to illustrate protocol modules
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	echo_module
<b>Source File:</b>	mod_echo.c
<b>Compatibility:</b>	Available in Apache 2.0 and later

### Summary

This module provides an example protocol module to illustrate the concept. It provides a simple echo server. Telnet to it and type stuff, and it will echo it.



<b>Description:</b>	Turn the echo server on or off
<b>Syntax:</b>	ProtocolEcho On Off
<b>Default:</b>	ProtocolEcho Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Experimental
<b>Module:</b>	mod_echo
<b>Compatibility:</b>	ProtocolEcho is only available in 2.0 and later.

The **ProtocolEcho** directive enables or disables the echo server.

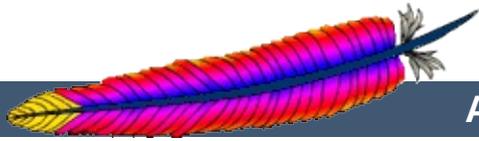
### Example

```
ProtocolEcho On
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_env

<b>Description:</b>	Modifies the environment which is passed to CGI scripts and SSI pages
<b>Status:</b>	Base
<b>Module Identifier:</b>	env_module
<b>Source File:</b>	mod_env.c

### Summary

This module allows for control of the environment that will be provided to CGI scripts and SSI pages. Environment variables may be passed from the shell which invoked the [httpd](#) process. Alternatively, environment variables may be set or unset within the configuration process.

### See also

[Environment Variables](#)



<b>Description:</b>	Passes environment variables from the shell
<b>Syntax:</b>	PassEnv <i>env-variable</i> [ <i>env-variable</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_env

Specifies one or more environment variables to pass to CGI scripts and SSI pages from the environment of the shell which invoked the [httpd](#) process.

### Example

```
PassEnv LD_LIBRARY_PATH
```



<b>Description:</b>	Sets environment variables
<b>Syntax:</b>	<code>SetEnv <i>env-variable</i> <i>value</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_env

Sets an environment variable, which is then passed on to CGI scripts and SSI pages.

### Example

```
SetEnv SPECIAL_PATH /foo/bin
```



<b>Description:</b>	Removes variables from the environment
<b>Syntax:</b>	UnsetEnv <i>env-variable</i> [ <i>env-variable</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_env

Removes one or more environment variables from those passed on to CGI scripts and SSI pages.

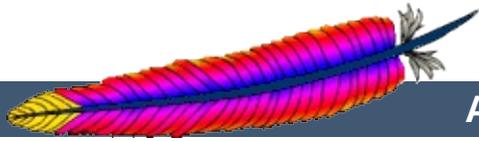
### Example

```
UnsetEnv LD_LIBRARY_PATH
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_example`

<b>Description:</b>	Illustrates the Apache module API
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>example_module</code>
<b>Source File:</b>	<code>mod_example.c</code>

## Summary

This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

The files in the `src/modules/example` directory under the Apache distribution directory tree are provided as an example to those that wish to write modules that use the Apache API.

The main file is `mod_example.c`, which illustrates all the different callback mechanisms and call syntaxes. By no means does an add-on module need to include routines for all of the callbacks - quite the contrary!

The example module is an actual working module. If you link it into your server, enable the "example-handler" handler for a location, and then browse to that location, you will see a display of some of the tracing the example module did as the various callbacks were made.



## Compiling the Example Module

To include the example module in your server, follow the steps below:

1. Uncomment the "AddModule modules/example/mod\_example" line near the bottom of the src/Configuration file. If there isn't one, add it; it should look like this:

```
AddModule modules/example/mod_example.o
```

2. Run the src/Configure script ("cd src; ./Configure"). This will build the Makefile for the server itself, and update the src/modules/Makefile for any additional modules you have requested from beneath that subdirectory.
3. Make the server (run "make" in the src directory).

To add another module of your own:

- A. `mkdir src/modules/mymodule`
- B. `cp src/modules/example/* src/modules/mymodule`
- C. Modify the files in the new directory.
- D. Follow steps [1] through [3] above, with appropriate changes.



To activate the example module, include a block similar to the following in your `srml.conf` file:

```
<Location /example-info>  
SetHandler example-handler  
</Location>
```

As an alternative, you can put the following into a `.htaccess` file and then request the file "test.example" from that location:

```
AddHandler example-handler .example
```

After reloading/restarting your server, you should be able to browse to this location and see the brief display mentioned earlier.



## Example Directive

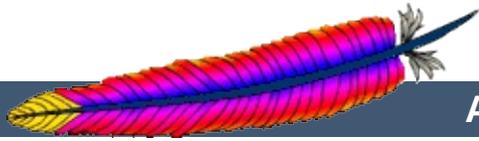
<b>Description:</b>	Demonstration directive to illustrate the Apache module API
<b>Syntax:</b>	Example
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Status:</b>	Experimental
<b>Module:</b>	mod_example

The **Example** directive just sets a demonstration flag which the example module's content handler displays. It takes no arguments. If you browse to an URL to which the example content-handler applies, you will get a display of the routines within the module and how and in what order they were called to service the document request. The effect of this directive one can observe under the point "Example directive declared here: YES/NO".

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_expires

<b>Description:</b>	Generation of Expires and Cache-Control HTTP headers according to user-specified criteria
<b>Status:</b>	Extension
<b>Module Identifier:</b>	expires_module
<b>Source File:</b>	mod_expires.c

### Summary

This module controls the setting of the Expires HTTP header and the max-age directive of the Cache-Control HTTP header in server responses. The expiration date can set to be relative to either the time the source file was last modified, or to the time of the client access.

These HTTP headers are an instruction to the client about the document's validity and persistence. If cached, the document may be fetched from the cache rather than from the source until this time has passed. After that, the cache copy is considered "expired" and invalid, and a new copy must be obtained from the source.

To modify Cache-Control directives other than max-age (see [RFC 2616 section 14.9](#)), you can use the [Header](#) directive.



The `ExpiresDefault` and `ExpiresByType` directives can also be defined in a more readable syntax of the form:

```
ExpiresDefault "<base> [plus] {<num> <type>}"  
ExpiresByType type/encoding "<base> [plus] {<num> <type>}"
```

where `<base>` is one of:

- access
- now (equivalent to 'access')
- modification

The `plus` keyword is optional. `<num>` should be an integer value [acceptable to `atoi()`], and `<type>` is one of:

- years
- months
- weeks
- days
- hours
- minutes
- seconds

For example, any of the following directives can be used to make documents expire 1 month after being accessed, by default:

```
ExpiresDefault "access plus 1 month"  
ExpiresDefault "access plus 4 weeks"  
ExpiresDefault "access plus 30 days"
```

The expiry time can be fine-tuned by adding several '`<num> <type>`' clauses:

```
ExpiresByType text/html "access plus 1 month 15 days 2 hours"  
ExpiresByType image/gif "modification plus 5 hours 3 minutes"
```

---

Note that if you use a modification date based setting, the Expires header will **not** be added to content that does not come from a file on disk. This is due to the fact that there is no modification time for such content.



## Expires Directive

<b>Description:</b>	Enables generation of Expires headers
<b>Syntax:</b>	ExpiresActive On Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_expires

This directive enables or disables the generation of the Expires and Cache-Control headers for the document realm in question. (That is, if found in an .htaccess file, for instance, it applies only to documents generated from that directory.) If set to Off, the headers will not be generated for any document in the realm (unless overridden at a lower level, such as an .htaccess file overriding a server config file). If set to On, the headers will be added to served documents according to the criteria defined by the ExpiresByType and ExpiresDefault directives (q.v.).

Note that this directive does not guarantee that an Expires or Cache-Control header will be generated. If the criteria aren't met, no header will be sent, and the effect will be as though this directive wasn't even specified.



## ExpiresByType Directive

<b>Description:</b>	Value of the Expires header configured by MIME type
<b>Syntax:</b>	ExpiresByType <i>MIME-type</i> <code>&lt;code&gt;</code> <i>seconds</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Extension
<b>Module:</b>	mod_expires

This directive defines the value of the Expires header and the max-age directive of the Cache-Control header generated for documents of the specified type (e.g., text/html). The second argument sets the number of seconds that will be added to a base time to construct the expiration date. The Cache-Control: max-age is calculated by subtracting the request time from the expiration date and expressing the result in seconds.

The base time is either the last modification time of the file, or the time of the client's access to the document. Which should be used is specified by the `<code>` field; M means that the file's last modification time should be used as the base time, and A means the client's access time should be used.

The difference in effect is subtle. If M is used, all current copies of the document in all caches will expire at the same time, which can be good for something like a weekly notice that's always found at the same URL. If A is used, the date of expiration is different for each client; this can be good for image files that don't change very often, particularly for a set of related documents that all refer to the same images (i.e., the images will be accessed repeatedly within a relatively short timespan).

### Example:

```
# enable expirations
ExpiresActive On
# expire GIF images after a month in the client's cache
ExpiresByType image/gif A2592000
# HTML documents are good for a week from the
# time they were changed
ExpiresByType text/html M604800
```

Note that this directive only has effect if `ExpiresActive On` has been specified. It overrides, for the specified MIME type *only*, any expiration date set by the [ExpiresDefault](#) directive.

You can also specify the expiration time calculation using an [alternate syntax](#), described earlier in this document.



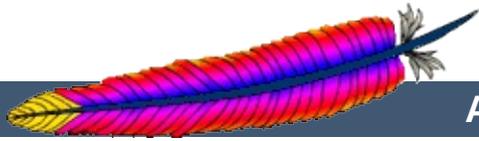
<b><u>Description:</u></b>	Default algorithm for calculating expiration time
<b><u>Syntax:</u></b>	ExpiresDefault <i>&lt;code&gt;seconds</i>
<b><u>Context:</u></b>	server config, virtual host, directory, .htaccess
<b><u>Override:</u></b>	Indexes
<b><u>Status:</u></b>	Extension
<b><u>Module:</u></b>	mod_expires

This directive sets the default algorithm for calculating the expiration time for all documents in the affected realm. It can be overridden on a type-by-type basis by the [ExpiresByType](#) directive. See the description of that directive for details about the syntax of the argument, and the [alternate syntax](#) description as well.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_ext_filter`

<b>Description:</b>	Pass the response body through an external program before delivery to the client
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>ext_filter_module</code>
<b>Source File:</b>	<code>mod_ext_filter.c</code>

## Summary

`mod_ext_filter` presents a simple and familiar programming model for [filters](#). With this module, a program which reads from stdin and writes to stdout (i.e., a Unix-style filter command) can be a filter for Apache. This filtering mechanism is much slower than using a filter which is specially written for the Apache API and runs inside of the Apache server process, but it does have the following benefits:

- the programming model is much simpler
- any programming/scripting language can be used, provided that it allows the program to read from standard input and write to standard output
- existing programs can be used unmodified as Apache filters

Even when the performance characteristics are not suitable for production use, `mod_ext_filter` can be used as a prototype environment for filters.

## See also

[Filters](#)



## Generating HTML from some other type of response

```
# mod_ext_filter directive to define a filter
# to HTML-ize text/c files using the external
# program /usr/bin/enscript, with the type of
# the result set to text/html
ExtFilterDefine c-to-html mode=output \
    intype=text/c outtype=text/html \
    cmd="/usr/bin/enscript --color -W html -Ec -o - -"

<Directory "/export/home/trawick/apacheinst/htdocs/c">
    # core directive to cause the new filter to
    # be run on output
    SetOutputFilter c-to-html

    # mod_mime directive to set the type of .c
    # files to text/c
    AddType text/c .c

    # mod_ext_filter directive to set the debug
    # level just high enough to see a log message
    # per request showing the configuration in force
    ExtFilterOptions DebugLevel=1
</Directory>
```

## Implementing a content encoding filter

Note: this gzip example is just for the purposes of illustration. Please refer to [mod\\_deflate](#) for a practical implementation.

```
# mod_ext_filter directive to define the external filter
ExtFilterDefine gzip mode=output cmd=/bin/gzip

<Location /gzipped>
    # core directive to cause the gzip filter to be
    # run on output
    SetOutputFilter gzip

    # mod_header directive to add
    # "Content-Encoding: gzip" header field
    Header set Content-Encoding gzip
</Location>
```

## Slowing down the server

```
# mod_ext_filter directive to define a filter
# which runs everything through cat; cat doesn't
# modify anything; it just introduces extra pathlength
# and consumes more resources
ExtFilterDefine slowdown mode=output cmd=/bin/cat \
    preservescontentlength

<Location />
    # core directive to cause the slowdown filter to
    # be run several times on output
    #
    SetOutputFilter slowdown;slowdown;slowdown
</Location>
```

## Using sed to replace text in the response

```
# mod_ext_filter directive to define a filter which
# replaces text in the response
#
ExtFilterDefine fixtext mode=output intype=text/html \
    cmd="/bin/sed s/verdana/arial/g"

<Location />
    # core directive to cause the fixtext filter to
    # be run on output
    SetOutputFilter fixtext
</Location>
```

## Tracing another filter

```
# Trace the data read and written by mod_deflate
# for a particular client (IP 192.168.1.31)
# experiencing compression problems.
# This filter will trace what goes into mod_deflate.
ExtFilterDefine tracebefore \
    cmd="/bin/tracefilter.pl /tmp/tracebefore" \
    EnableEnv=trace_this_client

# This filter will trace what goes after mod_deflate.
# Note that without the ftype parameter, the default
# filter type of AP_FTYPE_RESOURCE would cause the
# filter to be placed *before* mod_deflate in the filter
```

```
# chain. Giving it a numeric value slightly higher than
# AP_FTYPE_CONTENT_SET will ensure that it is placed
# after mod_deflate.
ExtFilterDefine traceafter \
    cmd="/bin/tracefilter.pl /tmp/traceafter" \
    EnableEnv=trace_this_client ftype=21

<Directory /usr/local/docs>
    SetEnvIf Remote_Addr 192.168.1.31 trace_this_client
    SetOutputFilter tracebefore;deflate;traceafter
</Directory>
```

### Here is the filter which traces the data:

```
#!/usr/local/bin/perl -w
use strict;

open(SAVE, ">$ARGV[0]")
    or die "can't open $ARGV[0]: $?";

while (<STDIN>) {
    print SAVE $_;
    print $_;
}

close(SAVE);
```



<b>Description:</b>	Define an external filter
<b>Syntax:</b>	<code>ExtFilterDefine <i>filtername</i> <i>parameters</i></code>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ext_filter</code>

The `ExtFilterDefine` directive defines the characteristics of an external filter, including the program to run and its arguments.

*filtername* specifies the name of the filter being defined. This name can then be used in `SetOutputFilter` directives. It must be unique among all registered filters. *At the present time, no error is reported by the register-filter API, so a problem with duplicate names isn't reported to the user.*

Subsequent parameters can appear in any order and define the external command to run and certain other characteristics. The only required parameter is `cmd=`. These parameters are:

#### **`cmd=cmdline`**

The `cmd=` keyword allows you to specify the external command to run. If there are arguments after the program name, the command line should be surrounded in quotation marks (e.g., `cmd="/bin/mypgm arg1 arg2"`). Normal shell quoting is not necessary since the program is run directly, bypassing the shell. Program arguments are blank-delimited. A backslash can be used to escape blanks which should be part of a program argument. Any backslashes which are part of the argument must be escaped with backslash themselves. In addition to the standard CGI environment variables, `DOCUMENT_URI`, `DOCUMENT_PATH_INFO`, and `QUERY_STRING_UNESCAPED` will also be set for the program.

**mode=*mode***

mode should be output for now (the default). In the future, mode=input will be used to specify a filter for request bodies.

**intype=*imt***

This parameter specifies the internet media type (*i.e.*, MIME type) of documents which should be filtered. By default, all documents are filtered. If intype= is specified, the filter will be disabled for documents of other types.

**outtype=*imt***

This parameter specifies the internet media type (*i.e.*, MIME type) of filtered documents. It is useful when the filter changes the internet media type as part of the filtering operation. By default, the internet media type is unchanged.

**PreservesContentLength**

The PreservesContentLength keyword specifies that the filter preserves the content length. This is not the default, as most filters change the content length. In the event that the filter doesn't modify the length, this keyword should be specified.

**f~~type~~=*filtertype***

This parameter specifies the numeric value for filter type that the filter should be registered as. The default value, AP\_FTYPE\_RESOURCE, is sufficient in most cases. If the filter needs to operate at a different point in the filter chain than resource filters, then this parameter will be necessary. See the AP\_FTYPE\_foo definitions in util\_filter.h for appropriate values.

**disableenv=*env***

This parameter specifies the name of an environment variable which, if set, will disable the filter.

**enableenv=*env***

This parameter specifies the name of an environment variable which must be set, or the filter will be disabled.



<b>Description:</b>	Configure <a href="#">mod_ext_filter</a> options
<b>Syntax:</b>	<code>ExtFilterOptions option [option] ...</code>
<b>Default:</b>	<code>ExtFilterOptions DebugLevel=0 NoLogStderr</code>
<b>Context:</b>	directory
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ext_filter</code>

The `ExtFilterOptions` directive specifies special processing options for [mod\\_ext\\_filter](#). *Option* can be one of

### **DebugLevel=*n***

The `DebugLevel` keyword allows you to specify the level of debug messages generated by [mod\\_ext\\_filter](#). By default, no debug messages are generated. This is equivalent to `DebugLevel=0`. With higher numbers, more debug messages are generated, and server performance will be degraded. The actual meanings of the numeric values are described with the definitions of the `DBGLVL_` constants near the beginning of `mod_ext_filter.c`.

Note: The core directive `LogLevel` should be used to cause debug messages to be stored in the Apache error log.

### **LogStderr | NoLogStderr**

The `LogStderr` keyword specifies that messages written to standard error by the external filter program will be saved in the Apache error log. `NoLogStderr` disables this feature.

### **Example**

```
ExtFilterOptions LogStderr DebugLevel=0
```

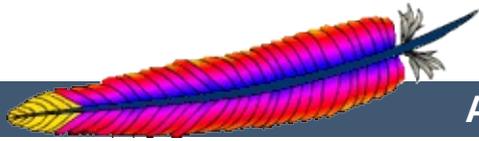
Messages written to the filter's standard error will be stored in the

Apache error log. No debug messages will be generated by [mod\\_ext\\_filter](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_file_cache`

<b>Description:</b>	Caches a static list of files in memory
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>file_cache_module</code>
<b>Source File:</b>	<code>mod_file_cache.c</code>

## Summary

This module should be used with care. You can easily create a broken site using `mod_file_cache`, so read this document carefully.

*Caching* frequently requested files that change very infrequently is a technique for reducing server load. `mod_file_cache` provides two techniques for caching frequently requested *static* files. Through configuration directives, you can direct `mod_file_cache` to either open then `mmap()` a file, or to pre-open a file and save the file's open *file handle*. Both techniques reduce server load when processing requests for these files by doing part of the work (specifically, the file I/O) for serving the file when the server is started rather than during each request.

Notice: You cannot use this for speeding up CGI programs or other files which are served by special content handlers. It can only be used for regular files which are usually served by the Apache core content handler.

This module is an extension of and borrows heavily from the `mod_mmap_static` module in Apache 1.3.



`mod_file_cache` caches a list of statically configured files via `MMapFile` or `CacheFile` directives in the main server configuration.

Not all platforms support both directives. For example, Apache on Windows does not currently support the `MMapStatic` directive, while other platforms, like AIX, support both. You will receive an error message in the server error log if you attempt to use an unsupported directive. If given an unsupported directive, the server will start but the file will not be cached. On platforms that support both directives, you should experiment with both to see which works best for you.

## MMapFile Directive

The `MMapFile` directive of `mod_file_cache` maps a list of statically configured files into memory through the system call `mmap()`. This system call is available on most modern Unix derivatives, but not on all. There are sometimes system-specific limits on the size and number of files that can be `mmap()`ed, experimentation is probably the easiest way to find out.

This `mmap()`ing is done once at server start or restart, only. So whenever one of the mapped files changes on the filesystem you *have* to restart the server (see the [Stopping and Restarting](#) documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` and `mv` do this. The reason why this module doesn't take care of changes to the files is that this check would need an extra `stat()` every time which is a waste and against the intent of I/O reduction.

## CacheFile Directive

The [CacheFile](#) directive of [mod\\_file\\_cache](#) opens an active *handle* or *file descriptor* to the file (or files) listed in the configuration directive and places these open file handles in the cache. When the file is requested, the server retrieves the handle from the cache and passes it to the `sendfile()` (or `TransmitFile()` on Windows), socket API.

This file handle caching is done once at server start or restart, only. So whenever one of the cached files changes on the filesystem you *have* to restart the server (see the [Stopping and Restarting](#) documentation). To reiterate that point: if the files are modified *in place* without restarting the server you may end up serving requests that are completely bogus. You should update files by unlinking the old copy and putting a new copy in place. Most tools such as `rdist` and `mv` do this.

### Note

Don't bother asking for a directive which recursively caches all the files in a directory. Try this instead... See the [Include](#) directive, and consider this command:

```
find /www/htdocs -type f -print \  
| sed -e 's/.*\/mmapfile &/' > /www/conf/mmap.conf
```



<b>Description:</b>	Cache a list of file handles at startup time
<b>Syntax:</b>	CacheFile <i>file-path</i> [ <i>file-path</i> ] ...
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_file_cache

The **CacheFile** directive opens handles to one or more files (given as whitespace separated arguments) and places these handles into the cache at server startup time. Handles to cached files are automatically closed on a server shutdown. When the files have changed on the filesystem, the server should be restarted to re-cache them.

Be careful with the *file-path* arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links *etc.* because that again would cost extra `stat()` system calls which is not acceptable. This module may or may not work with filenames rewritten by [mod\\_alias](#) or [mod\\_rewrite](#).

### Example

```
CacheFile /usr/local/apache/htdocs/index.html
```



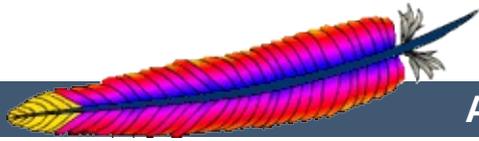
<b>Description:</b>	Map a list of files into memory at startup time
<b>Syntax:</b>	<code>MMapFile <i>file-path</i> [<i>file-path</i>] ...</code>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_file_cache</code>

The `MMapFile` directive maps one or more files (given as whitespace separated arguments) into memory at server startup time. They are automatically unmapped on a server shutdown. When the files have changed on the filesystem at least a HUP or USR1 signal should be send to the server to `re-mmap()` them.

Be careful with the *file-path* arguments: They have to literally match the filesystem path Apache's URL-to-filename translation handlers create. We cannot compare inodes or other stuff to match paths through symbolic links *etc.* because that again would cost extra `stat()` system calls which is not acceptable. This module may or may not work with filenames rewritten by [mod\\_alias](#) or [mod\\_rewrite](#).

### Example

```
MMapFile /usr/local/apache/htdocs/index.html
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_headers

<b>Description:</b>	Customization of HTTP request and response headers
<b>Status:</b>	Extension
<b>Module Identifier:</b>	headers_module
<b>Source File:</b>	mod_headers.c

## Summary

This module provides directives to control and modify HTTP request and response headers. Headers can be merged, replaced or removed.



The directives provided by `mod_headers` can occur almost anywhere within the server configuration. They are valid in the main server config and virtual host sections, inside `<Directory>`, `<Location>` and `<Files>` sections, and within `.htaccess` files.

The directives are processed in the following order:

1. main server
2. virtual host
3. `<Directory>` sections and `.htaccess`
4. `<Files>`
5. `<Location>`

Order is important. These two headers have a different effect if reversed:

```
RequestHeader append MirrorID "mirror 12"  
RequestHeader unset MirrorID
```

This way round, the `MirrorID` header is not set. If reversed, the `MirrorID` header is set to "mirror 12".



1. Copy all request headers that begin with "TS" to the response headers:

```
Header echo ^TS
```

2. Add a header, MyHeader, to the response including a timestamp for when the request was received and how long it took to begin serving the request. This header can be used by the client to intuit load on the server or in isolating bottlenecks between the client and the server.

```
Header add MyHeader "%D %t"
```

results in this header being added to the response:

```
MyHeader: D=3775428 t=991424704447256
```

3. Say hello to Joe

```
Header add MyHeader "Hello Joe. It took %D microseconds \\  
for Apache to serve this request."
```

results in this header being added to the response:

```
MyHeader: Hello Joe. It took D=3775428 microseconds for  
Apache to serve this request.
```

4. Conditionally send MyHeader on the response if and only if header "MyRequestHeader" is present on the request. This is useful for constructing headers in response to some client stimulus. Note that this example requires the services of the [mod\\_setenvif](#) module.

```
SetEnvIf MyRequestHeader value HAVE_MyRequestHeader
Header add MyHeader "%D %t mytext"
env=HAVE_MyRequestHeader
```

If the header `MyRequestHeader: value` is present on the HTTP request, the response will contain the following header:

```
MyHeader: D=3775428 t=991424704447256 mytext
```



<b>Description:</b>	Configure HTTP response headers
<b>Syntax:</b>	Header [ <i>condition</i> ] set   append   add   unset   echo <i>header</i> [ <i>value</i> ] [env=[!] <i>variable</i> ]
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_headers
<b>Compatibility:</b>	<i>Condition</i> is available in version 2.0.51 and later

This directive can replace, merge or remove HTTP response headers. The header is modified just after the content handler and output filters are run, allowing outgoing headers to be modified.

The optional *condition* can be either `onsuccess` or `always`. It determines, which internal header table should be operated on. `onsuccess` stands for 2xx status codes and `always` for all status codes (including 2xx). Especially if you want to unset headers set by certain modules, you should try out, which table is affected.

The action it performs is determined by the second argument. This can be one of the following values:

### **set**

The response header is set, replacing any previous header with this name. The *value* may be a format string.

### **append**

The response header is appended to any existing header of the same name. When a new value is merged onto an existing header it is separated from the existing header with a comma. This is the HTTP standard way of giving a header

multiple values.

### **add**

The response header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name. This can lead to unforeseen consequences, and in general "append" should be used instead.

### **unset**

The response header of this name is removed, if it exists. If there are multiple headers of the same name, all will be removed.

### **echo**

Request headers with this name are echoed back in the response headers. *header* may be a regular expression.

This argument is followed by a *header* name, which can include the final colon, but it is not required. Case is ignored for set, append, add and unset. The *header* name for echo is case sensitive and may be a regular expression.

For add, append and set a *value* is specified as the third argument. If *value* contains spaces, it should be surrounded by doublequotes. *value* may be a character string, a string containing format specifiers or a combination of both. The following format specifiers are supported in *value*:

%t	The time the request was received in Universal Coordinated Time since the epoch (Jan. 1, 1970) measured in microseconds. The value is preceded by t=.
%D	The time from when the request was received to the time the headers are sent on the wire. This is a measure of the duration of the request. The value is

	preceded by D=.
% {FOOBAR}e	The contents of the <a href="#">environment variable</a> FOOBAR.

When the **Header** directive is used with the add, append, or set argument, a fourth argument may be used to specify conditions under which the action will be taken. If the [environment variable](#) specified in the env= . . . argument exists (or if the environment variable does not exist and env=! . . . is specified) then the action specified by the **Header** directive will take effect. Otherwise, the directive will have no effect on the request.

The **Header** directives are processed just before the response is sent to the network. These means that it is possible to set and/or override most headers, except for those headers added by the header filter.



## RequestHeader Directive

<b>Description:</b>	Configure HTTP request headers
<b>Syntax:</b>	<code>RequestHeader set append add unset header [value [env=[!]variable]]</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_headers

This directive can replace, merge or remove HTTP request headers. The header is modified just before the content handler is run, allowing incoming headers to be modified. The action it performs is determined by the first argument. This can be one of the following values:

### **set**

The request header is set, replacing any previous header with this name

### **append**

The request header is appended to any existing header of the same name. When a new value is merged onto an existing header it is separated from the existing header with a comma. This is the HTTP standard way of giving a header multiple values.

### **add**

The request header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name. This can lead to unforeseen consequences, and in general append should be used instead.

### **unset**

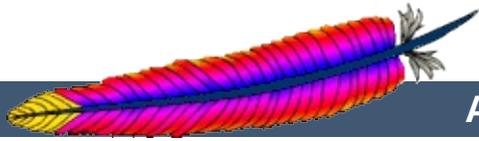
The request header of this name is removed, if it exists. If

there are multiple headers of the same name, all will be removed.

This argument is followed by a header name, which can include the final colon, but it is not required. Case is ignored. For add, append and set a *value* is given as the third argument. If *value* contains spaces, it should be surrounded by double quotes. For unset, no *value* should be given.

When the **RequestHeader** directive is used with the add, append, or set argument, a fourth argument may be used to specify conditions under which the action will be taken. If the [environment variable](#) specified in the env= . . . argument exists (or if the environment variable does not exist and env=! . . . is specified) then the action specified by the **RequestHeader** directive will take effect. Otherwise, the directive will have no effect on the request.

The **RequestHeader** directive is processed just before the request is run by its handler in the fixup phase. This should allow headers generated by the browser, or by Apache input filters to be overridden or modified.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_imap

<b>Description:</b>	Server-side imagemap processing
<b>Status:</b>	Base
<b>Module Identifier:</b>	imap_module
<b>Source File:</b>	mod_imap.c

## Summary

This module processes `.map` files, thereby replacing the functionality of the `imagemap` CGI program. Any directory or document type configured to use the handler `imap-file` (using either [AddHandler](#) or [SetHandler](#)) will be processed by this module.

The following directive will activate files ending with `.map` as imagemap files:

```
AddHandler imap-file map
```

Note that the following is still supported:

```
AddType application/x-httpd-imap map
```

However, we are trying to phase out "magic MIME types" so we are deprecating this method.



## NEW FEATURES

The imagemap module adds some new features that were not possible with previously distributed imagemap programs.

- URL references relative to the Referer: information.
- Default <base> assignment through a new map directive base.
- No need for `imagemap.conf` file.
- Point references.
- Configurable generation of imagemap menus.



The lines in the imagemap files can have one of several formats:

```
directive value [x,y ...]
directive value "Menu text" [x,y ...]
directive value x,y ... "Menu text"
```

The directive is one of `base`, `default`, `poly`, `circle`, `rect`, or `point`. The value is an absolute or relative URL, or one of the special values listed below. The coordinates are `x, y` pairs separated by whitespace. The quoted text is used as the text of the link if a imagemap menu is generated. Lines beginning with '#' are comments.

## Imagemap File Directives

There are six directives allowed in the imagemap file. The directives can come in any order, but are processed in the order they are found in the imagemap file.

### base Directive

Has the effect of `<base href="value">` . The non-absolute URLs of the map-file are taken relative to this value. The base directive overrides [ImapBase](#) as set in a `.htaccess` file or in the server configuration files. In the absence of an [ImapBase](#) configuration directive, base defaults to `http://server_name/`.

`base_uri` is synonymous with `base`. Note that a trailing slash on the URL is significant.

### default Directive

The action taken if the coordinates given do not fit any of the `poly`, `circle` or `rect` directives, and there are no `point` directives. Defaults to `nocontent` in the absence of an

ImapDefault configuration setting, causing a status code of 204 No Content to be returned. The client should keep the same page displayed.

### **poly Directive**

Takes three to one-hundred points, and is obeyed if the user selected coordinates fall within the polygon defined by these points.

### **circle**

Takes the center coordinates of a circle and a point on the circle. Is obeyed if the user selected point is with the circle.

### **rect Directive**

Takes the coordinates of two opposing corners of a rectangle. Obeyed if the point selected is within this rectangle.

### **point Directive**

Takes a single point. The point directive closest to the user selected point is obeyed if no other directives are satisfied. Note that default will not be followed if a point directive is present and valid coordinates are given.

## **Values**

The values for each of the directives can any of the following:

### **a URL**

The URL can be relative or absolute URL. Relative URLs can contain '..' syntax and will be resolved relative to the base value.

base itself will not resolved according to the current value. A statement base `mailto:` will work properly, though.

### **map**

Equivalent to the URL of the imagemap file itself. No

coordinates are sent with this, so a menu will be generated unless [ImapMenu](#) is set to none.

### **menu**

Synonymous with map.

### **referer**

Equivalent to the URL of the referring document. Defaults to `http://servername/` if no `Referer:` header was present.

### **nocontent**

Sends a status code of 204 No Content, telling the client to keep the same page displayed. Valid for all but base.

### **error**

Fails with a 500 Server Error. Valid for all but base, but sort of silly for anything but default.

## **Coordinates**

### **0,0 200,200**

A coordinate consists of an *x* and a *y* value separated by a comma. The coordinates are separated from each other by whitespace. To accommodate the way Lynx handles imagemaps, should a user select the coordinate 0, 0, it is as if no coordinate had been selected.

## **Quoted Text**

### **"Menu Text"**

After the value or after the coordinates, the line optionally may contain text within double quotes. This string is used as the text for the link if a menu is generated:

```
<a href="http://foo.com/">Menu text</a>
```

If no quoted text is present, the name of the link will be used

as the text:

```
<a href="http://foo.com/">http://foo.com</a>
```

If you want to use double quotes within this text, you have to write them as `&quot;` ; .



## Example menu

```
#Comments are printed in a 'formatted' or 'semiformatted' menu.  
#And can contain html tags. <hr>  
base referer  
poly map "Could I have a menu, please?" 0,0 0,10 10,10 10,0  
rect .. 0,0 77,27 "the directory of the referer"  
circle http://www.inetnebr.com/lincoln/feedback/ 195,0 305,27  
rect another_file "in same directory as referer" 306,0 419,27  
point http://www.zyzyva.com/ 100,100  
point http://www.tripod.com/ 200,200  
rect mailto:nate@tripod.com 100,150 200,0 "Bugs?"
```



### HTML example

```
<a href="/maps/imagemap1.map">  
    
</a>
```

### XHTML example

```
<a href="/maps/imagemap1.map">  
    
</a>
```



<b>Description:</b>	Default base for imagemap files
<b>Syntax:</b>	ImapBase map referrer URL
<b>Default:</b>	ImapBase http://servername/
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_imap

The **ImapBase** directive sets the default base used in the imagemap files. Its value is overridden by a base directive within the imagemap file. If not present, the base defaults to `http://servername/`.

### See also

- [UseCanonicalName](#)



## imapDefault Directive

<b>Description:</b>	Default action when an imagemap is called with coordinates that are not explicitly mapped
<b>Syntax:</b>	ImapDefault error   nocontent   map   referer   <i>URL</i>
<b>Default:</b>	ImapDefault nocontent
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_imap

The `ImapDefault` directive sets the default default used in the imagemap files. Its value is overridden by a default directive within the imagemap file. If not present, the default action is nocontent, which means that a 204 No Content is sent to the client. In this case, the client should continue to display the original page.



## Implementation Details

<b>Description:</b>	Action if no coordinates are given when calling an imagemap
<b>Syntax:</b>	ImapMenu none   formatted   semiformatted   unformatted
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Indexes
<b>Status:</b>	Base
<b>Module:</b>	mod_imap

The `ImapMenu` directive determines the action taken if an imagemap file is called without valid coordinates.

### **none**

If `ImapMenu` is `none`, no menu is generated, and the default action is performed.

### **formatted**

A formatted menu is the simplest menu. Comments in the imagemap file are ignored. A level one header is printed, then an hrule, then the links each on a separate line. The menu has a consistent, plain look close to that of a directory listing.

### **semiformatted**

In the `semiformatted` menu, comments are printed where they occur in the imagemap file. Blank lines are turned into HTML breaks. No header or hrule is printed, but otherwise the menu is the same as a formatted menu.

### **unformatted**

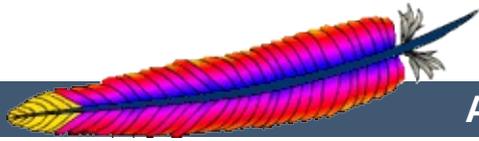
Comments are printed, blank lines are ignored. Nothing is printed that does not appear in the imagemap file. All breaks and headers must be included as comments in the imagemap file. This gives you the most flexibility over the appearance of your menus, but requires you to treat your map files as HTML

instead of plaintext.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_include

<b>Description:</b>	Server-parsed html documents (Server Side Includes)
<b>Status:</b>	Base
<b>Module Identifier:</b>	include_module
<b>Source File:</b>	mod_include.c
<b>Compatibility:</b>	Implemented as an output filter since Apache 2.0

## Summary

This module provides a filter which will process files before they are sent to the client. The processing is controlled by specially formatted SGML comments, referred to as *elements*. These elements allow conditional text, the inclusion of other files or programs, as well as the setting and printing of environment variables.

## See also

[Options](#)

[AcceptPathInfo](#)

[International Customized Server Error Messages](#)

[Filters](#)

[SSI Tutorial](#)



## Enabling Server Side Includes

Server Side Includes are implemented by the `INCLUDES` [filter](#). If documents containing server-side include directives are given the extension `.shtml`, the following directives will make Apache parse them and assign the resulting document the mime type of `text/html`:

```
AddType text/html .shtml
AddOutputFilter INCLUDES .shtml
```

The following directive must be given for the directories containing the `shtml` files (typically in a `<Directory>` section, but this directive is also valid in `.htaccess` files if `AllowOverride Options` is set):

```
Options +Includes
```

For backwards compatibility, the server-parsed [handler](#) also activates the `INCLUDES` filter. As well, Apache will activate the `INCLUDES` filter for any document with mime type `text/x-server-parsed-html` or `text/x-server-parsed-html3` (and the resulting output will have the mime type `text/html`).

For more information, see our [Tutorial on Server Side Includes](#).



---

Files processed for server-side includes no longer accept requests with PATH\_INFO (trailing pathname information) by default. You can use the [AcceptPathInfo](#) directive to configure the server to accept requests with PATH\_INFO.



The document is parsed as an HTML document, with special commands embedded as SGML comments. A command has the syntax:

```
<!--#element attribute=value attribute=value ... -->
```

The value will often be enclosed in double quotes, but single quotes (') and backticks (`) are also possible. Many commands only allow a single attribute-value pair. Note that the comment terminator (- ->) should be preceded by whitespace to ensure that it isn't considered part of an SSI token. Note that the leading <! - - # is *one* token and may not contain any whitespaces.

The allowed elements are listed in the following table:

Element	Description
<a href="#">config</a>	configure output formats
<a href="#">echo</a>	print variables
<a href="#">exec</a>	execute external programs
<a href="#">fsize</a>	print size of a file
<a href="#">flastmod</a>	print last modification time of a file
<a href="#">include</a>	include a file
<a href="#">printenv</a>	print all available variables
<a href="#">set</a>	set a value of a variable

SSI elements may be defined by modules other than [mod\\_include](#). In fact, the [exec](#) element is provided by [mod\\_cgi](#), and will only be available if this module is loaded.

## The config Element

This command controls various aspects of the parsing. The valid

attributes are:

### **errmsg**

The value is a message that is sent back to the client if an error occurs while parsing the document. This overrides any [SSIErrorMsg](#) directives.

### **sizefmt**

The value sets the format to be used when displaying the size of a file. Valid values are `bytes` for a count in bytes, or `abbrev` for a count in Kb or Mb as appropriate, for example a size of 1024 bytes will be printed as "1K".

### **timefmt**

The value is a string to be used by the `strftime(3)` library routine when printing dates.

## **The echo Element**

This command prints one of the [include variables](#), defined below. If the variable is unset, the result is determined by the [SSIUndefinedEcho](#) directive. Any dates printed are subject to the currently configured `timefmt`.

Attributes:

### **var**

The value is the name of the variable to print.

### **encoding**

Specifies how Apache should encode special characters contained in the variable before outputting them. If set to `none`, no encoding will be done. If set to `url`, then URL encoding (also known as %-encoding; this is appropriate for use within URLs in links, etc.) will be performed. At the start of an echo element, the default is set to `entity`, resulting in entity encoding (which is appropriate in the context of a block-

level HTML element, e.g. a paragraph of text). This can be changed by adding an encoding attribute, which will remain in effect until the next encoding attribute is encountered or the element ends, whichever comes first.

The encoding attribute must *precede* the corresponding var attribute to be effective, and only special characters as defined in the ISO-8859-1 character encoding will be encoded. This encoding process may not have the desired result if a different character encoding is in use.

In order to avoid cross-site scripting issues, you should *always* encode user supplied data.

## The exec Element

The exec command executes a given shell command or CGI script. It requires `mod_cgi` to be present in the server. If `Options IncludesNOEXEC` is set, this command is completely disabled.

The valid attributes are:

### **cgi**

The value specifies a (%-encoded) URL-path to the CGI script. If the path does not begin with a slash (/), then it is taken to be relative to the current document. The document referenced by this path is invoked as a CGI script, even if the server would not normally recognize it as such. However, the directory containing the script must be enabled for CGI scripts (with `ScriptAlias` or `Options ExecCGI`).

The CGI script is given the `PATH_INFO` and query string (`QUERY_STRING`) of the original request from the client; these *cannot* be specified in the URL path. The include variables will be available to the script in addition to the standard `CGI`

environment.

### Example

```
<!--#exec cgi="/cgi-bin/example.cgi" -->
```

If the script returns a `Location:` header instead of output, then this will be translated into an HTML anchor.

The `include virtual` element should be used in preference to `exec cgi`. In particular, if you need to pass additional arguments to a CGI program, using the query string, this cannot be done with `exec cgi`, but can be done with `include virtual`, as shown here:

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value" -->
```

### cmd

The server will execute the given string using `/bin/sh`. The `include variables` are available to the command, in addition to the usual set of CGI variables.

The use of `#include virtual` is almost always preferred to using either `#exec cgi` or `#exec cmd`. The former (`#include virtual`) uses the standard Apache sub-request mechanism to include files or scripts. It is much better tested and maintained.

In addition, on some platforms, like Win32, and on unix when using `suexec`, you cannot pass arguments to a command in an `exec` directive, or otherwise include spaces in the command. Thus, while the following will work under a non-`suexec` configuration on unix, it will not produce the desired result under Win32, or when running `suexec`:

```
<!--#exec cmd="perl /path/to/perlscript arg1 arg2" -->
```

## The `fsize` Element

This command prints the size of the specified file, subject to the `sizefmt` format specification. Attributes:

### **file**

The value is a path relative to the directory containing the current document being parsed.

### **virtual**

The value is a (%-encoded) URL-path. If it does not begin with a slash (/) then it is taken to be relative to the current document. Note, that this does *not* print the size of any CGI output, but the size of the CGI script itself.

## The `flastmod` Element

This command prints the last modification date of the specified file, subject to the `timefmt` format specification. The attributes are the same as for the [fsize](#) command.

## The `include` Element

This command inserts the text of another document or file into the parsed file. Any included file is subject to the usual access control. If the directory containing the parsed file has [Options](#) `IncludesNOEXEC` set, then only documents with a text MIME type (`text/plain`, `text/html` etc.) will be included. Otherwise CGI scripts are invoked as normal using the complete URL given in the command, including any query string.

An attribute defines the location of the document; the inclusion is done for each attribute given to the include command. The valid attributes are:

## **file**

The value is a path relative to the directory containing the current document being parsed. It cannot contain `../`, nor can it be an absolute path. Therefore, you cannot include files that are outside of the document root, or above the current document in the directory structure. The `virtual` attribute should always be used in preference to this one.

## **virtual**

The value is a (%-encoded) URL-path. The URL cannot contain a scheme or hostname, only a path and an optional query string. If it does not begin with a slash (/) then it is taken to be relative to the current document.

A URL is constructed from the attribute, and the output the server would return if the URL were accessed by the client is included in the parsed output. Thus included files can be nested.

If the specified URL is a CGI program, the program will be executed and its output inserted in place of the directive in the parsed file. You may include a query string in a CGI url:

```
<!--#include virtual="/cgi-bin/example.cgi?argument=value"
-->
```

`include virtual` should be used in preference to `exec cgi` to include the output of CGI programs into an HTML document.

## **The printenv Element**

This prints out a listing of all existing variables and their values. Special characters are entity encoded (see the [echo](#) element for details) before being output. There are no attributes.

---

## Example

```
<!--#printenv -->
```

## The set Element

This sets the value of a variable. Attributes:

### **var**

The name of the variable to set.

### **value**

The value to give a variable.

## Example

```
<!--#set var="category" value="help" -->
```



In addition to the variables in the standard CGI environment, these are available for the `echo` command, for `if` and `elif`, and to any program invoked by the document.

**DATE\_GMT**

The current date in Greenwich Mean Time.

**DATE\_LOCAL**

The current date in the local time zone.

**DOCUMENT\_NAME**

The filename (excluding directories) of the document requested by the user.

**DOCUMENT\_URI**

The (%-decoded) URL path of the document requested by the user. Note that in the case of nested include files, this is *not* the URL for the current document. Note also that if the URL is modified internally (e.g. by an [alias](#) or [directoryindex](#)), the modified URL is shown.

**LAST\_MODIFIED**

The last modification date of the document requested by the user.

**QUERY\_STRING\_UNESCAPED**

If a query string is present, this variable contains the (%-decoded) query string, which is *escaped* for shell usage (special characters like `&` etc. are preceded by backslashes).



Variable substitution is done within quoted strings in most cases where they may reasonably occur as an argument to an SSI directive. This includes the `config`, `exec`, `flastmod`, `fsize`, `include`, `echo`, and `set` directives, as well as the arguments to conditional operators. You can insert a literal dollar sign into the string using backslash quoting:

```
<!--#if expr="$a = \$test" -->
```

If a variable reference needs to be substituted in the middle of a character sequence that might otherwise be considered a valid identifier in its own right, it can be disambiguated by enclosing the reference in braces, a *la* shell substitution:

```
<!--#set var="Zed" value="{REMOTE_HOST}_{REQUEST_METHOD}" -->
```

This will result in the Zed variable being set to "X\_Y" if REMOTE\_HOST is "X" and REQUEST\_METHOD is "Y".

The below example will print "in foo" if the DOCUMENT\_URI is /foo/file.html, "in bar" if it is /bar/file.html and "in neither" otherwise:

```
<!--#if expr="$DOCUMENT_URI" = "/foo/file.html" -->
  in foo
<!--#elif expr="$DOCUMENT_URI" = "/bar/file.html" -->
  in bar
<!--#else -->
  in neither
<!--#endif -->
```



The basic flow control elements are:

```
<!--#if expr="test_condition" -->  
<!--#elif expr="test_condition" -->  
<!--#else -->  
<!--#endif -->
```

The `if` element works like an if statement in a programming language. The test condition is evaluated and if the result is true, then the text until the next `elif`, `else` or `endif` element is included in the output stream.

The `elif` or `else` statements are be used to put text into the output stream if the original *test\_condition* was false. These elements are optional.

The `endif` element ends the `if` element and is required.

*test\_condition* is one of the following:

***string***

true if *string* is not empty

***string1 = string2***

***string1 != string2***

Compare *string1* with *string2*. If *string2* has the form */string2/* then it is treated as a regular expression. Regular expressions are implemented by the [PCRE](#) engine and have the same syntax as those in [perl 5](#).

If you are matching positive (=), you can capture grouped parts of the regular expression. The captured parts are stored in the special variables \$1 .. \$9.

**Example**

```
<!--#if expr="$QUERY_STRING = /^sid=[a-zA-Z0-9]*/" -->  
  <!--#set var="session" value="$1" -->  
<!--#endif -->
```

***string1 < string2***  
***string1 <= string2***  
***string1 > string2***  
***string1 >= string2***

Compare *string1* with *string2*. Note, that strings are compared *literally* (using `strcmp(3)`). Therefore the string "100" is less than "20".

***( test\_condition )***

true if *test\_condition* is true

***! test\_condition***

true if *test\_condition* is false

***test\_condition1 && test\_condition2***

true if both *test\_condition1* and *test\_condition2* are true

***test\_condition1 || test\_condition2***

true if either *test\_condition1* or *test\_condition2* is true

"=" and "!=" bind more tightly than "&&" and "||". "!" binds most tightly. Thus, the following are equivalent:

```
<!--#if expr="$a = test1 && $b = test2" -->  
<!--#if expr="($a = test1) && ($b = test2)" -->
```

The boolean operators `&&` and `||` share the same priority. So if you want to bind such an operator more tightly, you should use parentheses.

Anything that's not recognized as a variable or an operator is treated as a string. Strings can also be quoted: `'string'`.

Unquoted strings can't contain whitespace (blanks and tabs) because it is used to separate tokens such as variables. If multiple

strings are found in a row, they are concatenated using blanks. So,

```
string1 string2 results in string1 string2
```

and

```
'string1 string2' results in string1 string2.
```

### **Escaping slashes in regex strings**

All slashes which are not intended to act as delimiters in your regex must be escaped. This is regardless of their meaning to the regex engine.



<b>Description:</b>	String that ends an include element
<b>Syntax:</b>	SSIEndTag <i>tag</i>
<b>Default:</b>	SSIEndTag "-->"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.0.30 and later.

This directive changes the string that [mod\\_include](#) looks for to mark the end of an include element.

### Example

```
SSIEndTag "%>"
```

### See also

- [SSIStartTag](#)



<b>Description:</b>	Error message displayed when there is an SSI error
<b>Syntax:</b>	SSIErrorMsg <i>message</i>
<b>Default:</b>	SSIErrorMsg "[an error occurred while processing this directive]"
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.0.30 and later.

The `SSIErrorMsg` directive changes the error message displayed when `mod_include` encounters an error. For production servers you may consider changing the default error message to "`<!-- Error -->`" so that the message is not presented to the user.

This directive has the same effect as the `<!--#config errmsg=message -->` element.

**Example**

```
SSIErrorMsg "<!-- Error -->"
```



<b>Description:</b>	String that starts an include element
<b>Syntax:</b>	<code>SSIStartTag tag</code>
<b>Default:</b>	<code>SSIStartTag "&lt;! - -#"</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_include</code>
<b>Compatibility:</b>	Available in version 2.0.30 and later.

This directive changes the string that `mod_include` looks for to mark an include element to process.

You may want to use this option if you have 2 servers parsing the output of a file each processing different commands (possibly at different times).

### Example

```
SSIStartTag "<%"  
SSIEndTag "%>"
```

The example given above, which also specifies a matching `SSIEndTag`, will allow you to use SSI directives as shown in the example below:

### SSI directives with alternate start and end tags

```
<%printenv %>
```

### See also

- [SSIEndTag](#)



<b>Description:</b>	Configures the format in which date strings are displayed
<b>Syntax:</b>	SSITimeFormat <i>formatstring</i>
<b>Default:</b>	SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.0.30 and later.

This directive changes the format in which date strings are displayed when echoing DATE environment variables. The *formatstring* is as in `strftime(3)` from the C standard library.

This directive has the same effect as the `<!--#config timefmt=formatstring -->` element.

### Example

```
SSITimeFormat "%R, %B %d, %Y"
```

The above directive would cause times to be displayed in the format "22:26, June 14, 2002".



<b>Description:</b>	String displayed when an unset variable is echoed
<b>Syntax:</b>	SSIUndefinedEcho <i>string</i>
<b>Default:</b>	SSIUndefinedEcho "(none)"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_include
<b>Compatibility:</b>	Available in version 2.0.34 and later.

This directive changes the string that `mod_include` displays when a variable is not set and "echoed".

### Example

```
SSIUndefinedEcho "<!-- undef -->"
```



<b>Description:</b>	Parse SSI directives in files with the execute bit set
<b>Syntax:</b>	XBitHack on off full
<b>Default:</b>	XBitHack off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Base
<b>Module:</b>	mod_include

The **XBitHack** directive controls the parsing of ordinary html documents. This directive only affects files associated with the MIME type `text/html`. **XBitHack** can take on the following values:

### **off**

No special treatment of executable files.

### **on**

Any `text/html` file that has the user-execute bit set will be treated as a server-parsed html document.

### **full**

As for `on` but also test the group-execute bit. If it is set, then set the `Last-Modified` date of the returned file to be the last modified time of the file. If it is not set, then no last-modified date is sent. Setting this bit allows clients and proxies to cache the result of the request.

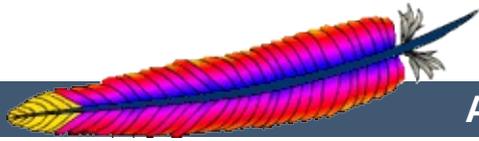
### **Note**

You would not want to use the `full` option, unless you assure the group-execute bit is unset for every SSI script which might `#include` a CGI or otherwise produces different output on each hit (or could potentially change on subsequent requests).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_info

<b>Description:</b>	Provides a comprehensive overview of the server configuration
<b>Status:</b>	Extension
<b>Module Identifier:</b>	info_module
<b>Source File:</b>	mod_info.c

## Summary

To configure `mod_info`, add the following to your `httpd.conf` file.

```
<Location /server-info>
  SetHandler server-info
</Location>
```

You may wish to use `mod_access` inside the `<Location>` directive to limit access to your server configuration information:

```
<Location /server-info>
  SetHandler server-info
  Order deny,allow
  Deny from all
  Allow from yourcompany.com
</Location>
```

Once configured, the server information is obtained by accessing `http://your.host.dom/server-info`

Note that the configuration files are read by the module at run-time, and therefore the display may *not* reflect the running server's active configuration if the files have been changed since the server was last reloaded. Also, the configuration files must be readable by the user as which the server is running (see the `User` directive), or else the directive settings will not be listed.

It should also be noted that if `mod_info` is compiled into the server, its handler capability is available in *all* configuration files, including per-directory files (e.g., `.htaccess`). This may have security-related ramifications for your site.

In particular, this module can leak sensitive information from the configuration directives of other Apache modules such as system paths, usernames/passwords, database names, etc. Due to the way this module works there is no way to block information from it. Therefore, this module should **only** be used in a controlled environment and always with caution.



<b>Description:</b>	Adds additional information to the module information displayed by the server-info handler
<b>Syntax:</b>	AddModuleInfo <i>module-name string</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_info
<b>Compatibility:</b>	Apache 1.3 and above

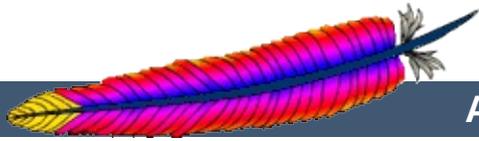
This allows the content of *string* to be shown as HTML interpreted, **Additional Information** for the module *module-name*. Example:

```
AddModuleInfo mod_auth.c 'See <a \
href="http://www.apache.org/docs/2.0/mod/mod_auth.html">\
http://www.apache.org/docs/2.0/mod/mod_auth.html</a>'
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_isapi

<b>Description:</b>	ISAPI Extensions within Apache for Windows
<b>Status:</b>	Base
<b>Module Identifier:</b>	isapi_module
<b>Source File:</b>	mod_isapi.c
<b>Compatibility:</b>	Win32 only

### Summary

This module implements the Internet Server extension API. It allows Internet Server extensions (e.g. ISAPI .dll modules) to be served by Apache for Windows, subject to the noted restrictions.

ISAPI extension modules (.dll files) are written by third parties. The Apache Group does not author these modules, so we provide no support for them. Please contact the ISAPI's author directly if you are experiencing problems running their ISAPI extension. **Please do not post such problems to Apache's lists or bug reporting pages.**



In the server configuration file, use the [AddHandler](#) directive to associate ISAPI files with the `isapi-handler` handler, and map it to them with their file extensions. To enable any .dll file to be processed as an ISAPI extension, edit the `httpd.conf` file and add the following line:

```
AddHandler isapi-handler .dll
```

In versions of the Apache server prior to 2.0.37, use `isapi-isa` instead of `isapi-handler`. The new handler name is not available prior to version 2.0.37. For compatibility, configurations may continue using `isapi-isa` through all versions of Apache prior to 2.3.0.

There is no capability within the Apache server to leave a requested module loaded. However, you may preload and keep a specific module loaded by using the following syntax in your `httpd.conf`:

```
ISAPICacheFile c:/WebWork/Scripts/ISAPI/mytest.dll
```

Whether or not you have preloaded an ISAPI extension, all ISAPI extensions are governed by the same permissions and restrictions as CGI scripts. That is, [Options](#) `ExecCGI` must be set for the directory that contains the ISAPI .dll file.

Review the [Additional Notes](#) and the [Programmer's Journal](#) for additional details and clarification of the specific ISAPI support offered by [mod\\_isapi](#).



---

Apache's ISAPI implementation conforms to all of the ISAPI 2.0 specification, except for some "Microsoft-specific" extensions dealing with asynchronous I/O. Apache's I/O model does not allow asynchronous reading and writing in a manner that the ISAPI could access. If an ISA tries to access unsupported features, including async I/O, a message is placed in the error log to help with debugging. Since these messages can become a flood, the directive `ISAPILogNotSupported Off` exists to quiet this noise.

Some servers, like Microsoft IIS, load the ISAPI extension into the server and keep it loaded until memory usage is too high, or unless configuration options are specified. Apache currently loads and unloads the ISAPI extension each time it is requested, unless the `ISAPICacheFile` directive is specified. This is inefficient, but Apache's memory model makes this the most effective method. Many ISAPI modules are subtly incompatible with the Apache server, and unloading these modules helps to ensure the stability of the server.

Also, remember that while Apache supports ISAPI Extensions, it **does not support ISAPI Filters**. Support for filters may be added at a later date, but no support is planned at this time.



If you are programming Apache 2.0 `mod_isapi` modules, you must limit your calls to `ServerSupportFunction` to the following directives:

#### **HSE\_REQ\_SEND\_URL\_REDIRECT\_RESP**

Redirect the user to another location.  
This must be a fully qualified URL (e.g. `http://server/location`).

#### **HSE\_REQ\_SEND\_URL**

Redirect the user to another location.  
This cannot be a fully qualified URL, you are not allowed to pass the protocol or a server name (e.g. simply `/location`).  
This redirection is handled by the server, not the browser.

#### **Warning**

In their recent documentation, Microsoft appears to have abandoned the distinction between the two `HSE_REQ_SEND_URL` functions. Apache continues to treat them as two distinct functions with different requirements and behaviors.

#### **HSE\_REQ\_SEND\_RESPONSE\_HEADER**

Apache accepts a response body following the header if it follows the blank line (two consecutive newlines) in the headers string argument. This body cannot contain NULLs, since the headers argument is NULL terminated.

#### **HSE\_REQ\_DONE\_WITH\_SESSION**

Apache considers this a no-op, since the session will be finished when the ISAPI returns from processing.

#### **HSE\_REQ\_MAP\_URL\_TO\_PATH**

Apache will translate a virtual name to a physical name.

## **HSE\_APPEND\_LOG\_PARAMETER**

This logged message may be captured in any of the following logs:

- in the `\"%{isapi-parameter}n\"` component in a [CustomLog](#) directive
- in the `%q` log component with the [ISAPIAppendLogToQuery On](#) directive
- in the error log with the [ISAPIAppendLogToErrors On](#) directive

The first option, the `%{isapi-parameter}n` component, is always available and preferred.

## **HSE\_REQ\_IS\_KEEP\_CONN**

Will return the negotiated Keep-Alive status.

## **HSE\_REQ\_SEND\_RESPONSE\_HEADER\_EX**

Will behave as documented, although the `fKeepConn` flag is ignored.

## **HSE\_REQ\_IS\_CONNECTED**

Will report false if the request has been aborted.

Apache returns FALSE to any unsupported call to `ServerSupportFunction`, and sets the `GetLastError` value to `ERROR_INVALID_PARAMETER`.

`ReadClient` retrieves the request body exceeding the initial buffer (defined by [ISAPIReadAheadBuffer](#)). Based on the [ISAPIReadAheadBuffer](#) setting (number of bytes to buffer prior to calling the ISAPI handler) shorter requests are sent complete to the extension when it is invoked. If the request is longer, the ISAPI extension must use `ReadClient` to retrieve the remaining request body.

`WriteClient` is supported, but only with the `HSE_IO_SYNC` flag or no option flag (value of 0). Any other `WriteClient` request will be rejected with a return value of `FALSE`, and a `GetLastError` value of `ERROR_INVALID_PARAMETER`.

`GetServerVariable` is supported, although extended server variables do not exist (as defined by other servers.) All the usual Apache CGI environment variables are available from `GetServerVariable`, as well as the `ALL_HTTP` and `ALL_RAW` values.

Apache 2.0 [`mod\_isapi`](#) supports additional features introduced in later versions of the ISAPI specification, as well as limited emulation of async I/O and the `TransmitFile` semantics. Apache also supports preloading ISAPI .dlls for performance, neither of which were not available under Apache 1.3 `mod_isapi`.



<b>Description:</b>	Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log
<b>Syntax:</b>	ISAPIAppendLogToErrors on off
<b>Default:</b>	ISAPIAppendLogToErrors off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Record HSE\_APPEND\_LOG\_PARAMETER requests from ISAPI extensions to the server error log.



## ISAPIAppendLogToQuery Directive

<b>Description:</b>	Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field
<b>Syntax:</b>	ISAPIAppendLogToQuery on off
<b>Default:</b>	ISAPIAppendLogToQuery on
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Record HSE\_APPEND\_LOG\_PARAMETER requests from ISAPI extensions to the query field (appended to the [CustomLog %q](#) component).



<b>Description:</b>	ISAPI .dll files to be loaded at startup
<b>Syntax:</b>	ISAPICacheFile <i>file-path</i> [ <i>file-path</i> ] ...
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Specifies a space-separated list of file names to be loaded when the Apache server is launched, and remain loaded until the server is shut down. This directive may be repeated for every ISAPI .dll file desired. The full path name of each file should be specified. If the path name is not absolute, it will be treated relative to [ServerRoot](#).



<b>Description:</b>	Fake asynchronous support for ISAPI callbacks
<b>Syntax:</b>	ISAPIFakeAsync on off
<b>Default:</b>	ISAPIFakeAsync off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

While set to on, asynchronous support for ISAPI callbacks is simulated.



<b>Description:</b>	Log unsupported feature requests from ISAPI extensions
<b>Syntax:</b>	ISAPILogNotSupported on off
<b>Default:</b>	ISAPILogNotSupported off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Logs all requests for unsupported features from ISAPI extensions in the server error log. This may help administrators to track down problems. Once set to on and all desired ISAPI modules are functioning, it should be set back to off.



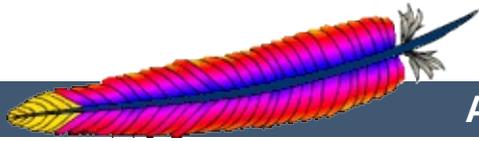
<b>Description:</b>	Size of the Read Ahead Buffer sent to ISAPI extensions
<b>Syntax:</b>	ISAPIReadAheadBuffer <i>size</i>
<b>Default:</b>	ISAPIReadAheadBuffer 49152
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_isapi

Defines the maximum size of the Read Ahead Buffer sent to ISAPI extensions when they are initially invoked. All remaining data must be retrieved using the `ReadClient` callback; some ISAPI extensions may not support the `ReadClient` function. Refer questions to the ISAPI extension's author.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_ldap

<b>Description:</b>	LDAP connection pooling and result caching services for use by other LDAP modules
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	ldap_module
<b>Source File:</b>	util_ldap.c
<b>Compatibility:</b>	Available in version 2.0.41 and later

### Summary

This module was created to improve the performance of websites relying on backend connections to LDAP servers. In addition to the functions provided by the standard LDAP libraries, this module adds an LDAP connection pool and an LDAP shared memory cache.

To enable this module, LDAP support must be compiled into apr-util. This is achieved by adding the `--with-ldap` flag to the [configure](#) script when building Apache.

SSL support requires that [mod\\_ldap](#) be linked with one of the following LDAP SDKs: [OpenLDAP SDK](#) (both 1.x and 2.x), [Novell LDAP SDK](#) or the [iPlanet\(Netscape\) SDK](#).



## Example Configuration

The following is an example configuration that uses `mod_ldap` to increase the performance of HTTP Basic authentication provided by `mod_auth_ldap`.

```
# Enable the LDAP connection pool and shared
# memory cache. Enable the LDAP cache status
# handler. Requires that mod_ldap and mod_auth_ldap
# be loaded. Change the "yourdomain.example.com" to
# match your domain.

LDAPSharedCacheSize 200000
LDAPCacheEntries 1024
LDAPCacheTTL 600
LDAPOpCacheEntries 1024
LDAPOpCacheTTL 600

<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPEnabled on
  AuthLDAPURL ldap://127.0.0.1/dc=example,dc=com?uid?one
  AuthLDAPAuthoritative on
  Require valid-user
</Location>
```



---

LDAP connections are pooled from request to request. This allows the LDAP server to remain connected and bound ready for the next request, without the need to unbind/connect/rebind. The performance advantages are similar to the effect of HTTP keepalives.

On a busy server it is possible that many requests will try and access the same LDAP server connection simultaneously. Where an LDAP connection is in use, Apache will create a new connection alongside the original one. This ensures that the connection pool does not become a bottleneck.

There is no need to manually enable connection pooling in the Apache configuration. Any module using this module for access to LDAP services will share the connection pool.



For improved performance, [mod\\_ldap](#) uses an aggressive caching strategy to minimize the number of times that the LDAP server must be contacted. Caching can easily double or triple the throughput of Apache when it is serving pages protected with `mod_auth_ldap`. In addition, the load on the LDAP server will be significantly decreased.

[mod\\_ldap](#) supports two types of LDAP caching during the search/bind phase with a *search/bind cache* and during the compare phase with two *operation caches*. Each LDAP URL that is used by the server has its own set of these three caches.

## The Search/Bind Cache

The process of doing a search and then a bind is the most time-consuming aspect of LDAP operation, especially if the directory is large. The search/bind cache is used to cache all searches that resulted in successful binds. Negative results (*i.e.*, unsuccessful searches, or searches that did not result in a successful bind) are not cached. The rationale behind this decision is that connections with invalid credentials are only a tiny percentage of the total number of connections, so by not caching invalid credentials, the size of the cache is reduced.

[mod\\_ldap](#) stores the username, the DN retrieved, the password used to bind, and the time of the bind in the cache. Whenever a new connection is initiated with the same username, [mod\\_ldap](#) compares the password of the new connection with the password in the cache. If the passwords match, and if the cached entry is not too old, [mod\\_ldap](#) bypasses the search/bind phase.

The search and bind cache is controlled with the [LDAPCacheEntries](#) and [LDAPCacheTTL](#) directives.

## Operation Caches

During attribute and distinguished name comparison functions, [mod\\_ldap](#) uses two operation caches to cache the compare operations. The first compare cache is used to cache the results of compares done to test for LDAP group membership. The second compare cache is used to cache the results of comparisons done between distinguished names.

The behavior of both of these caches is controlled with the [LDAPOpCacheEntries](#) and [LDAPOpCacheTTL](#) directives.

## Monitoring the Cache

[mod\\_ldap](#) has a content handler that allows administrators to monitor the cache performance. The name of the content handler is `ldap-status`, so the following directives could be used to access the [mod\\_ldap](#) cache information:

```
<Location /server/cache-info>
  SetHandler ldap-status
</Location>
```

By fetching the URL `http://servername/cache-info`, the administrator can get a status report of every cache that is used by [mod\\_ldap](#) cache. Note that if Apache does not support shared memory, then each [httpd](#) instance has its own cache, so reloading the URL will result in different information each time, depending on which [httpd](#) instance processes the request.



The ability to create an SSL connections to an LDAP server is defined by the directives [LDAPTrustedCA](#) and [LDAPTrustedCAType](#). These directives specify the certificate file or database and the certificate type. Whenever the LDAP url includes *ldaps://*, [mod\\_ldap](#) will establish a secure connection to the LDAP server.

```
# Establish an SSL LDAP connection. Requires that
# mod_ldap and mod_auth_ldap be loaded. Change the
# "yourdomain.example.com" to match your domain.

LDAPTrustedCA /certs/certfile.der
LDAPTrustedCAType DER_FILE

<Location /ldap-status>
  SetHandler ldap-status
  Order deny,allow
  Deny from all
  Allow from yourdomain.example.com
  AuthLDAPEnabled on
  AuthLDAPURL ldaps://127.0.0.1/dc=example,dc=com?uid?one
  AuthLDAPAuthoritative on
  Require valid-user
</Location>
```

If [mod\\_ldap](#) is linked against the Netscape/iPlanet LDAP SDK, it will not talk to any SSL server unless that server has a certificate signed by a known Certificate Authority. As part of the configuration [mod\\_ldap](#) needs to be told where it can find a database containing the known CAs. This database is in the same format as Netscape Communicator's `cert7.db` database. The easiest way to get this file is to start up a fresh copy of Netscape, and grab the resulting `$HOME/.netscape/cert7.db` file.



<b>Description:</b>	Maximum number of entries in the primary LDAP cache
<b>Syntax:</b>	LDAPCacheEntries <i>number</i>
<b>Default:</b>	LDAPCacheEntries 1024
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the maximum size of the primary LDAP cache. This cache contains successful search/binds. Set it to 0 to turn off search/bind caching. The default size is 1024 cached searches.



<b>Description:</b>	Time that cached items remain valid
<b>Syntax:</b>	LDAPCacheTTL <i>seconds</i>
<b>Default:</b>	LDAPCacheTTL 600
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the time (in seconds) that an item in the search/bind cache remains valid. The default is 600 seconds (10 minutes).



<b>Description:</b>	Specifies the socket connection timeout in seconds
<b>Syntax:</b>	LDAPConnectionTimeout <i>seconds</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the timeout value (in seconds) in which the module will attempt to connect to the LDAP server. If a connection is not successful with the timeout period, either an error will be returned or the module will attempt to connect to a secondary LDAP server if one is specified. The default is 10 seconds.



**Description:** Number of entries used to cache LDAP compare operations

**Syntax:** LDAPOpCacheEntries *number*

**Default:** LDAPOpCacheEntries 1024

**Context:** server config

**Status:** Experimental

**Module:** mod\_ldap

This specifies the number of entries [mod\\_ldap](#) will use to cache LDAP compare operations. The default is 1024 entries. Setting it to 0 disables operation caching.



<b>Description:</b>	Time that entries in the operation cache remain valid
<b>Syntax:</b>	LDAPOpCacheTTL <i>seconds</i>
<b>Default:</b>	LDAPOpCacheTTL 600
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the time (in seconds) that entries in the operation cache remain valid. The default is 600 seconds.



<b>Description:</b>	Sets the shared memory cache file
<b>Syntax:</b>	LDAPSharedCacheFile <i>directory-path/filename</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the directory path and file name of the shared memory cache file. If not set, anonymous shared memory will be used if the platform supports it.



<b>Description:</b>	Size in bytes of the shared-memory cache
<b>Syntax:</b>	LDAPSharedCacheSize <i>bytes</i>
<b>Default:</b>	LDAPSharedCacheSize 102400
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

Specifies the number of bytes to allocate for the shared memory cache. The default is 100kb. If set to 0, shared memory caching will not be used.



<b>Description:</b>	Sets the file containing the trusted Certificate Authority certificate or database
<b>Syntax:</b>	LDAPTrustedCA <i>directory-path/filename</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

It specifies the directory path and file name of the trusted CA [mod\\_ldap](#) should use when establishing an SSL connection to an LDAP server. If using the Netscape/iPlanet Directory SDK, the file name should be `cert7.db`.



<b>Description:</b>	Specifies the type of the Certificate Authority file
<b>Syntax:</b>	LDAPTrustedCAType <i>type</i>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_ldap

The following types are supported:

DER\_FILE - file in binary DER format

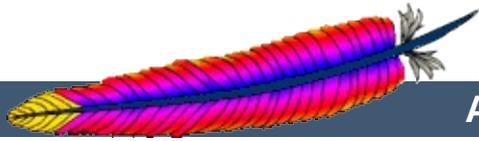
BASE64\_FILE - file in Base64 format

CERT7\_DB\_PATH - Netscape certificate database file ")

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_log_config`

<b>Description:</b>	Logging of the requests made to the server
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>log_config_module</code>
<b>Source File:</b>	<code>mod_log_config.c</code>

## Summary

This module provides for flexible logging of client requests. Logs are written in a customizable format, and may be written directly to a file, or to an external program. Conditional logging is provided so that individual requests may be included or excluded from the logs based on characteristics of the request.

Three directives are provided by this module: [TransferLog](#) to create a log file, [LogFormat](#) to set a custom format, and [CustomLog](#) to define a log file and format in one step. The [TransferLog](#) and [CustomLog](#) directives can be used multiple times in each server to cause each request to be logged to multiple files.

## See also

[Apache Log Files](#)



## Custom Log Format

The format argument to the [LogFormat](#) and [CustomLog](#) directives is a string. This string is used to log each request to the log file. It can contain literal characters copied into the log files and the C-style control characters "\n" and "\t" to represent new-lines and tabs. Literal quotes and backslashes should be escaped with backslashes.

The characteristics of the request itself are logged by placing "%" directives in the format string, which are replaced in the log file by the values as follows:

Format String	Description
%%	The percent sign ( <i>Apache 2.0.44 and later</i> )
%. . . a	Remote IP-address
%. . . A	Local IP-address
%. . . B	Size of response in bytes, excluding HTTP headers.
%. . . b	Size of response in bytes, excluding HTTP headers. In CLF format, <i>i.e.</i> a '-' rather than a 0 when no bytes are sent.
%. . . { <i>Foo</i> bar}C	The contents of cookie <i>Foo</i> bar in the request sent to the server.
%. . . D	The time taken to serve the request, in microseconds.
%. . . { <i>FOO</i> BAR}e	The contents of the environment variable <i>FOO</i> BAR
%. . . f	Filename
%. . . h	Remote host
%. . . H	The request protocol
%. . . { <i>Foo</i> bar}i	The contents of <i>Foo</i> bar : header line(s) in the request sent to the server. Changes made by

	other modules (e.g. <a href="#">mod_headers</a> ) affect this.
<code>%. . . l</code>	Remote logname (from identd, if supplied). This will return a dash unless <a href="#">IdentityCheck</a> is set On.
<code>%. . . m</code>	The request method
<code>%. . . {Foobar}n</code>	The contents of note <i>Foobar</i> from another module.
<code>%. . . {Foobar}o</code>	The contents of <i>Foobar</i> : header line(s) in the reply.
<code>%. . . p</code>	The canonical port of the server serving the request
<code>%. . . P</code>	The process ID of the child that serviced the request.
<code>%. . . {format}P</code>	The process ID or thread id of the child that serviced the request. Valid formats are <code>pid</code> and <code>tid</code> . ( <i>Apache 2.0.46 and later</i> )
<code>%. . . q</code>	The query string (prepended with a <code>?</code> if a query string exists, otherwise an empty string)
<code>%. . . r</code>	First line of request
<code>%. . . s</code>	Status. For requests that got internally redirected, this is the status of the <i>*original*</i> request --- <code>%. . . &gt;s</code> for the last.
<code>%. . . t</code>	Time the request was received (standard english format)
<code>%. . . {format}t</code>	The time, in the form given by <code>format</code> , which should be in <code>strftime(3)</code> format. (potentially localized)
<code>%. . . T</code>	The time taken to serve the request, in seconds.
<code>%. . . u</code>	Remote user (from auth; may be bogus if return status (%s) is 401)

%...U	The URL path requested, not including any query string.
%...V	The canonical <a href="#">ServerName</a> of the server serving the request.
%...V	The server name according to the <a href="#">UseCanonicalName</a> setting.
%...X	<p>Connection status when response is completed:</p> <div style="border: 1px solid black; padding: 5px;"> <p>X connection aborted before the response = completed.  + connection may be kept alive after the = response is sent.  - connection will be closed after the response = is sent.</p> </div> <p>(This directive was %...c in late versions of Apache 1.3, but this conflicted with the historical ssl %...{var}c syntax.)</p>
%...I	Bytes received, including request and headers, cannot be zero. You need to enable <a href="#">mod_logio</a> to use this.
%...O	Bytes sent, including headers, cannot be zero. You need to enable <a href="#">mod_logio</a> to use this.

The "... " can be nothing at all (e.g., "%h %u %r %s %b"), or it can indicate conditions for inclusion of the item (which will cause it to be replaced with "-" if the condition is not met). The forms of condition are a list of HTTP status codes, which may or may not be preceded by "!". Thus, "%400,501{User-agent}i" logs User-agent : on 400 errors and 501 errors (Bad Request, Not Implemented) only; "%!200,304,302{Referer}i" logs Referer : on all requests which did *not* return some sort of normal status.

The modifiers "<" and ">" can be used for requests that have been internally redirected to choose whether the original or final (respectively) request should be consulted. By default, the % directives %s, %U, %T, %D, and %r look at the original request while all others look at the final request. So for example, %>s can be used to record the final status of the request and %<u can be used to record the original authenticated user on a request that is internally redirected to an unauthenticated resource.

Note that in httpd 2.0 versions prior to 2.0.46, no escaping was performed on the strings from %...r, %...i and %...o. This was mainly to comply with the requirements of the Common Log Format. This implied that clients could insert control characters into the log, so you had to be quite careful when dealing with raw log files.

For security reasons, starting with 2.0.46, non-printable and other special characters are escaped mostly by using \xhh sequences, where hh stands for the hexadecimal representation of the raw byte. Exceptions from this rule are " and \ which are escaped by prepending a backslash, and all whitespace characters which are written in their C-style notation (\n, \t etc).

Note that in httpd 2.0, unlike 1.3, the %b and %B format strings do not represent the number of bytes sent to the client, but simply the size in bytes of the HTTP response (which will differ, for instance, if the connection is aborted, or if SSL is used). The %O format provided by [mod\\_logio](#) will log the actual number of bytes sent over the network.

Some commonly used log format strings are:

### **Common Log Format (CLF)**

```
"%h %l %u %t \"%r\" %>s %b"
```

### **Common Log Format with Virtual Host**

```
"%v %h %l %u %t \"%r\" %>s %b"
```

### **NCSA extended/combined log format**

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"  
\"%{User-agent}i\""
```

### **Referer log format**

```
"%{Referer}i -> %U"
```

### **Agent (Browser) log format**

```
"%{User-agent}i"
```

Note that the canonical [ServerName](#) and [Listen](#) of the server serving the request are used for %v and %p respectively. This happens regardless of the [UseCanonicalName](#) setting because otherwise log analysis programs would have to duplicate the entire vhost matching algorithm in order to decide what host really served the request.



See the [security tips](#) document for details on why your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.



## BufferedLogs Directive

<b>Description:</b>	Buffer log entries in memory before writing to disk
<b>Syntax:</b>	BufferedLogs On Off
<b>Default:</b>	BufferedLogs Off
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_log_config
<b>Compatibility:</b>	Available in versions 2.0.41 and later.

The **BufferedLogs** directive causes `mod_log_config` to store several log entries in memory and write them together to disk, rather than writing them after each request. On some systems, this may result in more efficient disk access and hence higher performance. It may be set only once for the entire server; it cannot be configured per virtual-host.

This directive is experimental and should be used with caution.



## CookieLog Directive

<b>Description:</b>	Sets filename for the logging of cookies
<b>Syntax:</b>	CookieLog <i>filename</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_log_config
<b>Compatibility:</b>	This directive is deprecated.

The **CookieLog** directive sets the filename for logging of cookies. The filename is relative to the **ServerRoot**. This directive is included only for compatibility with `mod_cookies`, and is deprecated.



<b>Description:</b>	Sets filename and format of log file
<b>Syntax:</b>	<code>CustomLog <i>file</i> <i>pipe</i> <i>format</i> <i>nickname</i> [env=[!]<i>environment-variable</i>]</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_log_config

The **CustomLog** directive is used to log requests to the server. A log format is specified, and the logging can optionally be made conditional on request characteristics using environment variables.

The first argument, which specifies the location to which the logs will be written, can take one of the following two types of values:

#### ***file***

A filename, relative to the **ServerRoot**.

#### ***pipe***

The pipe character "|", followed by the path to a program to receive the log information on its standard input.

#### **Security:**

If a program is used, then it will be run as the user who started `httpd`. This will be root if the server was started by root; be sure that the program is secure.

#### **Note**

When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashes are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.

---

The second argument specifies what will be written to the log file. It can specify either a *nickname* defined by a previous [LogFormat](#) directive, or it can be an explicit *format* string as described in the [log formats](#) section.

For example, the following two sets of directives have exactly the same effect:

```
# 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"
```

The third argument is optional and controls whether or not to log a particular request based on the presence or absence of a particular variable in the server environment. If the specified [environment variable](#) is set for the request (or is not set, in the case of a 'env=!*name*' clause), then the request will be logged.

Environment variables can be set on a per-request basis using the [mod\\_setenvif](#) and/or [mod\\_rewrite](#) modules. For example, if you want to record requests for all GIF images on your server in a separate logfile but not in your main log, you can use:

```
SetEnvIf Request_URI \.gif$ gif-image
CustomLog gif-requests.log common env=gif-image
CustomLog nongif-requests.log common env=!gif-image
```

Or, to reproduce the behavior of the old `RefererIgnore` directive, you might use the following:

```
SetEnvIf Referer example\.com localreferer
CustomLog referer.log referer env=!localreferer
```



## LogFormat Directive

<b>Description:</b>	Describes a format for use in a log file
<b>Syntax:</b>	LogFormat <i>format</i> [ <i>nickname</i> [ <i>nickname</i> ]
<b>Default:</b>	LogFormat "%h %l %u %t \"%r\" %>s %b"
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_log_config

This directive specifies the format of the access log file.

The **LogFormat** directive can take one of two forms. In the first form, where only one argument is specified, this directive sets the log format which will be used by logs specified in subsequent **TransferLog** directives. The single argument can specify an explicit *format* as discussed in the [custom log formats](#) section above. Alternatively, it can use a *nickname* to refer to a log format defined in a previous **LogFormat** directive as described below.

The second form of the **LogFormat** directive associates an explicit *format* with a *nickname*. This *nickname* can then be used in subsequent **LogFormat** or **CustomLog** directives rather than repeating the entire format string. A **LogFormat** directive that defines a nickname **does nothing else** -- that is, it *only* defines the nickname, it doesn't actually apply the format and make it the default. Therefore, it will not affect subsequent **TransferLog** directives. In addition, **LogFormat** cannot use one nickname to define another nickname. Note that the nickname should not contain percent signs (%).

### Example

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b" vhost_common
```

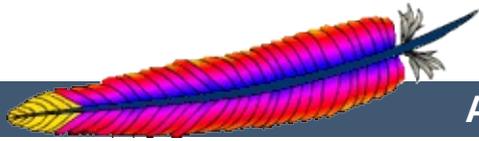


<b>Description:</b>	Specify location of a log file
<b>Syntax:</b>	TransferLog <i>file pipe</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_log_config

This directive has exactly the same arguments and effect as the [CustomLog](#) directive, with the exception that it does not allow the log format to be specified explicitly or for conditional logging of requests. Instead, the log format is determined by the most recently specified [LogFormat](#) directive which does not define a nickname. Common Log Format is used if no other format has been specified.

### Example

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""  
TransferLog logs/access_log
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_log_forensic`

<b>Description:</b>	Forensic Logging of the requests made to the server
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>log_forensic_module</code>
<b>Source File:</b>	<code>mod_log_forensic.c</code>
<b>Compatibility:</b>	Available in version 2.0.50 and later

## Summary

This module provides for forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request. The forensic logger is very strict, which means:

- The format is fixed. You cannot modify the logging format at runtime.
- If it cannot write its data, the child process exits immediately and may dump core (depending on your [CoreDumpDirectory](#) configuration).

The `check_forensic` script, which can be found in the distribution's support directory, may be helpful in evaluating the forensic log output.

This module was backported from version 2.1 which uses a more powerful APR version in order to generate the forensic IDs. If you want to run `mod_log_forensic` in version 2.0, you need to include `mod_unique_id` as well.

## See also

[Apache Log Files](#)

mod\_log\_config



## Forensic Log Format

Each request is logged two times. The first time is *before* it's processed further (that is, after receiving the headers). The second log entry is written *after* the request processing at the same time where normal logging occurs.

In order to identify each request, a unique request ID is assigned. This forensic ID can be cross logged in the normal transfer log using the `%{forensic-id}` format string. If you're using [mod\\_unique\\_id](#), its generated ID will be used.

The first line logs the forensic ID, the request line and all received headers, separated by pipe characters (`|`). A sample line looks like the following (all on one line):

```
+yQtJf8CoAB4AAFNBIEAAAAA|GET /manual/de/images/down.gif
HTTP/1.1|Host:localhost%3a8080|User-Agent:Mozilla/5.0 (X11; U;
Linux i686; en-US; rv%3a1.6) Gecko/20040216
Firefox/0.8|Accept:image/png, etc...
```

The plus character at the beginning indicates that this is the first log line of this request. The second line just contains a minus character and the ID again:

```
-yQtJf8CoAB4AAFNBIEAAAAA
```

The `check_forensic` script takes as its argument the name of the logfile. It looks for those +/- ID pairs and complains if a request was not completed.



## Security Considerations

See the [security tips](#) document for details on why your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.



## ForensicLog Directive

<b>Description:</b>	Sets filename of the forensic log
<b>Syntax:</b>	ForensicLog <i>filename pipe</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_log_forensic

The **ForensicLog** directive is used to log requests to the server for forensic analysis. Each log entry is assigned a unique ID which can be associated with the request using the normal **CustomLog** directive. **mod\_log\_forensic** takes the unique ID from **mod\_unique\_id**, so you need to load this module as well. (This requirement will not be necessary in version 2.1 and later, because of a more powerful APR version.) The ID token is attached to the request under the name `forensic-id`, which can be added to the transfer log using the `%{forensic-id}n` format string.

The argument, which specifies the location to which the logs will be written, can take one of the following two types of values:

### ***filename***

A filename, relative to the **ServerRoot**.

### ***pipe***

The pipe character "|", followed by the path to a program to receive the log information on its standard input. The program name can be specified relative to the **ServerRoot** directive.

### **Security:**

If a program is used, then it will be run as the user who started **httpd**. This will be root if the server was started by root; be sure that the program is secure or switches to a

less privileged user.

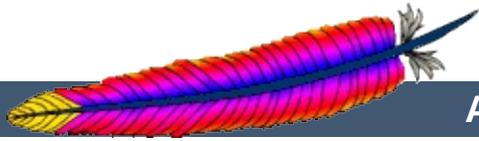
### **Note**

When entering a file path on non-Unix platforms, care should be taken to make sure that only forward slashed are used even though the platform may allow the use of back slashes. In general it is a good idea to always use forward slashes throughout the configuration files.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_logio

<b>Description:</b>	Logging of input and output bytes per request
<b>Status:</b>	Extension
<b>Module Identifier:</b>	logio_module
<b>Source File:</b>	mod_logio.c

## Summary

This module provides the logging of input and output number of bytes received/sent per request. The numbers reflect the actual bytes as received on the network, which then takes into account the headers and bodies of requests and responses. The counting is done before SSL/TLS on input and after SSL/TLS on output, so the numbers will correctly reflect any changes made by encryption.

This module requires [mod\\_log\\_config](#).

## See also

[mod\\_log\\_config](#)

[Apache Log Files](#)



## Custom Log Format

This module adds two new logging directives. The characteristics of the request itself are logged by placing "%" directives in the format string, which are replaced in the log file by the values as follows:

Format String	Description
%. . . I	Bytes received, including request and headers, cannot be zero.
%. . . O	Bytes sent, including headers, cannot be zero.

Usually, the functionality is used like this:

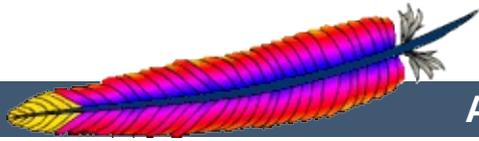
### Combined I/O log format:

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\"  
\"%{User-agent}i\" %I %O"
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_mem_cache`

<b>Description:</b>	Content cache keyed to URIs
<b>Status:</b>	Experimental
<b>Module Identifier:</b>	<code>mem_cache_module</code>
<b>Source File:</b>	<code>mod_mem_cache.c</code>

## Summary

This module is experimental. Documentation is still under development...

This module *requires* the service of `mod_cache`. It acts as a support module for `mod_cache` and provides a memory based storage manager. `mod_mem_cache` can be configured to operate in two modes: caching open file descriptors or caching objects in heap storage. `mod_mem_cache` is most useful when used to cache locally generated content or to cache backend server content for `mod_proxy` configured for `ProxyPass` (aka *reverse proxy*).

Content is stored in and retrieved from the cache using URI based keys. Content with access protection is not cached.

## See also

[`mod\_cache`](#)

[`mod\_disk\_cache`](#)



## MCACHEMAXOBJECTCOUNT DIRECTIVE

<b>Description:</b>	The maximum number of objects allowed to be placed in the cache
<b>Syntax:</b>	<code>MCacheMaxObjectCount <i>value</i></code>
<b>Default:</b>	<code>MCacheMaxObjectCount 1009</code>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_mem_cache</code>

The `MCacheMaxObjectCount` directive sets the maximum number of objects to be cached. The value is used to create the open hash table. If a new object needs to be inserted in the cache and the maximum number of objects has been reached, an object will be removed to allow the new object to be cached. The object to be removed is selected using the algorithm specified by [MCacheRemovalAlgorithm](#).

**Example**

```
MCacheMaxObjectCount 13001
```



## MCacheMaxObjectSize Directive

<b>Description:</b>	The maximum size (in bytes) of a document allowed in the cache
<b>Syntax:</b>	<code>MCacheMaxObjectSize</code> <i>bytes</i>
<b>Default:</b>	<code>MCacheMaxObjectSize</code> 10000
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_mem_cache</code>

The `MCacheMaxObjectSize` directive sets the maximum allowable size, in bytes, of a document for it to be considered cacheable.

**Example**

```
MCacheMaxObjectSize 6400000
```

**Note**

The value of `MCacheMaxObjectSize` must be greater than the value specified by the `MCacheMinObjectSize` directive.



## MCacheMaxStreamingBuffer Directive

<b>Description:</b>	Maximum amount of a streamed response to buffer in memory before declaring the response uncacheable
<b>Syntax:</b>	<code>MCacheMaxStreamingBuffer</code> <i>size_in_bytes</i>
<b>Default:</b>	<code>MCacheMaxStreamingBuffer</code> the smaller of 100000 or <code>MCacheMaxObjectSize</code>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_mem_cache</code>

The `MCacheMaxStreamingBuffer` directive specifies the maximum number of bytes of a streamed response to buffer before deciding that the response is too big to cache. A streamed response is one in which the entire content is not immediately available and in which the Content - Length may not be known. Sources of streaming responses include proxied responses and the output of CGI scripts. By default, a streamed response will *not* be cached unless it has a Content - Length header. The reason for this is to avoid using a large amount of memory to buffer a partial response that might end up being too large to fit in the cache. The `MCacheMaxStreamingBuffer` directive allows buffering of streamed responses that don't contain a Content - Length up to the specified maximum amount of space. If the maximum buffer space is reached, the buffered content is discarded and the attempt to cache is abandoned.

**Note:**

Using a nonzero value for `MCacheMaxStreamingBuffer` will not delay the transmission of the response to the client. As soon as `mod_mem_cache` copies a block of streamed content into a

buffer, it sends the block on to the next output filter for delivery to the client.

```
# Enable caching of streamed responses up to 64KB:  
MCacheMaxStreamingBuffer 65536
```



## MCacheMinObjectSize Directive

<b>Description:</b>	The minimum size (in bytes) of a document to be allowed in the cache
<b>Syntax:</b>	<code>MCacheMinObjectSize <i>bytes</i></code>
<b>Default:</b>	<code>MCacheMinObjectSize 0</code>
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	<code>mod_mem_cache</code>

The `MCacheMinObjectSize` directive sets the minimum size in bytes of a document for it to be considered cacheable.

**Example**

```
MCacheMinObjectSize 10000
```



## MCacheRemovalAlgorithm Directive

<b>Description:</b>	The algorithm used to select documents for removal from the cache
<b>Syntax:</b>	MCacheRemovalAlgorithm LRU GDSF
<b>Default:</b>	MCacheRemovalAlgorithm GDSF
<b>Context:</b>	server config
<b>Status:</b>	Experimental
<b>Module:</b>	mod_mem_cache

The `MCacheRemovalAlgorithm` directive specifies the algorithm used to select documents for removal from the cache. Two choices are available:

### LRU (Least Recently Used)

LRU removes the documents that have not been accessed for the longest time.

### GDSF (GreedyDual-Size)

GDSF assigns a priority to cached documents based on the cost of a cache miss and the size of the document. Documents with the lowest priority are removed first.

```
Example  
MCacheRemovalAlgorithm GDSF  
MCacheRemovalAlgorithm LRU
```



**Description:** The maximum amount of memory used by the cache in KBytes

**Syntax:** `MCacheSize` *KBytes*

**Default:** `MCacheSize` 100

**Context:** server config

**Status:** Experimental

**Module:** `mod_mem_cache`

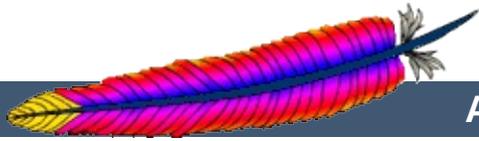
The `MCacheSize` directive sets the maximum amount of memory to be used by the cache, in KBytes (1024-byte units). If a new object needs to be inserted in the cache and the size of the object is greater than the remaining memory, objects will be removed until the new object can be cached. The object to be removed is selected using the algorithm specified by `MCacheRemovalAlgorithm`.

### Example

```
MCacheSize 700000
```

### Note

The `MCacheSize` value must be greater than the value specified by the `MCacheMaxObjectSize` directive.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_mime

<b>Description:</b>	Associates the requested filename's extensions with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding)
<b>Status:</b>	Base
<b>Module Identifier:</b>	mime_module
<b>Source File:</b>	mod_mime.c

## Summary

This module is used to associate various bits of "meta information" with files by their filename extensions. This information relates the filename of the document to its mime-type, language, character set and encoding. This information is sent to the browser, and participates in content negotiation, so the user's preferences are respected when choosing one of several possible files to serve. See [mod\\_negotiation](#) for more information about [content negotiation](#).

The directives [AddCharset](#), [AddEncoding](#), [AddLanguage](#) and [AddType](#) are all used to map file extensions onto the meta-information for that file. Respectively they set the character set, content-encoding, content-language, and MIME-type (content-type) of documents. The directive [TypesConfig](#) is used to specify a file which also maps extensions onto MIME types.

In addition, [mod\\_mime](#) may define the [handler](#) and [filters](#) that originate and process content. The directives [AddHandler](#), [AddOutputFilter](#), and [AddInputFilter](#) control the modules or scripts that serve the document. The [MultiviewsMatch](#) directive allows [mod\\_negotiation](#) to consider these file extensions to be included when testing Multiviews matches.

While [mod\\_mime](#) associates meta-information with filename extensions, the [core](#) server provides directives that are used to associate all the files in a given container (e.g., [<Location>](#), [<Directory>](#), or [<Files>](#)) with particular meta-information. These directives include [ForceType](#), [SetHandler](#), [SetInputFilter](#), and [SetOutputFilter](#). The core directives override any filename extension mappings defined in [mod\\_mime](#).

Note that changing the meta-information for a file does not change the value of the Last-Modified header. Thus, previously cached copies may still be used by a client or proxy, with the previous headers. If you change the meta-information (language, content type, character set or encoding) you may need to 'touch' affected files (updating their last modified date) to ensure that all visitors are receive the corrected content headers.

## See also

- [MimeMagicFile](#)
- [AddDefaultCharset](#)
- [ForceType](#)
- [DefaultType](#)
- [SetHandler](#)
- [SetInputFilter](#)
- [SetOutputFilter](#)



## Files with Multiple Extensions

Files can have more than one extension, and the order of the extensions is *normally* irrelevant. For example, if the file `welcome.html.fr` maps onto content type `text/html` and language French then the file `welcome.fr.html` will map onto exactly the same information. If more than one extension is given which maps onto the same type of meta-information, then the one to the right will be used, except for languages and content encodings. For example, if `.gif` maps to the MIME-type `image/gif` and `.html` maps to the MIME-type `text/html`, then the file `welcome.gif.html` will be associated with the MIME-type `text/html`.

Languages and content encodings are treated accumulative, because one can assign more than one language or encoding to a particular resource. For example, the file `welcome.html.en.de` will be delivered with `Content-Language: en, de` and `Content-Type: text/html`.

Care should be taken when a file with multiple extensions gets associated with both a MIME-type and a handler. This will usually result in the request being by the module associated with the handler. For example, if the `.imap` extension is mapped to the handler `imap-file` (from `mod_imap`) and the `.html` extension is mapped to the MIME-type `text/html`, then the file `world.imap.html` will be associated with both the `imap-file` handler and `text/html` MIME-type. When it is processed, the `imap-file` handler will be used, and so it will be treated as a `mod_imap` imagemap file.



## Content-Encoding

A file of a particular MIME type can additionally be encoded a particular way to simplify transmission over the Internet. While this usually will refer to compression, such as `gzip`, it can also refer to encryption, such as `pgp` or to an encoding such as `UUencoding`, which is designed for transmitting a binary file in an ASCII (text) format.

The [HTTP/1.1 RFC](#), section 14.11 puts it this way:

*The Content-Encoding entity-header field is used as a modifier to the media-type. When present, its value indicates what additional content codings have been applied to the entity-body, and thus what decoding mechanisms must be applied in order to obtain the media-type referenced by the Content-Type header field. Content-Encoding is primarily used to allow a document to be compressed without losing the identity of its underlying media type.*

By using more than one file extension (see [section above about multiple file extensions](#)), you can indicate that a file is of a particular *type*, and also has a particular *encoding*.

For example, you may have a file which is a Microsoft Word document, which is pkzipped to reduce its size. If the `.doc` extension is associated with the Microsoft Word file type, and the `.zip` extension is associated with the `pkzip` file encoding, then the file `Resume.doc.zip` would be known to be a pkzip'ed Word document.

Apache sends a `Content-encoding` header with the resource, in order to tell the client browser about the encoding method.

```
Content-encoding: pkzip
```



In addition to file type and the file encoding, another important piece of information is what language a particular document is in, and in what character set the file should be displayed. For example, the document might be written in the Vietnamese alphabet, or in Cyrillic, and should be displayed as such. This information, also, is transmitted in HTTP headers.

The character set, language, encoding and mime type are all used in the process of content negotiation (See [mod\\_negotiation](#)) to determine which document to give to the client, when there are alternative documents in more than one character set, language, encoding or mime type. All filename extensions associations created with [AddCharset](#), [AddEncoding](#), [AddLanguage](#) and [AddType](#) directives (and extensions listed in the [MimeMagicFile](#)) participate in this select process. Filename extensions that are only associated using the [AddHandler](#), [AddInputFilter](#) or [AddOutputFilter](#) directives may be included or excluded from matching by using the [MultiviewsMatch](#) directive.

### Charset

To convey this further information, Apache optionally sends a Content - Language header, to specify the language that the document is in, and can append additional information onto the Content - Type header to indicate the particular character set that should be used to correctly render the information.

```
Content-Language: en, fr
Content-Type: text/plain; charset=ISO-8859-1
```

The language specification is the two-letter abbreviation for the language. The char set is the name of the particular character

set which should be used.



<b>Description:</b>	Maps the given filename extensions to the specified content charset
<b>Syntax:</b>	<code>AddCharset <i>charset</i> <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **AddCharset** directive maps the given filename extensions to the specified content charset. *charset* is the [MIME charset parameter](#) of filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

### Example

```
AddLanguage ja .ja
AddCharset EUC-JP .euc
AddCharset ISO-2022-JP .jis
AddCharset SHIFT_JIS .sjis
```

Then the document `xxxx.ja.jis` will be treated as being a Japanese document whose charset is ISO-2022-JP (as will the document `xxxx.jis.ja`). The **AddCharset** directive is useful for both to inform the client about the character encoding of the document so that the document can be interpreted and displayed appropriately, and for [content negotiation](#), where the server returns one from several documents based on the client's charset preference.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

## See also

- [mod\\_negotiation](#)
- [AddDefaultCharset](#)



<b>Description:</b>	Maps the given filename extensions to the specified encoding type
<b>Syntax:</b>	AddEncoding <i>MIME-enc extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **AddEncoding** directive maps the given filename extensions to the specified encoding type. *MIME-enc* is the MIME encoding to use for documents containing the *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

**Example**

```
AddEncoding x-gzip .gz
AddEncoding x-compress .Z
```

This will cause filenames containing the .gz extension to be marked as encoded using the x-gzip encoding, and filenames containing the .Z extension to be marked as encoded with x-compress.

Old clients expect x-gzip and x-compress, however the standard dictates that they're equivalent to gzip and compress respectively. Apache does content encoding comparisons by ignoring any leading x-. When responding with an encoding Apache will use whatever form (*i.e.*, x-foo or foo) the client requested. If the client didn't specifically request a particular form Apache will use the form given by the AddEncoding directive. To make this long story short, you should always use x-gzip and x-

compress for these two specific encodings. More recent encodings, such as deflate should be specified without the x - .

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.



<b>Description:</b>	Maps the filename extensions to the specified handler
<b>Syntax:</b>	AddHandler <i>handler-name extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

Files having the name *extension* will be served by the specified [handler-name](#). This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*. For example, to activate CGI scripts with the file extension `.cgi`, you might use:

```
AddHandler cgi-script .cgi
```

Once that has been put into your `httpd.conf` file, any file containing the `.cgi` extension will be treated as a CGI program.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

## See also

- [SetHandler](#)



<b>Description:</b>	Maps filename extensions to the filters that will process client requests
<b>Syntax:</b>	<code>AddInputFilter <i>filter</i> [<i>;filter...</i>] <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	AddInputFilter is only available in Apache 2.0.26 and later.

**AddInputFilter** maps the filename extension *extension* to the [filters](#) which will process client requests and POST input when they are received by the server. This is in addition to any filters defined elsewhere, including the [SetInputFilter](#) directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same *extension*.

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content. Both the filter and *extension* arguments are case-insensitive, and the extension may be specified with or without a leading dot.

## See also

- [RemoveInputFilter](#)
- [SetInputFilter](#)



## AddLanguage Directive

<b>Description:</b>	Maps the given filename extension to the specified content language
<b>Syntax:</b>	AddLanguage <i>MIME-lang extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **AddLanguage** directive maps the given filename extension to the specified content language. *MIME-lang* is the MIME language of filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*.

### Example

```
AddEncoding x-compress .Z
AddLanguage en .en
AddLanguage fr .fr
```

Then the document `xxxx.en.Z` will be treated as being a compressed English document (as will the document `xxxx.Z.en`). Although the content language is reported to the client, the browser is unlikely to use this information. The **AddLanguage** directive is more useful for [content negotiation](#), where the server returns one from several documents based on the client's language preference.

If multiple language assignments are made for the same extension, the last one encountered is the one that is used. That is, for the case of:

```
AddLanguage en .en
```

```
AddLanguage en-gb .en  
AddLanguage en-us .en
```

documents with the extension `.en` would be treated as being `en-us`.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

## See also

- [mod\\_negotiation](#)



<b>Description:</b>	Maps filename extensions to the filters that will process responses from the server
<b>Syntax:</b>	<code>AddOutputFilter <i>filter</i> [;<i>filter</i>...] <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	AddOutputFilter is only available in Apache 2.0.26 and later.

The `AddOutputFilter` directive maps the filename extension *extension* to the [filters](#) which will process responses from the server before they are sent to the client. This is in addition to any filters defined elsewhere, including [SetOutputFilter](#) and [AddOutputFilterByType](#) directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same *extension*.

For example, the following configuration will process all `.shtml` files for server-side includes and will then compress the output using [mod\\_deflate](#).

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content. Both the *filter* and *extension* arguments are case-insensitive, and the extension may be specified with or without a leading dot.

## See also

- [RemoveOutputFilter](#)

- SetOutputFilter



## AddType Directive

**Description:** Maps the given filename extensions onto the specified content type

**Syntax:** `AddType MIME-type extension [extension] ...`

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Status:** Base

**Module:** mod\_mime

The `AddType` directive maps the given filename extensions onto the specified content type. *MIME-type* is the MIME type to use for filenames containing *extension*. This mapping is added to any already in force, overriding any mappings that already exist for the same *extension*. This directive can be used to add mappings not listed in the MIME types file (see the `TypesConfig` directive).

### Example

```
AddType image/gif .gif
```

It is recommended that new MIME types be added using the `AddType` directive rather than changing the `TypesConfig` file.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

### See also

- [DefaultType](#)
- [ForceType](#)



## DefaultLanguage Directive

<b>Description:</b>	Sets all files in the given scope to the specified language
<b>Syntax:</b>	DefaultLanguage <i>MIME-lang</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **DefaultLanguage** directive tells Apache that all files in the directive's scope (e.g., all files covered by the current **<Directory>** container) that don't have an explicit language extension (such as `.fr` or `.de` as configured by **AddLanguage**) should be considered to be in the specified *MIME-lang* language. This allows entire directories to be marked as containing Dutch content, for instance, without having to rename each file. Note that unlike using extensions to specify languages, **DefaultLanguage** can only specify a single language.

If no **DefaultLanguage** directive is in force, and a file does not have any language extensions as configured by **AddLanguage**, then that file will be considered to have no language attribute.

### Example

```
DefaultLanguage en
```

### See also

- [mod\\_negotiation](#)



<b>Description:</b>	Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename
<b>Syntax:</b>	<code>ModMimeUsePathInfo On Off</code>
<b>Default:</b>	<code>ModMimeUsePathInfo Off</code>
<b>Context:</b>	directory
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_mime</code>
<b>Compatibility:</b>	Available in Apache 2.0.41 and later

The `ModMimeUsePathInfo` directive is used to combine the filename with the `path_info` URL component to apply `mod_mime`'s directives to the request. The default value is `Off` - therefore, the `path_info` component is ignored.

This directive is recommended when you have a virtual filesystem.

### Example

```
ModMimeUsePathInfo On
```

If you have a request for `/bar/foo.shtml` where `/bar` is a Location and `ModMimeUsePathInfo` is `On`, `mod_mime` will treat the incoming request as `/bar/foo.shtml` and directives like `AddOutputFilter INCLUDES ..shtml` will add the `INCLUDES` filter to the request. If `ModMimeUsePathInfo` is not set, the `INCLUDES` filter will not be added.

### See also

- [AcceptPathInfo](#)



<b>Description:</b>	The types of files that will be included when searching for a matching file with MultiViews
<b>Syntax:</b>	MultiViewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters]
<b>Default:</b>	MultiViewsMatch NegotiatedOnly
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	Available in Apache 2.0.26 and later.

**MultiViewsMatch** permits three different behaviors for [mod\\_negotiation](#)'s MultiViews feature. MultiViews allows a request for a file, e.g. `index.html`, to match any negotiated extensions following the base request, e.g. `index.html.en`, `index.html.fr`, or `index.html.gz`.

The `NegotiatedOnly` option provides that every extension following the base name must correlate to a recognized [mod\\_mime](#) extension for content negotiation, e.g. `Charset`, `Content-Type`, `Language`, or `Encoding`. This is the strictest implementation with the fewest unexpected side effects, and is the default behavior.

To include extensions associated with `Handlers` and/or `Filters`, set the **MultiViewsMatch** directive to either `Handlers`, `Filters`, or both option keywords. If all other factors are equal, the smallest file will be served, e.g. in deciding between `index.html.cgi` of 500 bytes and `index.html.pl` of 1000 bytes, the `.cgi` file would win in this example. Users of `.asis` files might prefer to use the `Handler` option, if `.asis` files are associated with the

asis-handler.

You may finally allow Any extensions to match, even if [mod\\_mime](#) doesn't recognize the extension. This was the behavior in Apache 1.3, and can cause unpredictable results, such as serving .old or .bak files the webmaster never expected to be served.

For example, the following configuration will allow handlers and filters to participate in Multiviews, but will exclude unknown files:

```
MultiviewsMatch Handlers Filters
```

## See also

- [Options](#)
- [mod\\_negotiation](#)



<b>Description:</b>	Removes any character set associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveCharset <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	RemoveCharset is only available in Apache 2.0.24 and later.

The `RemoveCharset` directive removes any character set associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

### Example

```
RemoveCharset .html .shtml
```



<b>Description:</b>	Removes any content encoding associations for a set of file extensions
<b>Syntax:</b>	RemoveEncoding <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveEncoding** directive removes any encoding associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

#### **/foo/.htaccess:**

```
AddEncoding x-gzip .gz
AddType text/plain .asc
<Files *.gz.asc>
    RemoveEncoding .gz
</Files>
```

This will cause `foo.gz` to be marked as being encoded with the gzip method, but `foo.gz.asc` as an unencoded plaintext file.

#### **Note**

**RemoveEncoding** directives are processed *after* any **AddEncoding** directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

The *extension* argument is case-insensitive, and can be specified

with or without a leading dot.



<b>Description:</b>	Removes any handler associations for a set of file extensions
<b>Syntax:</b>	RemoveHandler <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveHandler** directive removes any handler associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

**/foo/.htaccess:**

```
AddHandler server-parsed .html
```

**/foo/bar/.htaccess:**

```
RemoveHandler .html
```

This has the effect of returning .html files in the /foo/bar directory to being treated as normal files, rather than as candidates for parsing (see the [mod\\_include](#) module).

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.



<b>Description:</b>	Removes any input filter associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveInputFilter <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	RemoveInputFilter is only available in Apache 2.0.26 and later.

The `RemoveInputFilter` directive removes any input [filter](#) associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

## See also

- [AddInputFilter](#)
- [SetInputFilter](#)



## RemoveLanguage Directive

<b>Description:</b>	Removes any language associations for a set of file extensions
<b>Syntax:</b>	RemoveLanguage <i>extension</i> [ <i>extension</i> ] ...
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	RemoveLanguage is only available in Apache 2.0.24 and later.

The **RemoveLanguage** directive removes any language associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.



<b>Description:</b>	Removes any output filter associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveOutputFilter <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime
<b>Compatibility:</b>	RemoveOutputFilter is only available in Apache 2.0.26 and later.

The `RemoveOutputFilter` directive removes any output [filter](#) associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.

### Example

```
RemoveOutputFilter shtml
```

### See also

- [AddOutputFilter](#)



## RemoveType Directive

<b>Description:</b>	Removes any content type associations for a set of file extensions
<b>Syntax:</b>	<code>RemoveType <i>extension</i> [<i>extension</i>] ...</code>
<b>Context:</b>	virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_mime

The **RemoveType** directive removes any MIME type associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files. An example of its use might be:

### **/foo/.htaccess:**

```
RemoveType .cgi
```

This will remove any special handling of .cgi files in the /foo/ directory and any beneath it, causing the files to be treated as being of the **DefaultType**.

### **Note**

**RemoveType** directives are processed *after* any **AddType** directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

The *extension* argument is case-insensitive, and can be specified with or without a leading dot.



## TypesConfig Directive

<b>Description:</b>	The location of the <code>mime.types</code> file
<b>Syntax:</b>	<code>TypesConfig file-path</code>
<b>Default:</b>	<code>TypesConfig conf/mime.types</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	<code>mod_mime</code>

The `TypesConfig` directive sets the location of the MIME types configuration file. *File-path* is relative to the `ServerRoot`. This file sets the default list of mappings from filename extensions to content types. Most administrators use the provided `mime.types` file, which associates common filename extensions with IANA registered content types. The current list is maintained at <http://www.iana.org/assignments/media-types/index.html>. This simplifies the `httpd.conf` file by providing the majority of media-type definitions, and may be overridden by `AddType` directives as needed. You should not edit the `mime.types` file, because it may be replaced when you upgrade your server.

The file contains lines in the format of the arguments to an `AddType` directive:

```
MIME-type [extension] ...
```

The case of the extension does not matter. Blank lines, and lines beginning with a hash character (`#`) are ignored.

Please do **not** send requests to the Apache HTTP Server Project to add any new entries in the distributed `mime.types` file unless (1) they are already registered with IANA, and (2) they use widely accepted, non-conflicting filename extensions across platforms. `category/x-subtype` requests will be

automatically rejected, as will any new two-letter extensions as they will likely conflict later with the already crowded language and character set namespace.

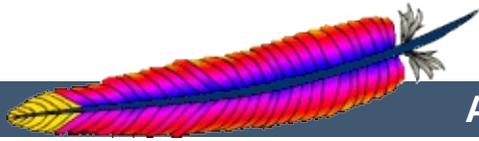
## See also

- [mod\\_mime\\_magic](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module `mod_mime_magic`

<b>Description:</b>	Determines the MIME type of a file by looking at a few bytes of its contents
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>mime_magic_module</code>
<b>Source File:</b>	<code>mod_mime_magic.c</code>

### Summary

This module determines the MIME type of files in the same way the Unix `file(1)` command works: it looks at the first few bytes of the file. It is intended as a "second line of defense" for cases that `mod_mime` can't resolve.

This module is derived from a free version of the `file(1)` command for Unix, which uses "magic numbers" and other hints from a file's contents to figure out what the contents are. This module is active only if the magic file is specified by the `MimeMagicFile` directive.



The contents of the file are plain ASCII text in 4-5 columns. Blank lines are allowed but ignored. Commented lines use a hash mark (#). The remaining lines are parsed for the following columns:

Column	Description																						
1	byte number to begin checking from ">" indicates a dependency upon the previous non-">" line																						
2	type of data to match <table border="1"> <tbody> <tr> <td>byte</td> <td>single character</td> </tr> <tr> <td>short</td> <td>machine-order 16-bit integer</td> </tr> <tr> <td>long</td> <td>machine-order 32-bit integer</td> </tr> <tr> <td>string</td> <td>arbitrary-length string</td> </tr> <tr> <td>date</td> <td>long integer date (seconds since Unix epoch/1970)</td> </tr> <tr> <td>beshort</td> <td>big-endian 16-bit integer</td> </tr> <tr> <td>belong</td> <td>big-endian 32-bit integer</td> </tr> <tr> <td>bedate</td> <td>big-endian 32-bit integer date</td> </tr> <tr> <td>leshort</td> <td>little-endian 16-bit integer</td> </tr> <tr> <td>lelong</td> <td>little-endian 32-bit integer</td> </tr> <tr> <td>ledate</td> <td>little-endian 32-bit integer date</td> </tr> </tbody> </table>	byte	single character	short	machine-order 16-bit integer	long	machine-order 32-bit integer	string	arbitrary-length string	date	long integer date (seconds since Unix epoch/1970)	beshort	big-endian 16-bit integer	belong	big-endian 32-bit integer	bedate	big-endian 32-bit integer date	leshort	little-endian 16-bit integer	lelong	little-endian 32-bit integer	ledate	little-endian 32-bit integer date
byte	single character																						
short	machine-order 16-bit integer																						
long	machine-order 32-bit integer																						
string	arbitrary-length string																						
date	long integer date (seconds since Unix epoch/1970)																						
beshort	big-endian 16-bit integer																						
belong	big-endian 32-bit integer																						
bedate	big-endian 32-bit integer date																						
leshort	little-endian 16-bit integer																						
lelong	little-endian 32-bit integer																						
ledate	little-endian 32-bit integer date																						
3	contents of data to match																						
4	MIME type if matched																						
5	MIME encoding if matched (optional)																						

For example, the following magic file lines would recognize some audio formats:

```
# Sun/NeXT audio data
```

```

0      string      .snd
>12   belong      1      audio/basic
>12   belong      2      audio/basic
>12   belong      3      audio/basic
>12   belong      4      audio/basic
>12   belong      5      audio/basic
>12   belong      6      audio/basic
>12   belong      7      audio/basic
>12   belong      23     audio/x-adpcm

```

Or these would recognize the difference between \* .doc files containing Microsoft Word or FrameMaker documents. (These are incompatible file formats which use the same file suffix.)

```

# Frame
0  string  \<MakerFile      application/x-frame
0  string  \<MIFFfile    application/x-frame
0  string  \<MakerDictionary application/x-frame
0  string  \<MakerScreenFon application/x-frame
0  string  \<MML      application/x-frame
0  string  \<Book     application/x-frame
0  string  \<Maker   application/x-frame

# MS-Word
0  string  \376\067\0\043      application/msword
0  string  \320\317\021\340\241\261 application/msword
0  string  \333\245-\0\0\0 application/msword

```

An optional MIME encoding can be included as a fifth column. For example, this can recognize gzipped files and set the encoding for them.

```

# gzip (GNU zip, not to be confused with
#      [Info-ZIP/PKWARE] zip archiver)

0  string  \037\213  application/octet-stream  x-gzip

```



This module is not for every system. If your system is barely keeping up with its load or if you're performing a web server benchmark, you may not want to enable this because the processing is not free.

However, an effort was made to improve the performance of the original `file(1)` code to make it fit in a busy web server. It was designed for a server where there are thousands of users who publish their own documents. This is probably very common on intranets. Many times, it's helpful if the server can make more intelligent decisions about a file's contents than the file name allows ...even if just to reduce the "why doesn't my page work" calls when users improperly name their own files. You have to decide if the extra work suits your environment.



The following notes apply to the `mod_mime_magic` module and are included here for compliance with contributors' copyright restrictions that require their acknowledgment.

`mod_mime_magic`: MIME type lookup via file magic numbers  
Copyright (c) 1996-1997 Cisco Systems, Inc.

This software was submitted by Cisco Systems to the Apache Group in July 1997. Future revisions and derivatives of this source code must acknowledge Cisco Systems as the original contributor of this module. All other licensing and usage conditions are those of the Apache Group.

Some of this code is derived from the free version of the file command originally posted to comp.sources.unix. Copyright info for that program is included below as required.

- Copyright (c) Ian F. Darwin, 1987. Written by Ian F. Darwin.

This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.

Permission is granted to anyone to use this software for any purpose on any computer system, and to alter it and redistribute it freely, subject to the following restrictions:

1. The author is not responsible for the consequences of use of this software, no matter how awful, even if they arise from flaws in it.
2. The origin of this software must not be misrepresented, either by explicit claim or by omission. Since few users ever read sources, credits must appear in the documentation.
3. Altered versions must be plainly marked as such, and must

not be misrepresented as being the original software. Since few users ever read sources, credits must appear in the documentation.

4. This notice may not be removed or altered.

For compliance with Mr Darwin's terms: this has been very significantly modified from the free "file" command.

- all-in-one file for compilation convenience when moving from one version of Apache to the next.
- Memory allocation is done through the Apache API's pool structure.
- All functions have had necessary Apache API request or server structures passed to them where necessary to call other Apache API routines. (*i.e.*, usually for logging, files, or memory allocation in itself or a called function.)
- struct magic has been converted from an array to a single-ended linked list because it only grows one record at a time, it's only accessed sequentially, and the Apache API has no equivalent of `realloc()`.
- Functions have been changed to get their parameters from the server configuration instead of globals. (It should be reentrant now but has not been tested in a threaded environment.)
- Places where it used to print results to stdout now saves them in a list where they're used to set the MIME type in the Apache request record.
- Command-line flags have been removed since they will never be used here.

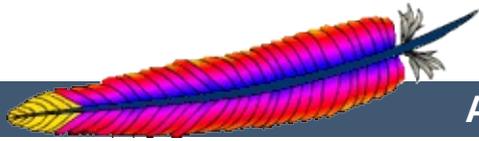


<b>Description:</b>	Enable MIME-type determination based on file contents using the specified magic file
<b>Syntax:</b>	MimeMagicFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_mime_magic

The `MimeMagicFile` directive can be used to enable this module, the default file is distributed at `conf/magic`. Non-rooted paths are relative to the `ServerRoot`. Virtual hosts will use the same file as the main server unless a more specific setting is used, in which case the more specific setting overrides the main server's file.

### Example

```
MimeMagicFile conf/magic
```



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_negotiation

<b>Description:</b>	Provides for <a href="#">content negotiation</a>
<b>Status:</b>	Base
<b>Module Identifier:</b>	negotiation_module
<b>Source File:</b>	mod_negotiation.c

## Summary

Content negotiation, or more accurately content selection, is the selection of the document that best matches the clients capabilities, from one of several available documents. There are two implementations of this.

- A type map (a file with the handler type-map) which explicitly lists the files containing the variants.
- A MultiViews search (enabled by the MultiViews [Options](#)), where the server does an implicit filename pattern match, and choose from amongst the results.

## See also

[Options](#)

[mod\\_mime](#)

[Content Negotiation](#)

[Environment Variables](#)



## Type Maps

A type map has a format similar to RFC822 mail headers. It contains document descriptions separated by blank lines, with lines beginning with a hash character ('#') treated as comments. A document description consists of several header records; records may be continued on multiple lines if the continuation lines start with spaces. The leading space will be deleted and the lines concatenated. A header record consists of a keyword name, which always ends in a colon, followed by a value. Whitespace is allowed between the header name and value, and between the tokens of value. The headers allowed are:

### **Content - Encoding:**

The encoding of the file. Apache only recognizes encodings that are defined by an [AddEncoding](#) directive. This normally includes the encodings x-compress for compress'd files, and x-gzip for gzip'd files. The x- prefix is ignored for encoding comparisons.

### **Content - Language:**

The language(s) of the variant, as an Internet standard language tag ([RFC 1766](#)). An example is en, meaning English. If the variant contains more than one language, they are separated by a comma.

### **Content - Length:**

The length of the file, in bytes. If this header is not present, then the actual length of the file is used.

### **Content - Type:**

The MIME media type of the document, with optional parameters. Parameters are separated from the media type and from one another by a semi-colon, with a syntax of name=value. Common parameters include:

#### **level**

an integer specifying the version of the media type. For

text/html this defaults to 2, otherwise 0.

## **qs**

a floating-point number with a value in the range 0.0 to 1.0, indicating the relative 'quality' of this variant compared to the other available variants, independent of the client's capabilities. For example, a jpeg file is usually of higher source quality than an ascii file if it is attempting to represent a photograph. However, if the resource being represented is ascii art, then an ascii file would have a higher source quality than a jpeg file. All qs values are therefore specific to a given resource.

### **Example**

```
Content-Type: image/jpeg; qs=0.8
```

## **URI :**

uri of the file containing the variant (of the given media type, encoded with the given content encoding). These are interpreted as URLs relative to the map file; they must be on the same server (!), and they must refer to files to which the client would be granted access if they were to be requested directly.

## **Body :**

New in Apache 2.0, the actual content of the resource may be included in the type-map file using the Body header. This header must contain a string that designates a delimiter for the body content. Then all following lines in the type map file will be considered part of the resource body until the delimiter string is found.

### **Example:**

```
Body: ----xyz----  
<html>
```

```
<body>  
<p>Content of the page.</p>  
</body>  
</html>  
-----xyz-----
```



## MULTIVIEWS

A MultiViews search is enabled by the MultiViews [Options](#). If the server receives a request for `/some/dir/foo` and `/some/dir/foo` does *not* exist, then the server reads the directory looking for all files named `foo.*`, and effectively fakes up a type map which names all those files, assigning them the same media types and content-encodings it would have if the client had asked for one of them by name. It then chooses the best match to the client's requirements, and returns that document.

The [MultiViewsMatch](#) directive configures whether Apache will consider files that do not have content negotiation meta-information assigned to them when choosing files.



## CacheNegotiatedDocs Directive

<b>Description:</b>	Allows content-negotiated documents to be cached by proxy servers
<b>Syntax:</b>	CacheNegotiatedDocs On Off
<b>Default:</b>	CacheNegotiatedDocs Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_negotiation
<b>Compatibility:</b>	The syntax changed in version 2.0.

If set, this directive allows content-negotiated documents to be cached by proxy servers. This could mean that clients behind those proxys could retrieve versions of the documents that are not the best match for their abilities, but it will make caching more efficient.

This directive only applies to requests which come from HTTP/1.0 browsers. HTTP/1.1 provides much better control over the caching of negotiated documents, and this directive has no effect in responses to HTTP/1.1 requests.

Prior to version 2.0, **CacheNegotiatedDocs** did not take an argument; it was turned on by the presence of the directive by itself.



## ForceLanguagePriority Directive

<b>Description:</b>	Action to take if a single acceptable document is not found
<b>Syntax:</b>	ForceLanguagePriority None   Prefer   Fallback [Prefer   Fallback]
<b>Default:</b>	ForceLanguagePriority Prefer
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_negotiation
<b>Compatibility:</b>	Available in version 2.0.30 and later

The `ForceLanguagePriority` directive uses the given `LanguagePriority` to satisfy negotiation where the server could otherwise not return a single matching document.

`ForceLanguagePriority Prefer` uses `LanguagePriority` to serve a one valid result, rather than returning an HTTP result 300 (MULTIPLE CHOICES) when there are several equally valid choices. If the directives below were given, and the user's `Accept-Language` header assigned `en` and `de` each as quality `.500` (equally acceptable) then the first matching variant, `en`, will be served.

```
LanguagePriority en fr de
ForceLanguagePriority Prefer
```

`ForceLanguagePriority Fallback` uses `LanguagePriority` to serve a valid result, rather than returning an HTTP result 406 (NOT ACCEPTABLE). If the directives below were given, and the user's `Accept-Language` only permitted an `es` language response, but such a variant isn't found, then the first

variant from the [LanguagePriority](#) list below will be served.

```
LanguagePriority en fr de  
ForceLanguagePriority Fallback
```

Both options, `Prefer` and `Fallback`, may be specified, so either the first matching variant from [LanguagePriority](#) will be served if more than one variant is acceptable, or first available document will be served if none of the variants matched the client's acceptable list of languages.

## See also

- [AddLanguage](#)



<b>Description:</b>	The precedence of language variants for cases where the client does not express a preference
<b>Syntax:</b>	LanguagePriority <i>MIME-lang</i> [ <i>MIME-lang</i> ] ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_negotiation

The `LanguagePriority` sets the precedence of language variants for the case where the client does not express a preference, when handling a MultiViews request. The list of *MIME-lang* are in order of decreasing preference.

### Example:

```
LanguagePriority en fr de
```

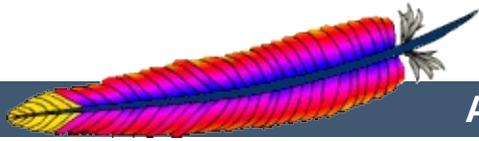
For a request for `foo.html`, where `foo.html.fr` and `foo.html.de` both existed, but the browser did not express a language preference, then `foo.html.fr` would be returned.

Note that this directive only has an effect if a 'best' language cannot be determined by any other means or the `ForceLanguagePriority` directive is not `None`. In general, the client determines the language preference, not the server.

### See also

- [AddLanguage](#)

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_nw\_ssl

<b>Description:</b>	Enable SSL encryption for NetWare
<b>Status:</b>	Base
<b>Module Identifier:</b>	nwssl_module
<b>Source File:</b>	mod_nw_ssl.c
<b>Compatibility:</b>	NetWare only

## Summary

This module enables SSL encryption for a specified port. It takes advantage of the SSL encryption functionality that is built into the NetWare operating system.



<b>Description:</b>	List of additional client certificates
<b>Syntax:</b>	NWSSLTrustedCerts <i>filename</i> [ <i>filename</i> ] ...
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Specifies a list of client certificate files (DER format) that are used when creating a proxied SSL connection. Each client certificate used by a server must be listed separately in its own .der file.



<b>Description:</b>	Allows a connection to be upgraded to an SSL connection upon request
<b>Syntax:</b>	NWSSLUpgradeable [ <i>IP-address:</i> ]portnumber
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Allow a connection that was created on the specified address and/or port to be upgraded to an SSL connection upon request from the client. The address and/or port must have already be defined previously with a [Listen](#) directive.



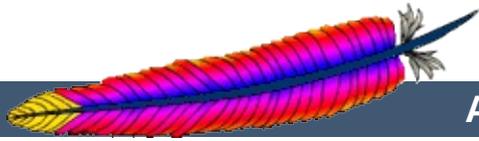
<b>Description:</b>	Enables SSL encryption for the specified port
<b>Syntax:</b>	<code>SecureListen [IP-address:]portnumber Certificate-Name [MUTUAL]</code>
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_nw_ssl

Specifies the port and the eDirectory based certificate name that will be used to enable SSL encryption. An optional third parameter also enables mutual authentication.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_proxy`

<b>Description:</b>	HTTP/1.1 proxy/gateway server
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_module</code>
<b>Source File:</b>	<code>mod_proxy.c</code>

## Summary

### Warning

Do not enable proxying with [ProxyRequests](#) until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

This module implements a proxy/gateway for Apache. It implements proxying capability for FTP, CONNECT (for SSL), HTTP/0.9, HTTP/1.0, and HTTP/1.1. The module can be configured to connect to other proxy modules for these and other protocols.

Apache's proxy features are divided into several modules in addition to `mod_proxy`: `mod_proxy_http`, `mod_proxy_ftp` and `mod_proxy_connect`. Thus, if you want to use one or more of the particular proxy functions, load `mod_proxy` and the appropriate module(s) into the server (either statically at compile-time or dynamically via the [LoadModule](#) directive).

In addition, extended features are provided by other modules. Caching is provided by `mod_cache` and related modules. The ability to contact remote servers using the SSL/TLS protocol is provided by the `SSLProxy*` directives of `mod_ssl`. These additional modules will need to be loaded and configured to take advantage of these features.

## See also

[mod\\_cache](#)

[mod\\_proxy\\_http](#)

[mod\\_proxy\\_ftp](#)

[mod\\_proxy\\_connect](#)

[mod\\_ssl](#)



---

Apache can be configured in both a *forward* and *reverse* proxy mode.

An ordinary *forward proxy* is an intermediate server that sits between the client and the *origin server*. In order to get content from the origin server, the client sends a request to the proxy naming the origin server as the target and the proxy then requests the content from the origin server and returns it to the client. The client must be specially configured to use the forward proxy to access other sites.

A typical usage of a forward proxy is to provide Internet access to internal clients that are otherwise restricted by a firewall. The forward proxy can also use caching (as provided by [mod\\_cache](#)) to reduce network usage.

The forward proxy is activated using the [ProxyRequests](#) directive. Because forward proxys allow clients to access arbitrary sites through your server and to hide their true origin, it is essential that you [secure your server](#) so that only authorized clients can access the proxy before activating a forward proxy.

A *reverse proxy*, by contrast, appears to the client just like an ordinary web server. No special configuration on the client is necessary. The client makes ordinary requests for content in the name-space of the reverse proxy. The reverse proxy then decides where to send those requests, and returns the content as if it was itself the origin.

A typical usage of a reverse proxy is to provide Internet users access to a server that is behind a firewall. Reverse proxies can also be used to balance load among several back-end servers, or to provide caching for a slower back-end server. In addition, reverse proxies can be used simply to bring several servers into

the same URL space.

A reverse proxy is activated using the [ProxyPass](#) directive or the [P] flag to the [RewriteRule](#) directive. It is **not** necessary to turn [ProxyRequests](#) on in order to configure a reverse proxy.



## Basic Examples

The examples below are only a very basic idea to help you get started. Please read the documentation on the individual directives.

In addition, if you wish to have caching enabled, consult the documentation from [mod\\_cache](#).

### Forward Proxy

```
ProxyRequests On
ProxyVia On
```

```
<Proxy *>
  Order deny,allow
  Deny from all
  Allow from internal.example.com
</Proxy>
```

### Reverse Proxy

```
ProxyRequests Off
```

```
<Proxy *>
  Order deny,allow
  Allow from all
</Proxy>
```

```
ProxyPass /foo http://foo.example.com/bar
ProxyPassReverse /foo http://foo.example.com/bar
```



## Limiting access to your proxy

You can control who can access your proxy via the [<Proxy>](#) control block as in the following example:

```
<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from 192.168.0
</Proxy>
```

For more information on access control directives, see [mod\\_access](#).

Strictly limiting access is essential if you are using a forward proxy (using the [ProxyRequests](#) directive). Otherwise, your server can be used by any client to access arbitrary hosts while hiding his or her true identity. This is dangerous both for your network and for the Internet at large. When using a reverse proxy (using the [ProxyPass](#) directive with `ProxyRequests Off`), access control is less critical because clients can only contact the hosts that you have specifically configured.



## Why doesn't file type xxx download via FTP?

You probably don't have that particular file type defined as `application/octet-stream` in your proxy's `mime.types` configuration file. A useful line can be

```
application/octet-stream  bin dms lha lzh exe class tgz taz
```

## How can I force an FTP ASCII download of File xxx?

In the rare situation where you must download a specific file using the FTP ASCII transfer method (while the default transfer is in binary mode), you can override `mod_proxy`'s default by suffixing the request with `;type=a` to force an ASCII transfer. (FTP Directory listings are always executed in ASCII mode, however.)

## How can I access FTP files outside of my home directory?

An FTP URI is interpreted relative to the home directory of the user who is logging in. Alas, to reach higher directory levels you cannot use `../`, as the dots are interpreted by the browser and not actually sent to the FTP server. To address this problem, the so called *Squid %2f hack* was implemented in the Apache FTP proxy; it is a solution which is also used by other popular proxy servers like the [Squid Proxy Cache](#). By prepending `/%2f` to the path of your request, you can make such a proxy change the FTP starting directory to `/` (instead of the home directory). For example, to retrieve the file `/etc/motd`, you would use the URL:

```
ftp://user@host/%2f/etc/motd
```

## How can I hide the FTP cleartext password in my browser's URL line?

To log in to an FTP server by username and password, Apache uses different strategies. In absence of a user name and password in the URL altogether, Apache sends an anonymous login to the FTP server, *i.e.*,

```
user: anonymous  
password: apache_proxy@
```

This works for all popular FTP servers which are configured for anonymous access.

For a personal login with a specific username, you can embed the user name into the URL, like in:

```
ftp://username@host/myfile
```

If the FTP server asks for a password when given this username (which it should), then Apache will reply with a 401 (Authorization required) response, which causes the Browser to pop up the username/password dialog. Upon entering the password, the connection attempt is retried, and if successful, the requested resource is presented. The advantage of this procedure is that your browser does not display the password in cleartext (which it would if you had used

```
ftp://username:password@host/myfile
```

in the first place).

### Note

The password which is transmitted in such a way is not encrypted on its way. It travels between your browser and the

Apache proxy server in a base64-encoded cleartext string, and between the Apache proxy and the FTP server as plaintext. You should therefore think twice before accessing your FTP server via HTTP (or before accessing your personal files via FTP at all!) When using unsecure channels, an eavesdropper might intercept your password on its way.



If you're using the [ProxyBlock](#) directive, hostnames' IP addresses are looked up and cached during startup for later match test. This may take a few seconds (or more) depending on the speed with which the hostname lookups occur.



## Intranet Proxy

An Apache proxy server situated in an intranet needs to forward external requests through the company's firewall (for this, configure the [ProxyRemote](#) directive to forward the respective *scheme* to the firewall proxy). However, when it has to access resources within the intranet, it can bypass the firewall when accessing hosts. The [NoProxy](#) directive is useful for specifying which hosts belong to the intranet and should be accessed directly.

Users within an intranet tend to omit the local domain name from their WWW requests, thus requesting "http://somehost/" instead of `http://somehost.example.com/`. Some commercial proxy servers let them get away with this and simply serve the request, implying a configured local domain. When the [ProxyDomain](#) directive is used and the server is [configured for proxy service](#), Apache can return a redirect response and send the client to the correct, fully qualified, server address. This is the preferred method since the user's bookmark files will then contain fully qualified hosts.



## Proxy Requirements

For circumstances where you have a application server which doesn't implement keepalives or HTTP/1.1 properly, there are 2 environment variables which when set send a HTTP/1.0 with no keepalive. These are set via the [SetEnv](#) directive.

These are the `force-proxy-request-1.0` and `proxy-nokeepalive` notes.

```
<Location /buggyappserver/>
  ProxyPass http://buggyappserver:7001/foo/
  SetEnv force-proxy-request-1.0 1
  SetEnv proxy-nokeepalive 1
</Location>
```



<b>Description:</b>	Ports that are allowed to CONNECT through the proxy
<b>Syntax:</b>	AllowCONNECT <i>port</i> [ <i>port</i> ] ...
<b>Default:</b>	AllowCONNECT 443 563
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `AllowCONNECT` directive specifies a list of port numbers to which the proxy CONNECT method may connect. Today's browsers use this method when a `https` connection is requested and proxy tunneling over HTTP is in effect.

By default, only the default `https` port (443) and the default `snews` port (563) are enabled. Use the `AllowCONNECT` directive to override this default and allow connections to the listed ports only.

Note that you'll need to have `mod_proxy_connect` present in the server in order to get the support for the CONNECT at all.



<b>Description:</b>	Hosts, domains, or networks that will be connected to directly
<b>Syntax:</b>	NoProxy <i>host</i> [ <i>host</i> ] ...
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is only useful for Apache proxy servers within intranets. The **NoProxy** directive specifies a list of subnets, IP addresses, hosts and/or domains, separated by spaces. A request to a host which matches one or more of these is always served directly, without forwarding to the configured **ProxyRemote** proxy server(s).

### Example

```
ProxyRemote * http://firewall.example.com:81
NoProxy .example.com 192.168.112.0/21
```

The *host* arguments to the **NoProxy** directive are one of the following type list:

### Domain

A *Domain* is a partially qualified DNS domain name, preceded by a period. It represents a list of hosts which logically belong to the same DNS domain or zone (*i.e.*, the suffixes of the hostnames are all ending in *Domain*).

### Examples

```
.com .apache.org.
```

To distinguish *Domains* from **Hostnames** (both syntactically and semantically; a DNS domain can have a DNS A record,

too!), *Domains* are always written with a leading period.

### Note

Domain name comparisons are done without regard to the case, and *Domains* are always assumed to be anchored in the root of the DNS tree, therefore two domains `.MyDomain.com` and `.mydomain.com`. (note the trailing period) are considered equal. Since a domain comparison does not involve a DNS lookup, it is much more efficient than subnet comparison.

### SubNet

A *SubNet* is a partially qualified internet address in numeric (dotted quad) form, optionally followed by a slash and the netmask, specified as the number of significant bits in the *SubNet*. It is used to represent a subnet of hosts which can be reached over a common network interface. In the absence of the explicit net mask it is assumed that omitted (or zero valued) trailing digits specify the mask. (In this case, the netmask can only be multiples of 8 bits wide.) Examples:

#### **192 . 168 or 192 . 168 . 0 . 0**

the subnet 192.168.0.0 with an implied netmask of 16 valid bits (sometimes used in the netmask form 255 . 255 . 0 . 0)

#### **192 . 168 . 112 . 0 / 21**

the subnet 192 . 168 . 112 . 0 / 21 with a netmask of 21 valid bits (also used in the form 255.255.248.0)

As a degenerate case, a *SubNet* with 32 valid bits is the equivalent to an [IPAddr](#), while a *SubNet* with zero valid bits (e.g., 0.0.0.0/0) is the same as the constant `_Default_`, matching any IP address.

## ***IPAddr***

A *IPAddr* represents a fully qualified internet address in numeric (dotted quad) form. Usually, this address represents a host, but there need not necessarily be a DNS domain name connected with the address.

### **Example**

192.168.123.7

### **Note**

An *IPAddr* does not need to be resolved by the DNS system, so it can result in more effective apache performance.

## ***Hostname***

A *Hostname* is a fully qualified DNS domain name which can be resolved to one or more [IPAddrs](#) via the DNS domain name service. It represents a logical host (in contrast to [Domains](#), see above) and must be resolvable to at least one [IPAddr](#) (or often to a list of hosts with different [IPAddrs](#)).

### **Examples**

prep.ai.mit.edu  
www.apache.org

### **Note**

In many situations, it is more effective to specify an [IPAddr](#) in place of a *Hostname* since a DNS lookup can be avoided. Name resolution in Apache can take a remarkable deal of time when the connection to the name server uses a slow PPP link.

*Hostname* comparisons are done without regard to the

case, and *Hostnames* are always assumed to be anchored in the root of the DNS tree, therefore two hosts `WWW.MyDomain.com` and `www.mydomain.com.` (note the trailing period) are considered equal.

## See also

- [DNS Issues](#)



<b>Description:</b>	Container for directives applied to proxied resources
<b>Syntax:</b>	<code>&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

Directives placed in `<Proxy>` sections apply only to matching proxied content. Shell-style wildcards are allowed.

For example, the following will allow only hosts in `yournetwork.example.com` to access content via your proxy server:

```
<Proxy *>
  Order Deny,Allow
  Deny from all
  Allow from yournetwork.example.com
</Proxy>
```

The following example will process all files in the `foo` directory of `example.com` through the `INCLUDES` filter when they are sent through the proxy server:

```
<Proxy http://example.com/foo/*>
  SetOutputFilter INCLUDES
</Proxy>
```



## ProxyBadHeader Directive

<b>Description:</b>	Determines how to handle bad header lines in a response
<b>Syntax:</b>	ProxyBadHeader IsError   Ignore   StartBody
<b>Default:</b>	ProxyBadHeader IsError
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.44 and later

The **ProxyBadHeader** directive determines the behaviour of **mod\_proxy** if it receives syntactically invalid header lines (*i.e.* containing no colon). The following arguments are possible:

### **IsError**

Abort the request and end up with a 502 (Bad Gateway) response. This is the default behaviour.

### **Ignore**

Treat bad header lines as if they weren't sent.

### **StartBody**

When receiving the first bad header line, finish reading the headers and treat the remainder as body. This helps to work around buggy backend servers which forget to insert an empty line between the headers and the body.



**Description:** Words, hosts, or domains that are banned from being proxied

**Syntax:** ProxyBlock \* |*word*|*host*|*domain*  
[*word*|*host*|*domain*] ...

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_proxy

The **ProxyBlock** directive specifies a list of words, hosts and/or domains, separated by spaces. HTTP, HTTPS, and FTP document requests to sites whose names contain matched words, hosts or domains are *blocked* by the proxy server. The proxy module will also attempt to determine IP addresses of list items which may be hostnames during startup, and cache them for match test as well. That may slow down the startup time of the server.

### Example

```
ProxyBlock joes-garage.com some-host.co.uk  
rocky.wotsamattau.edu
```

rocky.wotsamattau.edu would also be matched if referenced by IP address.

Note that wotsamattau would also be sufficient to match wotsamattau.edu.

Note also that

```
ProxyBlock *
```

blocks connections to all sites.



## ProxyDomain Directive

<b>Description:</b>	Default domain name for proxied requests
<b>Syntax:</b>	ProxyDomain <i>Domain</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive is only useful for Apache proxy servers within intranets. The `ProxyDomain` directive specifies the default domain which the apache proxy server will belong to. If a request to a host without a domain name is encountered, a redirection response to the same host with the configured *Domain* appended will be generated.

**Example**

```
ProxyRemote * http://firewall.example.com:81
NoProxy .example.com 192.168.112.0/21
ProxyDomain .example.com
```



<b>Description:</b>	Override error pages for proxied content
<b>Syntax:</b>	ProxyErrorOverride On Off
<b>Default:</b>	ProxyErrorOverride Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in version 2.0 and later

This directive is useful for reverse-proxy setups, where you want to have a common look and feel on the error pages seen by the end user. This also allows for included files (via mod\_include's SSI) to get the error code and act accordingly (default behavior would display the error page of the proxied server, turning this on shows the SSI Error message).



## ProxyFtpDirCharset Directive

<b>Description:</b>	Define the character set for proxied FTP listings
<b>Syntax:</b>	ProxyFtpDirCharset <i>character set</i>
<b>Default:</b>	ProxyFtpDirCharset ISO-8859-1
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.62 and later

The `ProxyFtpDirCharset` directive defines the character set to be set for FTP directory listings in HTML generated by [mod\\_proxy\\_ftp](#).



<b>Description:</b>	Determine size of internal data throughput buffer
<b>Syntax:</b>	ProxyIOBufferSize <i>bytes</i>
<b>Default:</b>	ProxyIOBufferSize 8192
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `ProxyIOBufferSize` directive adjusts the size of the internal buffer, which is used as a scratchpad for the data between input and output. The size must be less or equal 8192.

In almost every case there's no reason to change that value.



## ProxyMatch Directive

<b>Description:</b>	Container for directives applied to regular-expression-matched proxied resources
<b>Syntax:</b>	<code>&lt;ProxyMatch regex&gt; ...&lt;/ProxyMatch&gt;</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `<ProxyMatch>` directive is identical to the `<Proxy>` directive, except it matches URLs using regular expressions.



## ProxyMaxForwards Directive

<b>Description:</b>	Maximum number of proxies that a request can be forwarded through
<b>Syntax:</b>	ProxyMaxForwards <i>number</i>
<b>Default:</b>	ProxyMaxForwards 10
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0 and later

The **ProxyMaxForwards** directive specifies the maximum number of proxies through which a request may pass, if there's no Max-Forwards header supplied with the request. This is set to prevent infinite proxy loops, or a DoS attack.

### Example

```
ProxyMaxForwards 15
```



## ProxyPass Directive

<b>Description:</b>	Maps remote servers into the local server URL-space
<b>Syntax:</b>	ProxyPass [ <i>path</i> ] !  <i>url</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. *path* is the name of a local virtual path; *url* is a partial URL for the remote server and cannot include a query string.

Suppose the local server has address `http://example.com/`; then

```
ProxyPass /mirror/foo/ http://backend.example.com/
```

will cause a local request for `http://example.com/mirror/foo/bar` to be internally converted into a proxy request to `http://backend.example.com/bar`.

The `!` directive is useful in situations where you don't want to reverse-proxy a subdirectory, e.g.

```
ProxyPass /mirror/foo/i !  
ProxyPass /mirror/foo http://backend.example.com
```

will proxy all requests to `/mirror/foo` to `backend.example.com` *except* requests made to `/mirror/foo/i`.

## Note

Order is important. you need to put the exclusions *before* the general proxypass directive.

When used inside a [<Location>](#) section, the first argument is omitted and the local directory is obtained from the [<Location>](#).

The [ProxyRequests](#) directive should usually be set **off** when using [ProxyPass](#).

If you require a more flexible reverse-proxy configuration, see the [RewriteRule](#) directive with the [P] flag.



<b>Description:</b>	Adjusts the URL in HTTP response headers sent from a reverse proxied server
<b>Syntax:</b>	ProxyPassReverse [ <i>path</i> ] <i>url</i>
<b>Context:</b>	server config, virtual host, directory
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This directive lets Apache adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache is used as a reverse proxy to avoid by-passing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy.

Only the HTTP response headers specifically mentioned above will be rewritten. Apache will not rewrite other response headers, nor will it rewrite URL references inside HTML pages. This means that if the proxied content contains absolute URL references, they will by-pass the proxy. A third-party module that will look inside the HTML and rewrite URL references is Nick Kew's [mod\\_proxy\\_html](#).

*path* is the name of a local virtual path. *url* is a partial URL for the remote server - the same way they are used for the [ProxyPass](#) directive.

For example, suppose the local server has address `http://example.com/`; then

```
ProxyPass /mirror/foo/ http://backend.example.com/  
ProxyPassReverse /mirror/foo/ http://backend.example.com/
```

will not only cause a local request for the `http://example.com/mirror/foo/bar` to be internally

converted into a proxy request to `http://backend.example.com/bar` (the functionality `ProxyPass` provides here). It also takes care of redirects the server `backend.example.com` sends: when `http://backend.example.com/bar` is redirected by him to `http://backend.example.com/quux` Apache adjusts this to `http://example.com/mirror/foo/quux` before forwarding the HTTP redirect response to the client. Note that the hostname used for constructing the URL is chosen in respect to the setting of the `UseCanonicalName` directive.

Note that this `ProxyPassReverse` directive can also be used in conjunction with the proxy pass-through feature (`RewriteRule . . . [P]`) from `mod_rewrite` because its doesn't depend on a corresponding `ProxyPass` directive.

When used inside a `<Location>` section, the first argument is omitted and the local directory is obtained from the `<Location>`.



## ProxyPreserveHost Directive

<b>Description:</b>	Use incoming Host HTTP request header for proxy request
<b>Syntax:</b>	ProxyPreserveHost On Off
<b>Default:</b>	ProxyPreserveHost Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.31 and later.

When enabled, this option will pass the Host: line from the incoming request to the proxied host, instead of the hostname specified in the proxypass line.

This option should normally be turned Off. It is mostly useful in special configurations like proxied mass name-based virtual hosting, where the original Host header needs to be evaluated by the backend server.



**Description:** Network buffer size for proxied HTTP and FTP connections

**Syntax:** ProxyReceiveBufferSize *bytes*

**Default:** ProxyReceiveBufferSize 0

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_proxy

The `ProxyReceiveBufferSize` directive specifies an explicit (TCP/IP) network buffer size for proxied HTTP and FTP connections, for increased throughput. It has to be greater than 512 or set to 0 to indicate that the system's default buffer size should be used.

### Example

```
ProxyReceiveBufferSize 2048
```



## ProxyRemote Directive

<b>Description:</b>	Remote proxy used to handle certain requests
<b>Syntax:</b>	ProxyRemote <i>match remote-server</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This defines remote proxies to this proxy. *match* is either the name of a URL-scheme that the remote server supports, or a partial URL for which the remote server should be used, or \* to indicate the server should be contacted for all requests. *remote-server* is a partial URL for the remote server. Syntax:

```
remote-server = scheme://hostname[:port]
```

*scheme* is effectively the protocol that should be used to communicate with the remote server; only http is supported by this module.

**Example**

```
ProxyRemote http://goodguys.com/ http://mirrorguys.com:8000  
ProxyRemote * http://cleversite.com  
ProxyRemote ftp http://ftpproxy.mydomain.com:8080
```

In the last example, the proxy will forward FTP requests, encapsulated as yet another HTTP proxy request, to another proxy which can handle them.

This option also supports reverse proxy configuration - a backend webserver can be embedded within a virtualhost URL space even if that server is hidden by another forward proxy.



## ProxyRemoteMatch Directive

<b>Description:</b>	Remote proxy used to handle requests matched by regular expressions
<b>Syntax:</b>	<code>ProxyRemoteMatch <i>regex remote-server</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

The `ProxyRemoteMatch` is identical to the `ProxyRemote` directive, except the first argument is a regular expression match against the requested URL.



## ProxyRequests Directive

<b>Description:</b>	Enables forward (standard) proxy requests
<b>Syntax:</b>	ProxyRequests On Off
<b>Default:</b>	ProxyRequests Off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy

This allows or prevents Apache from functioning as a forward proxy server. (Setting ProxyRequests to Off does not disable use of the [ProxyPass](#) directive.)

In a typical reverse proxy configuration, this option should be set to Off.

In order to get the functionality of proxying HTTP or FTP sites, you need also [mod\\_proxy\\_http](#) or [mod\\_proxy\\_ftp](#) (or both) present in the server.

### Warning

Do not enable proxying with [ProxyRequests](#) until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.



<b>Description:</b>	Network timeout for proxied requests
<b>Syntax:</b>	ProxyTimeout <i>seconds</i>
<b>Default:</b>	ProxyTimeout 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_proxy
<b>Compatibility:</b>	Available in Apache 2.0.31 and later

This directive allows a user to specify a timeout on proxy requests. This is useful when you have a slow/buggy appserver which hangs, and you would rather just return a timeout and fail gracefully instead of waiting however long it takes the server to return.



<b>Description:</b>	Information provided in the <code>Via</code> HTTP response header for proxied requests
<b>Syntax:</b>	<code>ProxyVia On Off Full Block</code>
<b>Default:</b>	<code>ProxyVia Off</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_proxy</code>

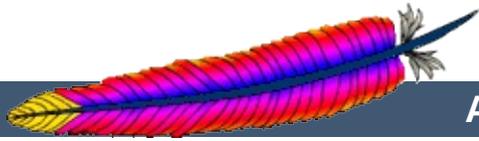
This directive controls the use of the `Via: HTTP` header by the proxy. Its intended use is to control the flow of proxy requests along a chain of proxy servers. See [RFC 2616](#) (HTTP/1.1), section 14.45 for an explanation of `Via: header` lines.

- If set to `Off`, which is the default, no special processing is performed. If a request or reply contains a `Via: header`, it is passed through unchanged.
- If set to `On`, each request and reply will get a `Via: header` line added for the current host.
- If set to `Full`, each generated `Via: header` line will additionally have the Apache server version shown as a `Via: comment` field.
- If set to `Block`, every proxy request will have all its `Via: header` lines removed. No new `Via: header` will be generated.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_proxy_connect`

<b>Description:</b>	<code>mod_proxy</code> extension for CONNECT request handling
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_connect_module</code>
<b>Source File:</b>	<code>proxy_connect.c</code>

## Summary

This module *requires* the service of `mod_proxy`. It provides support for the CONNECT HTTP method. This method is mainly used to tunnel SSL requests through proxy servers.

Thus, in order to get the ability of handling CONNECT requests, `mod_proxy` and `mod_proxy_connect` have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

## See also

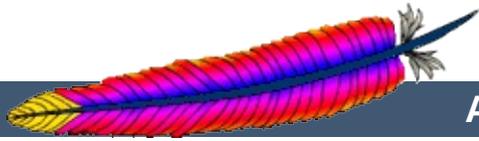
[AllowCONNECT](#)

[mod\\_proxy](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_proxy_ftp`

<b>Description:</b>	FTP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_ftp_module</code>
<b>Source File:</b>	<code>proxy_ftp.c</code>

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides support for the proxying FTP sites.

Thus, in order to get the ability of handling FTP proxy requests, [mod\\_proxy](#) and [mod\\_proxy\\_ftp](#) have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

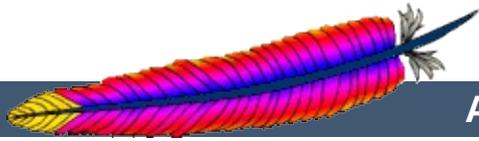
## See also

[mod\\_proxy](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_proxy_http`

<b>Description:</b>	HTTP support module for <a href="#">mod_proxy</a>
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>proxy_http_module</code>
<b>Source File:</b>	<code>proxy_http.c</code>

## Summary

This module *requires* the service of [mod\\_proxy](#). It provides the features used for proxying HTTP requests. [mod\\_proxy\\_http](#) supports HTTP/0.9, HTTP/1.0 and HTTP/1.1. It does *not* provide any caching abilities. If you want to set up a caching proxy, you might want to use the additional service of the [mod\\_cache](#) module.

Thus, in order to get the ability of handling HTTP proxy requests, [mod\\_proxy](#) and [mod\\_proxy\\_http](#) have to be present in the server.

## Warning

Do not enable proxying until you have [secured your server](#). Open proxy servers are dangerous both to your network and to the Internet at large.

## See also

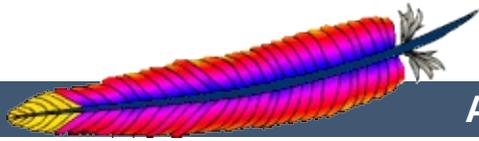
[mod\\_proxy](#)

[mod\\_proxy\\_connect](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_rewrite`

<b>Description:</b>	Provides a rule-based rewriting engine to rewrite requested URLs on the fly
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>rewrite_module</code>
<b>Source File:</b>	<code>mod_rewrite.c</code>
<b>Compatibility:</b>	Available in Apache 1.3 and later

## Summary

This module uses a rule-based rewriting engine (based on a regular-expression parser) to rewrite requested URLs on the fly. It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule, to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests, of server variables, environment variables, HTTP headers, or time stamps. Even external database lookups in various formats can be used to achieve highly granular URL matching.

This module operates on the full URLs (including the path-info part) both in per-server context (`httpd.conf`) and per-directory context (`.htaccess`) and can generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to an internal proxy throughput.

Further details, discussion, and examples, are provided in the [detailed `mod\_rewrite` documentation](#).

## See also

[Rewrite Flags](#)



Apache processes a HTTP request in several phases. A hook for each of these phases is provided by the Apache API.

`mod_rewrite` uses two of these hooks: the URL-to-filename translation hook (used after the HTTP request has been read, but before any authorization starts) and the Fixup hook (triggered after the authorization phases, and after the per-directory config files (`.htaccess`) have been read, but before the content handler is activated).

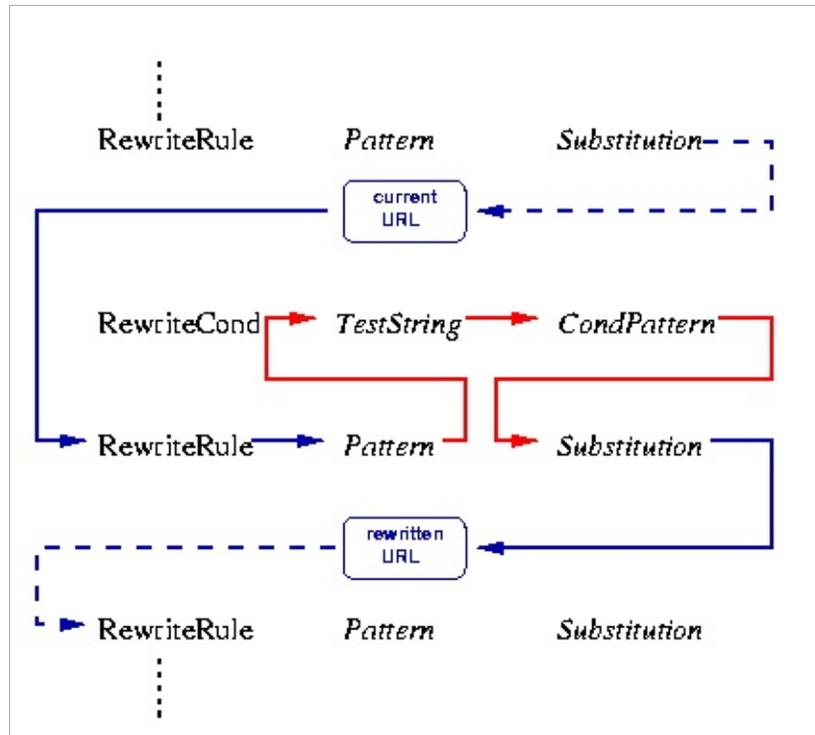
Once a request comes in, and Apache has determined the appropriate server (or virtual server), the rewrite engine starts the URL-to-filename translation, processing the `mod_rewrite` directives from the per-server configuration. A few steps later, when the final data directories are found, the per-directory configuration directives of `mod_rewrite` are triggered in the Fixup phase.



## URL Rewriting

When `mod_rewrite` is triggered during these two API phases, it reads the relevant rulesets from its configuration structure (which was either created on startup, for per-server context, or during the directory traversal for per-directory context). The URL rewriting engine is started with the appropriate ruleset (one or more rules together with their conditions), and its operation is exactly the same for both configuration contexts. Only the final result processing is different.

The order of rules in the ruleset is important because the rewrite engine processes them in a particular (not always obvious) order, as follows: The rewrite engine loops through the rulesets (each ruleset being made up of `RewriteRule` directives, with or without `RewriteConds`), rule by rule. When a particular rule is matched, `mod_rewrite` also checks the corresponding conditions (`RewriteCond` directives). For historical reasons the conditions are given first, making the control flow a little bit long-winded. See Figure 1 for more details.



**Figure**

**1:** The control flow of the rewrite engine through a rewrite ruleset

As above, first the URL is matched against the *Pattern* of a rule. If it does not match, [mod\\_rewrite](#) immediately stops processing that rule, and goes on to the next rule. If the *Pattern* matches, [mod\\_rewrite](#) checks for rule conditions. If none are present, the URL will be replaced with a new string, constructed from the *Substitution* string, and [mod\\_rewrite](#) goes on to the next rule.

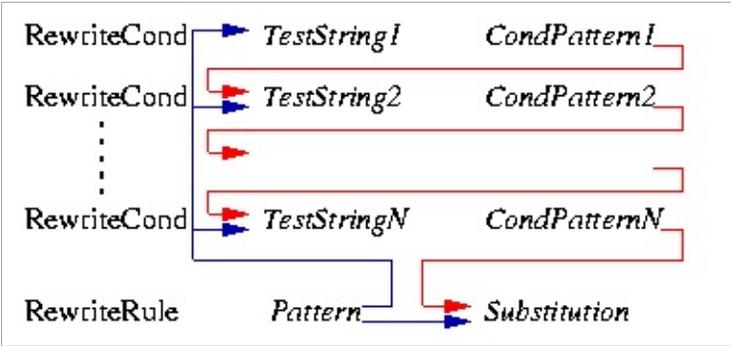
If **RewriteConds** exist, an inner loop is started, processing them in the order that they are listed. Conditions are not matched against the current URL directly. A *TestString* is constructed by expanding variables, back-references, map lookups, etc., against which the *CondPattern* is matched. If the pattern fails to match one of the conditions, the complete set of rule and associated conditions fails. If the pattern matches a given condition, then matching continues to the next condition, until no more conditions are available. If all conditions match, processing is continued with

the substitution of the *Substitution* string for the URL.



## Regular Back-Reference Availability

Using parentheses in *Pattern* or in one of the *CondPatterns* causes back-references to be internally created. These can later be referenced using the strings  $\$N$  and  $\%N$  (see below), for creating the *Substitution* and *TestString* strings. Figure 2 attempts to show how the back-references are transferred through the process for later expansion.



**Figure 2:** The back-reference flow through a rule.



## Escaping Special Characters

As of Apache 1.3.20, special characters in *TestString* and *Substitution* strings can be escaped (that is, treated as normal characters without their usual special meaning) by prefixing them with a backslash ('\') character. In other words, you can include an actual dollar-sign character in a *Substitution* string by using '\\$'; this keeps mod\_rewrite from trying to treat it as a backreference.



This module keeps track of two additional (non-standard) CGI/SSI environment variables named `SCRIPT_URL` and `SCRIPT_URI`. These contain the *logical* Web-view to the current resource, while the standard CGI/SSI variables `SCRIPT_NAME` and `SCRIPT_FILENAME` contain the *physical* System-view.

Notice: These variables hold the URI/URL *as they were initially requested*, that is, *before* any rewriting. This is important to note because the rewriting process is primarily used to rewrite logical URLs to physical pathnames.

### Example

```
SCRIPT_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.html
SCRIPT_FILENAME=/u/rse/.www/index.html
SCRIPT_URL=/u/rse/
SCRIPT_URI=http://en1.engelschall.com/u/rse/
```



For numerous examples of common, and not-so-common, uses for `mod_rewrite`, see the [Rewrite Guide](#), and the [Advanced Rewrite Guide](#) documents.



<b>Description:</b>	Sets the base URL for per-directory rewrites
<b>Syntax:</b>	RewriteBase <i>URL-path</i>
<b>Default:</b>	See usage for information.
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteBase` directive explicitly sets the base URL for per-directory rewrites. As you will see below, `RewriteRule` can be used in per-directory config files (`.htaccess`). In such a case, it will act locally, stripping the local directory prefix before processing, and applying rewrite rules only to the remainder. When processing is complete, the prefix is automatically added back to the path. The default setting is; `RewriteBase physical-directory-path`

When a substitution occurs for a new URL, this module has to re-inject the URL into the server processing. To be able to do this it needs to know what the corresponding URL-prefix or URL-base is. By default this prefix is the corresponding filepath itself. **However, for most websites, URLs are NOT directly related to physical filename paths, so this assumption will often be wrong!** Therefore, you can use the `RewriteBase` directive to specify the correct URL-prefix.

If your webserver's URLs are **not** directly related to physical file paths, you will need to use `RewriteBase` in every `.htaccess` file where you want to use `RewriteRule` directives.

For example, assume the following per-directory config file:

```

#
# /abc/def/.htaccess -- per-dir config file for directory /abc/c
# Remember: /abc/def is the physical path of /xyz, i.e., the ser
#         has a 'Alias /xyz /abc/def' directive e.g.
#
RewriteEngine On

# let the server know that we were reached via /xyz and not
# via the physical path prefix /abc/def
RewriteBase /xyz

# now the rewriting rules
RewriteRule ^oldstuff\.html$ newstuff.html

```

In the above example, a request to `/xyz/oldstuff.html` gets correctly rewritten to the physical file `/abc/def/newstuff.html`.

## For Apache Hackers

The following list gives detailed information about the internal processing steps:

Request:

```
/xyz/oldstuff.html
```

Internal Processing:

```

/xyz/oldstuff.html -> /abc/def/oldstuff.html (per-server /
/abc/def/oldstuff.html -> /abc/def/newstuff.html (per-dir F
/abc/def/newstuff.html -> /xyz/newstuff.html (per-dir F
/xyz/newstuff.html -> /abc/def/newstuff.html (per-server /

```

Result:

```
/abc/def/newstuff.html
```

This seems very complicated, but is in fact correct Apache internal processing. Because the per-directory rewriting comes late in the process, the rewritten request has to be re-injected into the Apache kernel, as if it were a new request. (See [mod\\_rewrite technical details](#).) This is not the serious overhead it may seem to be - this re-injection is completely internal to the Apache server (and the same procedure is used by many other

operations within Apache).



<b>Description:</b>	Defines a condition under which rewriting will take place
<b>Syntax:</b>	<code>RewriteCond <i>TestString CondPattern</i></code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteCond` directive defines a rule condition. One or more `RewriteCond` can precede a `RewriteRule` directive. The following rule is then only used if both the current state of the URI matches its pattern, **and** if these conditions are met.

`TestString` is a string which can contain the following expanded constructs in addition to plain text:

- **RewriteRule backreferences:** These are backreferences of the form `$N` ( $0 \leq N \leq 9$ ), which provide access to the grouped parts (in parentheses) of the pattern, from the `RewriteRule` which is subject to the current set of `RewriteCond` conditions..
- **RewriteCond backreferences:** These are backreferences of the form `%N` ( $1 \leq N \leq 9$ ), which provide access to the grouped parts (again, in parentheses) of the pattern, from the last matched `RewriteCond` in the current set of conditions.
- **RewriteMap expansions:** These are expansions of the form `${mapname:key|default}`. See [the documentation for RewriteMap](#) for more details.
- **Server-Variables:** These are variables of the form `%{NAME_OF_VARIABLE}` where `NAME_OF_VARIABLE` can be a string taken from the following list:

---

<b>HTTP headers:</b>	<b>connection &amp;</b>
----------------------	-------------------------

	<b>request:</b>	
HTTP_USER_AGENT	REMOTE_ADDR	
HTTP_REFERER	REMOTE_HOST	
HTTP_COOKIE	REMOTE_PORT	
HTTP_FORWARDED	REMOTE_USER	
HTTP_HOST	REMOTE_IDENT	
HTTP_PROXY_CONNECTION	REQUEST_METHOD	
HTTP_ACCEPT	SCRIPT_FILENAME	
	PATH_INFO	
	QUERY_STRING	
	AUTH_TYPE	
<b>server internals:</b>	<b>system stuff:</b>	<b>special:</b>
DOCUMENT_ROOT	TIME_YEAR	API_VERSION
SERVER_ADMIN	TIME_MON	THE_REQUEST
SERVER_NAME	TIME_DAY	REQUEST_URI
SERVER_ADDR	TIME_HOUR	REQUEST_FILENAME
SERVER_PORT	TIME_MIN	IS_SUBREQUEST
SERVER_PROTOCOL	TIME_SEC	HTTP_PROTOCOL
SERVER_SOFTWARE	TIME_WDAY	
	TIME	

These variables all correspond to the similarly named HTTP MIME-headers, C variables of the Apache server or struct tm fields of the Unix system. Most are documented elsewhere in the Manual or in the CGI specification. Those that are special to mod\_rewrite include those below.

### **IS\_SUBREQ**

Will contain the text "true" if the request currently being processed is a sub-request, "false" otherwise. Sub-requests may be generated by modules that need to resolve additional files or URIs in order to complete their tasks.

### **API\_VERSION**

This is the version of the Apache module API (the internal interface between server and module) in the current httpd build, as defined in include/ap\_mmn.h. The module API version corresponds to the version of Apache in use (in the release version of Apache 1.3.14, for instance, it is 19990320:10), but is mainly of interest to module authors.

### **THE\_REQUEST**

The full HTTP request line sent by the browser to the server (e.g., "GET /index.html HTTP/1.1"). This does not include any additional headers sent by the browser.

### **REQUEST\_URI**

The resource requested in the HTTP request line. (In the example above, this would be "/index.html".)

### **REQUEST\_FILENAME**

The full local filesystem path to the file or script matching the request.

### **HTTPS**

Will contain the text "on" if the connection is using SSL/TLS, or "off" otherwise. (This variable can be safely used regardless of whether or not `mod_ssl` is loaded).

Other things you should be aware of:

1. The variables `SCRIPT_FILENAME` and `REQUEST_FILENAME` contain the same value - the value of the `filename` field of the internal `request_rec` structure of the Apache server. The first name is the commonly known CGI variable name while the second is the appropriate

counterpart of REQUEST\_URI (which contains the value of the uri field of request\_rec).

2. `%{ENV:variable}`, where *variable* can be any environment variable, is also available. This is looked-up via internal Apache structures and (if not found there) via `getenv()` from the Apache server process.
3. `%{SSL:variable}`, where *variable* is the name of an [SSL environment variable](#), can be used whether or not `mod_ssl` is loaded, but will always expand to the empty string if it is not. Example: `%{SSL:SSL_CIPHER_USEKEYSIZE}` may expand to 128.
4. `%{HTTP:header}`, where *header* can be any HTTP MIME-header name, can always be used to obtain the value of a header sent in the HTTP request. Example: `%{HTTP:Proxy-Connection}` is the value of the HTTP header `Proxy-Connection`.
5. `%{LA-U:variable}` can be used for look-aheads which perform an internal (URL-based) sub-request to determine the final value of *variable*. This can be used to access variable for rewriting which is not available at the current stage, but will be set in a later phase.

For instance, to rewrite according to the REMOTE\_USER variable from within the per-server context (`httpd.conf` file) you must use `%{LA-U:REMOTE_USER}` - this variable is set by the authorization phases, which come *after* the URL translation phase (during which `mod_rewrite` operates).

On the other hand, because `mod_rewrite` implements its per-directory context (`.htaccess` file) via the Fixup phase of the API and because the authorization phases come *before* this phase, you just can use `%{REMOTE_USER}` in that context.

6. `%{LA-F:variable}` can be used to perform an internal (filename-based) sub-request, to determine the final value of *variable*. Most of the time, this is the same as LA-U above.

*CondPattern* is the condition pattern, a regular expression which is applied to the current instance of the *TestString*. *TestString* is first evaluated, before being matched against *CondPattern*.

**Remember:** *CondPattern* is a *perl compatible regular expression* with some additions:

1. You can prefix the pattern string with a '!' character (exclamation mark) to specify a **non**-matching pattern.
2. There are some special variants of *CondPatterns*. Instead of real regular expression strings you can also use one of the following:
  - '**<CondPattern**' (lexicographically precedes)  
Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* lexicographically precedes *CondPattern*.
  - '**>CondPattern**' (lexicographically follows)  
Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* lexicographically follows *CondPattern*.
  - '**=CondPattern**' (lexicographically equal)  
Treats the *CondPattern* as a plain string and compares it lexicographically to *TestString*. True if *TestString* is lexicographically equal to *CondPattern* (the two strings are exactly equal, character for character). If *CondPattern* is "" (two quotation marks) this compares *TestString* to the empty string.
  - '**-d**' (is **directory**)  
Treats the *TestString* as a pathname and tests whether or

not it exists, and is a directory.

- **'-f'** (is regular file)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file.
- **'-s'** (is regular file, with size)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a regular file with size greater than zero.
- **'-l'** (is symbolic link)  
Treats the *TestString* as a pathname and tests whether or not it exists, and is a symbolic link.
- **'-F'** (is existing file, via subrequest)  
Checks whether or not *TestString* is a valid file, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!
- **'-U'** (is existing URL, via subrequest)  
Checks whether or not *TestString* is a valid URL, accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to do the check, so use it with care - it can impact your server's performance!

### Note

All of these tests can also be prefixed by an exclamation mark (!) to negate their meaning.

3. You can also set special flags for *CondPattern* by appending **[flags]** as the third argument to the `RewriteCond` directive, where *flags* is a comma-separated list of any of the following

flags:

- **'nocase | NC' (no case)**  
This makes the test case-insensitive - differences between 'A-Z' and 'a-z' are ignored, both in the expanded *TestString* and the *CondPattern*. This flag is effective only for comparisons between *TestString* and *CondPattern*. It has no effect on filesystem and subrequest checks.
- **'ornext | OR' (or next condition)**  
Use this to combine rule conditions with a local OR instead of the implicit AND. Typical example:

```
RewriteCond %{REMOTE_HOST} =host1 [OR]
RewriteCond %{REMOTE_HOST} =host2 [OR]
RewriteCond %{REMOTE_HOST} =host3
RewriteRule ...some special stuff for any of these hosts
```

Without this flag you would have to write the condition/rule pair three times.

### Example:

To rewrite the Homepage of a site according to the ``User - Agent :'' header of the request, you can use the following:

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla
RewriteRule ^/$ /homepage.max.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx
RewriteRule ^/$ /homepage.min.html [L]

RewriteRule ^/$ /homepage.std.html [L]
```

Explanation: If you use a browser which identifies itself as 'Mozilla' (including Netscape Navigator, Mozilla etc), then you get the max homepage (which could include frames, or other special features).

If you use the Lynx browser (which is terminal-based), then you get the min homepage (which could be a version designed for easy, text-only browsing). If neither of these conditions apply (you use any other browser, or your browser identifies itself as something non-standard), you get the std (standard) homepage.



## Runtime Engine Directive

<b>Description:</b>	Enables or disables runtime rewriting engine
<b>Syntax:</b>	RewriteEngine on off
<b>Default:</b>	RewriteEngine off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The `RewriteEngine` directive enables or disables the runtime rewriting engine. If it is set to `off` this module does no runtime processing at all. It does not even update the `SCRIPT_URx` environment variables.

Use this directive to disable the module instead of commenting out all the `RewriteRule` directives!

Note that, by default, rewrite configurations are not inherited. This means that you need to have a `RewriteEngine on` directive for each virtual host in which you wish to use it.

`RewriteMap` directives of the type `prg` are not started during server initialization if they're defined in a context that does not have `RewriteEngine` set to `on`



<b>Description:</b>	Sets the name of the lock file used for <a href="#">RewriteMap</a> synchronization
<b>Syntax:</b>	RewriteLock <i>file-path</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

This directive sets the filename for a synchronization lockfile which mod\_rewrite needs to communicate with [RewriteMap programs](#). Set this lockfile to a local path (not on a NFS-mounted device) when you want to use a rewriting map-program. It is not required for other types of rewriting maps.



## RewriteLog Directive

<b>Description:</b>	Sets the name of the file used for logging rewrite engine processing
<b>Syntax:</b>	RewriteLog <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite

The **RewriteLog** directive sets the name of the file to which the server logs any rewriting actions it performs. If the name does not begin with a slash ('/') then it is assumed to be relative to the *Server Root*. The directive should occur only once per server config.

To disable the logging of rewriting actions it is not recommended to set *Filename* to `/dev/null`, because although the rewriting engine does not then output to a logfile it still creates the logfile output internally. **This will slow down the server with no advantage to the administrator!** To disable logging either remove or comment out the **RewriteLog** directive or use `RewriteLogLevel 0!`

### Security

See the [Apache Security Tips](#) document for details on how your security could be compromised if the directory where logfiles are stored is writable by anyone other than the user that starts the server.

### Example

```
RewriteLog "/usr/local/var/apache/logs/rewrite.log"
```



<b>Description:</b>	Sets the verbosity of the log file used by the rewrite engine
<b>Syntax:</b>	<code>RewriteLogLevel <i>Level</i></code>
<b>Default:</b>	<code>RewriteLogLevel 0</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_rewrite</code>

The `RewriteLogLevel` directive sets the verbosity level of the rewriting logfile. The default level 0 means no logging, while 9 or more means that practically all actions are logged.

To disable the logging of rewriting actions simply set *Level* to 0. This disables all rewrite action logs.

Using a high value for *Level* will slow down your Apache server dramatically! Use the rewriting logfile at a *Level* greater than 2 only for debugging!

### Example

```
RewriteLogLevel 3
```



<b>Description:</b>	Defines a mapping function for key-lookup
<b>Syntax:</b>	<code>RewriteMap MapName MapType:MapSource</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite
<b>Compatibility:</b>	The choice of different dbm types is available in Apache 2.0.41 and later

The `RewriteMap` directive defines a *Rewriting Map* which can be used inside rule substitution strings by the mapping-functions to insert/substitute fields through a key lookup. The source of this lookup can be of various types.

The *MapName* is the name of the map and will be used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

```
${ MapName : LookupKey }  
${ MapName : LookupKey | DefaultValue }
```

When such a construct occurs, the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the key is not found then it is substituted by *DefaultValue* or by the empty string if no *DefaultValue* was specified.

For example, you might define a `RewriteMap` as:

```
RewriteMap examplemap txt:/path/to/file/map.txt
```

You would then be able to use this map in a `RewriteRule` as follows:

```
RewriteRule ^/ex/(.*) ${examplemap:$1}
```

The following combinations for *MapType* and *MapSource* can be used:

- **Standard Plain Text**

MapType: txt, MapSource: Unix filesystem path to valid regular file

This is the standard rewriting map feature where the *MapSource* is a plain ASCII file containing either blank lines, comment lines (starting with a '#' character) or pairs like the following - one per line.

### ***MatchingKey SubstValue***

#### **Example**

```
##  
## map.txt -- rewriting map  
##  
  
Ralf.S.Engelschall    rse    # Bastard Operator From Hell  
Mr.Joe.Average       joe    # Mr. Average
```

```
RewriteMap real-to-user txt:/path/to/file/map.txt
```

- **Randomized Plain Text**

MapType: rnd, MapSource: Unix filesystem path to valid regular file

This is identical to the Standard Plain Text variant above but with a special post-processing feature: After looking up a value it is parsed according to contained ``|" characters which have the meaning of ``or". In other words they indicate a set of alternatives from which the actual returned value is

chosen randomly. For example, you might use the following map file and directives to provide a random load balancing between several back-end server, via a reverse-proxy. Images are sent to one of the servers in the 'static' pool, while everything else is sent to one of the 'dynamic' pool.

Example:

### Rewrite map file

```
##
##  map.txt -- rewriting map
##

static  www1|www2|www3|www4
dynamic www5|www6
```

### Configuration directives

```
RewriteMap servers rnd:/path/to/file/map.txt

RewriteRule ^/(.*\.(png|gif|jpg))
http://${servers:static}/$1 [NC,P,L]
RewriteRule ^/(.*) http://${servers:dynamic}/$1 [P,L]
```

- **Hash File**

MapType: dbm[=*type*], MapSource: Unix filesystem path to valid regular file

Here the source is a binary format DBM file containing the same contents as a *Plain Text* format file, but in a special representation which is optimized for really fast lookups. The *type* can be sdbm, gdbm, ndbm, or db depending on [compile-time settings](#). If the *type* is omitted, the compile-time default will be chosen. You can create such a file with any DBM tool or with the following Perl script. Be sure to adjust it to create the appropriate type of DBM. The example creates an NDBM

file.

```
#!/path/to/bin/perl
##
##  txt2dbm -- convert txt map to dbm format
##

use NDBM_File;
use Fcntl;

($txtmap, $dbmmap) = @ARGV;

open(TXT, "<$txtmap") or die "Couldn't open $txtmap!\n";
tie (%DB, 'NDBM_File', $dbmmap, O_RDWR|O_TRUNC|O_CREAT, 0644)
    or die "Couldn't create $dbmmap!\n";

while (<TXT>) {
    next if (/^\s*#/ or /^\s*$/);
    $DB{$1} = $2 if (/^\s*(\S+)\s+(\S+)/);
}

untie %DB;
close(TXT);
```

```
$ txt2dbm map.txt map.db
```

- **Internal Function**

MapType: `int`, MapSource: Internal Apache function

Here, the source is an internal Apache function. Currently you cannot create your own, but the following functions already exist:

- **toupper:**

Converts the key to all upper case.

- **tolower:**  
Converts the key to all lower case.
- **escape:**  
Translates special characters in the key to hex-encodings.
- **unescape:**  
Translates hex-encodings in the key back to special characters.

- **External Rewriting Program**

MapType: prg, MapSource: Unix filesystem path to valid regular file

Here the source is a program, not a map file. To create it you can use a language of your choice, but the result has to be an executable program (either object-code or a script with the magic cookie trick '#!/path/to/interpreter' as the first line).

This program is started once, when the Apache server is started, and then communicates with the rewriting engine via its `stdin` and `stdout` file-handles. For each map-function lookup it will receive the key to lookup as a newline-terminated string on `stdin`. It then has to give back the looked-up value as a newline-terminated string on `stdout` or the four-character string ```NULL"` if it fails (*i.e.*, there is no corresponding value for the given key). A trivial program which will implement a 1:1 map (*i.e.*, `key == value`) could be:

External rewriting programs are not started if they're defined in a context that does not have `RewriteEngine` set to on

```
#!/usr/bin/perl
```

```
$| = 1;
while (<STDIN>) {
    # ...put here any transformations or lookups...
    print $_;
}
```

But be very careful:

1. *"Keep it simple, stupid"* (KISS). If this program hangs, it will cause Apache to hang when trying to use the relevant rewrite rule.
2. A common mistake is to use buffered I/O on stdout. Avoid this, as it will cause a deadlock! ```$|=1"` is used above, to prevent this.
3. The [RewriteLock](#) directive can be used to define a lockfile which `mod_rewrite` can use to synchronize communication with the mapping program. By default no such synchronization takes place.

The [RewriteMap](#) directive can occur more than once. For each mapping-function use one [RewriteMap](#) directive to declare its rewriting mapfile. While you cannot **declare** a map in per-directory context it is of course possible to **use** this map in per-directory context.

### Note

For plain text and DBM format files the looked-up keys are cached in-core until the `mtime` of the mapfile changes or the server does a restart. This way you can have map-functions in rules which are used for **every** request. This is no problem, because the external lookup only happens once!



<b>Description:</b>	Sets some special options for the rewrite engine
<b>Syntax:</b>	<code>RewriteOptions Options</code>
<b>Default:</b>	<code>RewriteOptions MaxRedirects=10</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite
<b>Compatibility:</b>	MaxRedirects is available in Apache 2.0.45 and later

The `RewriteOptions` directive sets some special options for the current per-server or per-directory configuration. The *Option* strings can be one of the following:

### **inherit**

This forces the current configuration to inherit the configuration of the parent. In per-virtual-server context this means that the maps, conditions and rules of the main server are inherited. In per-directory context this means that conditions and rules of the parent directory's `.htaccess` configuration are inherited.

### **MaxRedirects=number**

In order to prevent endless loops of internal redirects issued by per-directory `RewriteRules`, `mod_rewrite` aborts the request after reaching a maximum number of such redirects and responds with an 500 Internal Server Error. If you really need more internal redirects than 10 per request, you may increase the default to the desired value.

### **AllowAnyURI**

When `RewriteRule` is used in `VirtualHost` or server context with version 2.0.65 or later of `httpd`, `mod_rewrite`

will only process the rewrite rules if the request URI is a [URL-path](#). This avoids some security issues where particular rules could allow "surprising" pattern expansions (see [CVE-2011-3368](#) and [CVE-2011-4317](#)). To lift the restriction on matching a URL-path, the `AllowAnyURI` option can be enabled, and `mod_rewrite` will apply the rule set to any request URI string, regardless of whether that string matches the URL-path grammar required by the HTTP specification.

### Security Warning

Enabling this option will make the server vulnerable to security issues if used with rewrite rules which are not carefully authored. It is **strongly recommended** that this option is not used. In particular, beware of input strings containing the '@' character which could change the interpretation of the transformed URI, as per the above CVE names.

### MergeBase

With this option, the value of `RewriteBase` is copied from where it's explicitly defined into any sub-directory or sub-location that doesn't define its own `RewriteBase`. This flag is available for Apache HTTP Server 2.0.65 and later.



<b>Description:</b>	Defines rules for the rewriting engine
<b>Syntax:</b>	RewriteRule <i>Pattern Substitution</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_rewrite
<b>Compatibility:</b>	The cookie-flag is available in Apache 2.0.40 and later.

The **RewriteRule** directive is the real rewriting workhorse. The directive can occur more than once, with each instance defining a single rewrite rule. The order in which these rules are defined is important - this is the order in which they will be applied at run-time.

*Pattern* is a perl compatible regular expression, which is applied to the current URL. ``Current" means the value of the URL when this rule is applied. This may not be the originally requested URL, which may already have matched a previous rule, and have been altered.

Some hints on the syntax of regular expressions:

**Text:**

. Any single character  
 [chars] Character class: Any character of the class ``chars"  
 [^chars] Character class: Not a character of the class ``chars"  
 text1|text2 Alternative: text1 or text2

**Quantifiers:**

? 0 or 1 occurrences of the preceding text  
 \* 0 or N occurrences of the preceding text (N > 0)  
 + 1 or N occurrences of the preceding text (N > 1)

**Grouping:**

(text) Grouping of text

(used either to set the borders of an alternative or to make backreferences, where the **N**th group can be referred to on the RHS of a RewriteRule as **\$N**)

**Anchors:**

^ Start-of-line anchor  
\$ End-of-line anchor

**Escaping:**

\char escape the given char  
(for instance, to specify the chars ".[]()" etc.)

For more information about regular expressions, have a look at the perl regular expression manpage ("[perldoc perlre](#)"). If you are interested in more detailed information about regular expressions and their variants (POSIX regex etc.) the following book is dedicated to this topic:

*Mastering Regular Expressions, 2nd Edition*

Jeffrey E.F. Friedl

O'Reilly & Associates, Inc. 2002

ISBN 0-596-00289-0

In `mod_rewrite`, the NOT character ('!') is also available as a possible pattern prefix. This enables you to negate a pattern; to say, for instance: ```if the current URL does NOT match this pattern"`. This can be used for exceptional cases, where it is easier to match the negative pattern, or as a last default rule.

**Note**

When using the NOT character to negate a pattern, you cannot include grouped wildcard parts in that pattern. This is because, when the pattern does NOT match (ie, the negation matches), there are no contents for the groups. Thus, if negated patterns are used, you cannot use `$N` in the substitution string!

The *substitution* of a rewrite rule is the string which is substituted

for (or replaces) the original URL which *Pattern* matched. In addition to plain text, it can include

1. back-references ( $\$N$ ) to the RewriteRule pattern
2. back-references ( $\%N$ ) to the last matched RewriteCond pattern
3. server-variables as in rule condition test-strings ( $\% \{VARNAME\}$ )
4. [mapping-function](#) calls ( $\$\{mapname : key | default\}$ )

Back-references are identifiers of the form  $\$N$  ( $N=0..9$ ), which will be replaced by the contents of the  $N$ th group of the matched *Pattern*. The server-variables are the same as for the *TestString* of a RewriteCond directive. The mapping-functions come from the RewriteMap directive and are explained there. These three types of variables are expanded in the order above.

As already mentioned, all rewrite rules are applied to the *Substitution* (in the order in which they are defined in the config file). The URL is **completely replaced** by the *Substitution* and the rewriting process continues until all rules have been applied, or it is explicitly terminated by a **L** flag - see below.

There is a special substitution string named '-' which means: **NO substitution!** This is useful in providing rewriting rules which **only** match URLs but do not substitute anything for them. It is commonly used in conjunction with the **C** (chain) flag, in order to apply more than one pattern before substitution occurs.

Additionally you can set special flags for *Substitution* by appending **[flags]** as the third argument to the RewriteRule directive. *Flags* is a comma-separated list of any of the following flags:

- **'chain|C'** (chained with next rule)

This flag chains the current rule with the next rule (which itself can be chained with the following rule, and so on). This has the following effect: if a rule matches, then processing continues as usual - the flag has no effect. If the rule does **not** match, then all following chained rules are skipped. For instance, it can be used to remove the ``.www" part, inside a per-directory rule set, when you let an external redirect happen (where the ``.www" part should not occur!).

- '**cookie** | **CO**=*NAME*:*VAL*:*domain*[:*lifetime*[:*path*]]' (set **cookie**)  
This sets a cookie in the client's browser. The cookie's name is specified by *NAME* and the value is *VAL*. The *domain* field is the domain of the cookie, such as '.apache.org', the optional *lifetime* is the lifetime of the cookie in minutes, and the optional *path* is the path of the cookie
- '**env** | **E**=*VAR*:*VAL*' (set **environment variable**)  
This forces an environment variable named *VAR* to be set to the value *VAL*, where *VAL* can contain regexp backreferences (*\$N* and *%N*) which will be expanded. You can use this flag more than once, to set more than one variable. The variables can later be dereferenced in many situations, most commonly from within XSSI (via `<! - -#echo var="VAR" - ->`) or CGI (`ENV{'VAR'}`). You can also dereference the variable in a later RewriteCond pattern, using `%{ENV:VAR}`. Use this to strip information from URLs, while maintaining a record of that information.
- '**forbidden** | **F**' (force URL to be **forbidden**)  
This forces the current URL to be forbidden - it immediately sends back a HTTP response of 403 (FORBIDDEN). Use this flag in conjunction with appropriate RewriteConds to conditionally block some URLs.
- '**gone** | **G**' (force URL to be **gone**)  
This forces the current URL to be gone - it immediately sends back a HTTP response of 410 (GONE). Use this flag to mark

pages which no longer exist as gone.

- **'last | L'** (last rule)

Stop the rewriting process here and don't apply any more rewrite rules. This corresponds to the Perl `last` command or the `break` command in C. Use this flag to prevent the currently rewritten URL from being rewritten further by following rules. For example, use it to rewrite the root-path URL (`/`) to a real one, e.g., `/e/www/`.

- **'next | N'** (next round)

Re-run the rewriting process (starting again with the first rewriting rule). This time, the URL to match is no longer the original URL, but rather the URL returned by the last rewriting rule. This corresponds to the Perl `next` command or the `continue` command in C. Use this flag to restart the rewriting process - to immediately go to the top of the loop.

**Be careful not to create an infinite loop!**

- **'nocase | NC'** (no case)

This makes the *Pattern* case-insensitive, ignoring difference between 'A-Z' and 'a-z' when *Pattern* is matched against the current URL.

- **'noescape | NE'** (no URI escaping of output)

This flag prevents `mod_rewrite` from applying the usual URI escaping rules to the result of a rewrite. Ordinarily, special characters (such as '%', '\$', ';', and so on) will be escaped into their hexcode equivalents ('%25', '%24', and '%3B', respectively); this flag prevents this from happening. This allows percent symbols to appear in the output, as in

```
RewriteRule /foo/(.*) /bar?arg=P1\%3d$1 [R,NE]
```

which would turn `/foo/zed` into a safe request for `/bar?arg=P1=zed`.

- **'nosubreq | NS'** ( not for internal sub-requests)

This flag forces the rewrite engine to skip a rewrite rule if the current request is an internal sub-request. For instance, sub-requests occur internally in Apache when [mod\\_include](#) tries to find out information about possible directory default files (`index.xxx`). On sub-requests it is not always useful, and can even cause errors, if the complete set of rules are applied. Use this flag to exclude some rules.

To decide whether or not to use this rule: if you prefix URLs with CGI-scripts, to force them to be processed by the CGI-script, it's likely that you will run into problems (or significant overhead) on sub-requests. In these cases, use this flag.

- **'proxy|P'** (force proxy)

This flag forces the substitution part to be internally sent as a proxy request and immediately (rewrite processing stops here) put through the [proxy module](#). You must make sure that the substitution string is a valid URI (typically starting with `http://hostname`) which can be handled by the Apache proxy module. If not, you will get an error from the proxy module. Use this flag to achieve a more powerful implementation of the [ProxyPass](#) directive, to map remote content into the namespace of the local server.

Note: [mod\\_proxy](#) must be enabled in order to use this flag.

- **'passthrough|PT'** (pass through to next handler)

This flag forces the rewrite engine to set the `uri` field of the internal `request_rec` structure to the value of the `filename` field. This flag is just a hack to enable post-processing of the output of `RewriteRule` directives, using `Alias`, `ScriptAlias`, `Redirect`, and other directives from various URI-to-filename translators. For example, to rewrite `/abc` to `/def` using [mod\\_rewrite](#), and then `/def` to `/ghi` using [mod\\_alias](#):

```
RewriteRule ^/abc(.*) /def$1 [PT]
```

```
Alias /def /ghi
```

If you omit the PT flag, `mod_rewrite` will rewrite `uri=/abc/...` to `filename=/def/...` as a full API-compliant URI-to-filename translator should do. Then `mod_alias` will try to do a URI-to-filename transition, which will fail.

**Note: You must use this flag if you want to mix directives from different modules which allow URL-to-filename translators.** The typical example is the use of `mod_alias` and `mod_rewrite`.

- **'qsappend|QSA'** (query string append)  
This flag forces the rewrite engine to append a query string part of the substitution string to the existing string, instead of replacing it. Use this when you want to add more data to the query string via a rewrite rule.
- **'redirect|R [=code]'** (force redirect)  
Prefix *Substitution* with `http://thishost[:thisport]/` (which makes the new URL a URI) to force an external redirection. If no *code* is given, a HTTP response of 302 (MOVED TEMPORARILY) will be returned. If you want to use other response codes in the range 300-400, simply specify the appropriate number or use one of the following symbolic names: `temp` (default), `permanent`, `seeother`. Use this for rules to canonicalize the URL and return it to the client - to translate `~/~` into `~/u/`, or to always append a slash to `/u/user`, etc.  
**Note:** When you use this flag, make sure that the substitution field is a valid URL! Otherwise, you will be redirecting to an invalid location. Remember that this flag on its own will only prepend `http://thishost[:thisport]/` to the URL, and rewriting will continue. Usually, you will want to stop rewriting at this point, and redirect immediately. To stop rewriting, you

should add the 'L' flag.

- **'skip|S=num'** (skip next rule(s))

This flag forces the rewriting engine to skip the next *num* rules in sequence, if the current rule matches. Use this to make pseudo if-then-else constructs: The last rule of the then-clause becomes `skip=N`, where N is the number of rules in the else-clause. (This is **not** the same as the 'chain|C' flag!)

- **'type|T=MIME-type'** (force MIME type)

Force the MIME-type of the target file to be *MIME-type*. This can be used to set up the content-type based on some conditions. For example, the following snippet allows `.php` files to be *displayed* by `mod_php` if they are called with the `.phps` extension:

```
RewriteRule ^(.+\..php)s$ $1 [T=application/x-httpd-php-source]
```

## Home directory expansion

When the substitution string begins with a string resembling `"/~user"` (via explicit text or backreferences), `mod_rewrite` performs home directory expansion independent of the presence or configuration of [mod\\_userdir](#).

This expansion does not occur when the *PT* flag is used on the [RewriteRule](#) directive.

## Note: Enabling rewrites in per-directory context

To enable the rewriting engine for per-directory configuration files, you need to set ```RewriteEngine On``` in these files **and** ```Options FollowSymLinks``` must be enabled. If your administrator has disabled override of `FollowSymLinks` for a user's directory, then you cannot use the rewriting engine. This restriction is needed for security reasons.

### **Note: Pattern matching in per-directory context**

Never forget that *Pattern* is applied to a complete URL in per-server configuration files. **However, in per-directory configuration files, the per-directory prefix (which always is the same for a specific directory) is automatically *removed* for the pattern matching and automatically *added* after the substitution has been done.** This feature is essential for many sorts of rewriting - without this, you would always have to match the parent directory which is not always possible.

There is one exception: If a substitution string starts with ```http://```, then the directory prefix will **not** be added, and an external redirect or proxy throughput (if flag **P** is used) is forced!

### **Note: Substitution of Absolute URLs**

When you prefix a substitution field with `http://thishost[:thisport]`, `mod_rewrite` will automatically strip that out. This auto-reduction on URLs with an implicit external redirect is most useful in combination with a mapping-function which generates the hostname part.

**Remember:** An unconditional external redirect to your own server will not work with the prefix `http://thishost` because of this feature. To achieve such a self-redirect, you have to use the **R**-flag.

### **Note: Query String**

The *Pattern* will not be matched against the query string. Instead, you must use a `RewriteCond` with the `%{QUERY_STRING}` variable. You can, however, create URLs in the substitution string, containing a query string part. Simply use a question mark inside the substitution string, to indicate that the following text should be re-injected into the query string. When

you want to erase an existing query string, end the substitution string with just a question mark. To combine a new query string with an old one, use the [QSA] flag.

Here are all possible substitution combinations and their meanings:

**Inside per-server configuration (httpd.conf)  
for request `GET /somepath/pathinfo`:**

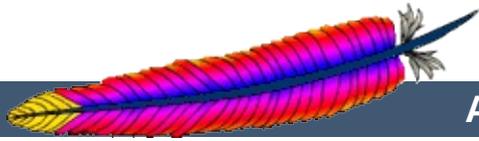
Given Rule	Resulting Substit
^/somepath(.*) otherpath\$1	invalid, not supp
^/somepath(.*) otherpath\$1 [R]	invalid, not supp
^/somepath(.*) otherpath\$1 [P]	invalid, not supp
^/somepath(.*) /otherpath\$1	/otherpath/pathir
^/somepath(.*) /otherpath\$1 [R]	http://thishost/c via external redi
^/somepath(.*) /otherpath\$1 [P]	doesn't make sens
^/somepath(.*) http://thishost/otherpath\$1	/otherpath/pathir
^/somepath(.*) http://thishost/otherpath\$1 [R]	http://thishost/c via external redi
^/somepath(.*) http://thishost/otherpath\$1 [P]	doesn't make sens
^/somepath(.*) http://otherhost/otherpath\$1	http://otherhost/ via external redi
^/somepath(.*) http://otherhost/otherpath\$1 [R]	http://otherhost/ via external redi (the [R] flag is
^/somepath(.*) http://otherhost/otherpath\$1 [P]	http://otherhost/ via internal pro

**Inside per-directory configuration for /somepath  
 (/physical/path/to/somepath/.htaccess, with  
 RewriteBase /somepath)  
 for request `GET /somepath/localpath/pathinfo`:**

Given Rule	Resulting Substitution
<code>^localpath(.*) otherpath\$1</code>	<code>/somepath/otherpath</code>
<code>^localpath(.*) otherpath\$1 [R]</code>	<code>http://thishost/somepath/otherpath</code> via external redirect
<code>^localpath(.*) otherpath\$1 [P]</code>	doesn't make sense
<code>^localpath(.*) /otherpath\$1</code>	<code>/otherpath/pathinfo</code>
<code>^localpath(.*) /otherpath\$1 [R]</code>	<code>http://thishost/somepath/otherpath</code> via external redirect
<code>^localpath(.*) /otherpath\$1 [P]</code>	doesn't make sense
<code>^localpath(.*) http://thishost/otherpath\$1</code>	<code>/otherpath/pathinfo</code>
<code>^localpath(.*) http://thishost/otherpath\$1 [R]</code>	<code>http://thishost/somepath/otherpath</code> via external redirect
<code>^localpath(.*) http://thishost/otherpath\$1 [P]</code>	doesn't make sense
<code>^localpath(.*) http://otherhost/otherpath\$1</code>	<code>http://otherhost/somepath/otherpath</code> via external redirect
<code>^localpath(.*) http://otherhost/otherpath\$1 [R]</code>	<code>http://otherhost/somepath/otherpath</code> via external redirect (the [R] flag is not needed)
<code>^localpath(.*) http://otherhost/otherpath\$1 [P]</code>	<code>http://otherhost/somepath/otherpath</code> via internal proxy

---

Copyright 2013 The Apache Software Foundation.  
 Licensed under the [Apache License, Version 2.0](#).



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_setenvif`

<b>Description:</b>	Allows the setting of environment variables based on characteristics of the request
<b>Status:</b>	Base
<b>Module Identifier:</b>	<code>setenvif_module</code>
<b>Source File:</b>	<code>mod_setenvif.c</code>

## Summary

The `mod_setenvif` module allows you to set environment variables according to whether different aspects of the request match regular expressions you specify. These environment variables can be used by other parts of the server to make decisions about actions to be taken.

The directives are considered in the order they appear in the configuration files. So more complex sequences can be used, such as this example, which sets `net_scape` if the browser is `mozilla` but not `MSIE`.

```
BrowserMatch ^Mozilla netscape
BrowserMatch MSIE !netscape
```

## See also

[Environment Variables in Apache](#)



**Description:** Sets environment variables conditional on HTTP User-Agent

**Syntax:** `BrowserMatch regex [!]env-variable[=value] [[!]env-variable[=value]] ...`

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Status:** Base

**Module:** mod\_setenvif

The `BrowserMatch` is a special cases of the `SetEnvIf` directive that sets environment variables conditional on the User - Agent HTTP request header. The following two lines have the same effect:

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```

Some additional examples:

```
BrowserMatch ^Mozilla forms jpeg=yes browser=netscape
BrowserMatch "^Mozilla/[2-3]" tables agif frames javascript
BrowserMatch MSIE !javascript
```



<b>Description:</b>	Sets environment variables conditional on User-Agent without respect to case
<b>Syntax:</b>	<code>BrowserMatchNoCase <i>regex</i> [!]env-variable[=<i>value</i>] [[!]env-variable[=<i>value</i>]] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_setenvif
<b>Compatibility:</b>	Apache 1.2 and above (in Apache 1.2 this directive was found in the now-obsolete mod_browser module)

The `BrowserMatchNoCase` directive is semantically identical to the `BrowserMatch` directive. However, it provides for case-insensitive matching. For example:

```
BrowserMatchNoCase mac platform=macintosh
BrowserMatchNoCase win platform=windows
```

The `BrowserMatch` and `BrowserMatchNoCase` directives are special cases of the `SetEnvIf` and `SetEnvIfNoCase` directives. The following two lines have the same effect:

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```



<b>Description:</b>	Sets environment variables based on attributes of the request
<b>Syntax:</b>	<code>SetEnvIf <i>attribute</i> <i>regex</i> [!]env-variable[=<i>value</i>] [[!]env-variable[=<i>value</i>]] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_setenvif

The `SetEnvIf` directive defines environment variables based on attributes of the request. The *attribute* specified in the first argument can be one of three things:

1. An HTTP request header field (see [RFC2616](#) for more information about these); for example: Host, User-Agent, Referer, and Accept-Language. A regular expression may be used to specify a set of request headers.
2. One of the following aspects of the request:
  - Remote\_Host - the hostname (if available) of the client making the request
  - Remote\_Addr - the IP address of the client making the request
  - Server\_Addr - the IP address of the server on which the request was received (only with versions later than 2.0.43)
  - Request\_Method - the name of the method being used (GET, POST, *et cetera*)
  - Request\_Protocol - the name and version of the protocol with which the request was made (e.g.,

"HTTP/0.9", "HTTP/1.1", etc.)

- Request\_URI - the resource requested on the HTTP request line -- generally the portion of the URL following the scheme and host portion without the query string. See the [RewriteCond](#) directive of [mod\\_rewrite](#) for extra information on how to match your query string.

3. The name of an environment variable in the list of those associated with the request. This allows [SetEnvIf](#) directives to test against the result of prior matches. Only those environment variables defined by earlier [SetEnvIf](#) [NoCase] directives are available for testing in this manner. 'Earlier' means that they were defined at a broader scope (such as server-wide) or previously in the current directive's scope. Environment variables will be considered only if there was no match among request characteristics and a regular expression was not used for the *attribute*.

The second argument (*regex*) is a [Perl compatible regular expression](#). This is similar to a POSIX.2 egrep-style regular expression. If the *regex* matches against the *attribute*, then the remainder of the arguments are evaluated.

The rest of the arguments give the names of variables to set, and optionally values to which they should be set. These take the form of

1. *varname*, or
2. *!varname*, or
3. *varname=value*

In the first form, the value will be set to "1". The second will remove the given variable if already defined, and the third will set

the variable to the literal value given by *value*. Since version 2.0.51 Apache will recognize occurrences of  $\$1..\$9$  within *value* and replace them by parenthesized subexpressions of *regex*.

### Example:

```
SetEnvIf Request_URI "\.gif$" object_is_image=gif
SetEnvIf Request_URI "\.jpg$" object_is_image=jpg
SetEnvIf Request_URI "\.xpm$" object_is_image=xpm
:
SetEnvIf Referer www\.mydomain\.com intra_site_referral
:
SetEnvIf object_is_image xpm XBIT_PROCESSING=1
:
SetEnvIf ^TS* ^[a-z].* HAVE_TS
```

The first three will set the environment variable `object_is_image` if the request was for an image file, and the fourth sets `intra_site_referral` if the referring page was somewhere on the `www.mydomain.com` Web site.

The last example will set environment variable `HAVE_TS` if the request contains any headers that begin with "TS" whose values begins with any character in the set `[a-z]`.

### See also

- [Environment Variables in Apache](#), for additional examples.

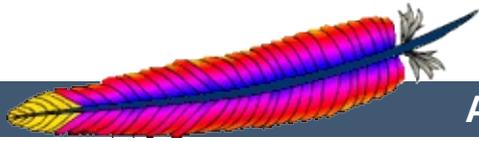


<b>Description:</b>	Sets environment variables based on attributes of the request without respect to case
<b>Syntax:</b>	<code>SetEnvIfNoCase <i>attribute regex</i> [!]env-variable[=value] [[!]env-variable[=value]] ...</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Base
<b>Module:</b>	mod_setenvif
<b>Compatibility:</b>	Apache 1.3 and above

The `SetEnvIfNoCase` is semantically identical to the `SetEnvIf` directive, and differs only in that the regular expression matching is performed in a case-insensitive manner. For example:

```
SetEnvIfNoCase Host Apache\.Org site=apache
```

This will cause the `site` environment variable to be set to "apache" if the HTTP request header field `Host :` was included and contained `Apache.Org`, `apache.org`, or any other combination.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_so

<b>Description:</b>	Loading of executable code and modules into the server at start-up or restart time
<b>Status:</b>	Extension
<b>Module Identifier:</b>	so_module
<b>Source File:</b>	mod_so.c
<b>Compatibility:</b>	This is a Base module (always included) on Windows

### Summary

On selected operating systems this module can be used to load modules into Apache at runtime via the [Dynamic Shared Object](#) (DSO) mechanism, rather than requiring a recompilation.

On Unix, the loaded code typically comes from shared object files (usually with .so extension), on Windows this may either be the .so or .dll extension.

### Warning

Apache 1.3 modules cannot be directly used with Apache 2.0 - the module must be modified to dynamically load or compile into Apache 2.0.



## Note

The module name format changed for Windows with Apache 1.3.15 and 2.0 - the modules are now named as `mod_foo.so`

While `mod_so` still loads modules with `ApacheModuleFoo.dll` names, the new naming convention is preferred; if you are converting your loadable module for 2.0, please fix the name to this 2.0 convention.

The Apache module API is unchanged between the Unix and Windows versions. Many modules will run on Windows with no or little change from Unix, although others rely on aspects of the Unix architecture which are not present in Windows, and will not work.

When a module does work, it can be added to the server in one of two ways. As with Unix, it can be compiled into the server. Because Apache for Windows does not have the `Configure` program of Apache for Unix, the module's source file must be added to the `ApacheCore` project file, and its symbols must be added to the `os\win32\modules.c` file.

The second way is to compile the module as a DLL, a shared library that can be loaded into the server at runtime, using the `LoadModule` directive. These module DLLs can be distributed and run on any Apache for Windows installation, without recompilation of the server.

To create a module DLL, a small change is necessary to the module's source file: The module record must be exported from the DLL (which will be created later; see below). To do this, add the `AP_MODULE_DECLARE_DATA` (defined in the Apache header files) to your module's module record definition. For example, if your module has:

```
module foo_module;
```

Replace the above with:

```
module AP_MODULE_DECLARE_DATA foo_module;
```

Note that this will only be activated on Windows, so the module can continue to be used, unchanged, with Unix if needed. Also, if you are familiar with `.DEF` files, you can export the module record with that method instead.

Now, create a DLL containing your module. You will need to link this against the `libhttpd.lib` export library that is created when the `libhttpd.dll` shared library is compiled. You may also have to change the compiler settings to ensure that the Apache header files are correctly located. You can find this library in your server root's modules directory. It is best to grab an existing module `.dsp` file from the tree to assure the build environment is configured correctly, or alternately compare the compiler and link options to your `.dsp`.

This should create a DLL version of your module. Now simply place it in the `modules` directory of your server root, and use the `LoadModule` directive to load it.



<b>Description:</b>	Link in the named object file or library
<b>Syntax:</b>	LoadFile <i>filename</i> [ <i>filename</i> ] ...
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_so

The LoadFile directive links in the named object files or libraries when the server is started or restarted; this is used to load additional code which may be required for some module to work. *Filename* is either an absolute path or relative to [ServerRoot](#).

For example:

```
LoadFile libexec/libxmlparse.so
```

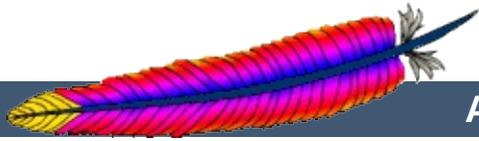


<b>Description:</b>	Links in the object file or library, and adds to the list of active modules
<b>Syntax:</b>	<code>LoadModule <i>module</i> <i>filename</i></code>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_so

The `LoadModule` directive links in the object file or library *filename* and adds the module structure named *module* to the list of active modules. *Module* is the name of the external variable of type `module` in the file, and is listed as the [Module Identifier](#) in the module documentation. Example:

```
LoadModule status_module modules/mod_status.so
```

loads the named module from the `modules` subdirectory of the `ServerRoot`.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_speling

<b>Description:</b>	Attempts to correct mistaken URLs that users might have entered by ignoring capitalization and by allowing up to one misspelling
<b>Status:</b>	Extension
<b>Module Identifier:</b>	speling_module
<b>Source File:</b>	mod_speling.c

## Summary

Requests to documents sometimes cannot be served by the core apache server because the request was misspelled or miscapitalized. This module addresses this problem by trying to find a matching document, even after all other modules gave up. It does its work by comparing each document name in the requested directory against the requested document name **without regard to case**, and allowing **up to one misspelling** (character insertion / omission / transposition or wrong character). A list is built with all document names which were matched using this strategy.

If, after scanning the directory,

- no matching document was found, Apache will proceed as usual and return a "document not found" error.
- only one document is found that "almost" matches the request, then it is returned in the form of a redirection response.
- more than one document with a close match was found, then the list of the matches is returned to the client, and the client can select the correct candidate.



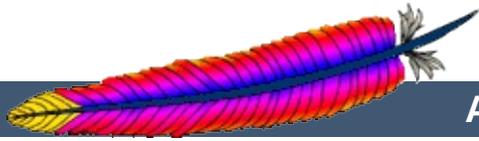
## CheckSpelling Directive

<b>Description:</b>	Enables the spelling module
<b>Syntax:</b>	CheckSpelling on off
<b>Default:</b>	CheckSpelling Off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_speling
<b>Compatibility:</b>	CheckSpelling was available as a separately available module for Apache 1.1, but was limited to miscapitalizations. As of Apache 1.3, it is part of the Apache distribution. Prior to Apache 1.3.2, the CheckSpelling directive was only available in the "server" and "virtual host" contexts.

This directive enables or disables the spelling module. When enabled, keep in mind that

- the directory scan which is necessary for the spelling correction will have an impact on the server's performance when many spelling corrections have to be performed at the same time.
- the document trees should not contain sensitive files which could be matched inadvertently by a spelling "correction".
- the module is unable to correct misspelled user names (as in `http://my.host/~apahce/`), just file names or directory names.
- spelling corrections apply strictly to existing files, so a request for the `<Location /status>` may get incorrectly treated as the negotiated file `"/stats.html"`.

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_ssl

<b>Description:</b>	Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols
<b>Status:</b>	Extension
<b>Module Identifier:</b>	ssl_module
<b>Source File:</b>	mod_ssl.c

## Summary

This module provides SSL v2/v3 and TLS v1 support for the Apache HTTP Server. It was contributed by Ralf S. Engeschall based on his mod\_ssl project and originally derived from work by Ben Laurie.

This module relies on [OpenSSL](#) to provide the cryptography engine.

Further details, discussion, and examples are provided in the [SSL documentation](#).



This module provides a lot of SSL information as additional environment variables to the SSI and CGI namespace. The generated variables are listed in the table below. For backward compatibility the information can be made available under different names, too. Look in the [Compatibility](#) chapter for details on the compatibility variables.

<b>Variable Name:</b>	<b>Value Type:</b>	<b>Description:</b>
HTTPS	flag	HTTPS is being used.
SSL_PROTOCOL	string	The SSL protocol version (SSLv2, SSLv3, TLSv1)
SSL_SESSION_ID	string	The hex-encoded SSL session id
SSL_CIPHER	string	The cipher specification name
SSL_CIPHER_EXPORT	string	true if cipher is an export cipher
SSL_CIPHER_USEKEYSIZE	number	Number of cipher bits (actually used)
SSL_CIPHER_ALGKEYSIZE	number	Number of cipher bits (possible)
SSL_VERSION_INTERFACE	string	The mod_ssl program version
SSL_VERSION_LIBRARY	string	The OpenSSL program version
SSL_CLIENT_M_VERSION	string	The version of the client certificate
SSL_CLIENT_M_SERIAL	string	The serial of the client certificate
SSL_CLIENT_S_DN	string	Subject DN in client's

		certificate
SSL_CLIENT_S_DN_x509	string	Component of client's Subject DN
SSL_CLIENT_I_DN	string	Issuer DN of client's certificate
SSL_CLIENT_I_DN_x509	string	Component of client's Issuer DN
SSL_CLIENT_V_START	string	Validity of client's certificate (start time)
SSL_CLIENT_V_END	string	Validity of client's certificate (end time)
SSL_CLIENT_A_SIG	string	Algorithm used for the signature of client's certificate
SSL_CLIENT_A_KEY	string	Algorithm used for the public key of client's certificate
SSL_CLIENT_CERT	string	PEM-encoded client certificate
SSL_CLIENT_CERT_CHAIN <i>n</i>	string	PEM-encoded certificates in client certificate chain
SSL_CLIENT_VERIFY	string	NONE, SUCCESS, GENEROUS or FAILED : <i>reason</i>
SSL_SERVER_M_VERSION	string	The version of the server certificate
SSL_SERVER_M_SERIAL	string	The serial of the server certificate
SSL_SERVER_S_DN	string	Subject DN in server's certificate
SSL_SERVER_S_DN_x509	string	Component of server's

		Subject DN
SSL_SERVER_I_DN	string	Issuer DN of server's certificate
SSL_SERVER_I_DN_x509	string	Component of server's Issuer DN
SSL_SERVER_V_START	string	Validity of server's certificate (start time)
SSL_SERVER_V_END	string	Validity of server's certificate (end time)
SSL_SERVER_A_SIG	string	Algorithm used for the signature of server's certificate
SSL_SERVER_A_KEY	string	Algorithm used for the public key of server's certificate
SSL_SERVER_CERT	string	PEM-encoded server certificate

[ where x509 is a component of a X.509 DN:  
C, ST, L, O, OU, CN, T, I, G, S, D, UID, Email ]



## Custom Log Format

When `mod_ssl` is built into Apache or at least loaded (under DSO situation) additional functions exist for the [Custom Log Format](#) of `mod_log_config`. First there is an additional ``%{varname}x`" eXtension format function which can be used to expand any variables provided by any module, especially those provided by `mod_ssl` which can you find in the above table.

For backward compatibility there is additionally a special ``%{name}c`" cryptography format function provided. Information about this function is provided in the [Compatibility](#) chapter.

Example:

```
CustomLog logs/ssl_request_log \ "%t %h %{SSL_PROTOCOL}x %  
{SSL_CIPHER}x \"%r\" %b"
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	<code>SSLCACertificateFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLCACertificatePath](#).

### Example

```
SSLCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-client.crt
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for Client Auth
<b>Syntax:</b>	SSLCACertificatePath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose clients you deal with. These are used to verify the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value*.N. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCACertificatePath /usr/local/apache2/conf/ssl.crt/
```



**Description:** File of concatenated PEM-encoded CA CRLs for Client Auth

**Syntax:** SSLCARevocationFile *file-path*

**Context:** server config, virtual host

**Status:** Extension

**Module:** mod\_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *clients* you deal with. These are used for Client Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLCARevocationPath](#).

### Example

```
SSLCARevocationFile /usr/local/apache2/conf/ssl.crl/ca-bundle-client.crl
```



<b>Description:</b>	Directory of PEM-encoded CA CRLs for Client Auth
<b>Syntax:</b>	SSLCARevocationPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose clients you deal with. These are used to revoke the client certificate on Client Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links.

### Example

```
SSLCARevocationPath /usr/local/apache2/conf/ssl.crl/
```



<b>Description:</b>	File of PEM-encoded Server CA Certificates
<b>Syntax:</b>	<code>SSLCertificateChainFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ssl</code>

This directive sets the optional *all-in-one* file where you can assemble the certificates of Certification Authorities (CA) which form the certificate chain of the server certificate. This starts with the issuing CA certificate of the server certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA Certificate files, usually in certificate chain order.

This should be used alternatively and/or additionally to [SSLCACertificatePath](#) for explicitly constructing the server certificate chain which is sent to the browser in addition to the server certificate. It is especially useful to avoid conflicts with CA certificates when using client authentication. Because although placing a CA certificate of the server certificate chain into [SSLCACertificatePath](#) has the same effect for the certificate chain construction, it has the side-effect that client certificates issued by this same CA certificate are also accepted on client authentication. That's usually not one expect.

But be careful: Providing the certificate chain works only if you are using a *single* (either RSA or DSA) based server certificate. If you are using a coupled RSA+DSA certificate pair, this will work only if actually both certificates use the *same* certificate chain. Else the browsers will be confused in this situation.

### Example

```
SSLCertificateChainFile /usr/local/apache2/conf/ssl.crt/ca.crt
```



<b>Description:</b>	Server PEM-encoded X.509 Certificate file
<b>Syntax:</b>	SSLCertificateFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive points to the PEM-encoded Certificate file for the server and optionally also to the corresponding RSA or DSA Private Key file for it (contained in the same file). If the contained Private Key is encrypted the Pass Phrase dialog is forced at startup time. This directive can be used up to two times (referencing different filenames) when both a RSA and a DSA based server certificate is used in parallel.

### Example

```
SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
```



<b>Description:</b>	Server PEM-encoded Private Key file
<b>Syntax:</b>	<code>SSLCertificateKeyFile</code> <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_ssl</code>

This directive points to the PEM-encoded Private Key file for the server. If the Private Key is not combined with the Certificate in the `SSLCertificateFile`, use this additional directive to point to the file with the stand-alone Private Key. When `SSLCertificateFile` is used and the file contains both the Certificate and the Private Key this directive need not be used. But we strongly discourage this practice. Instead we recommend you to separate the Certificate and the Private Key. If the contained Private Key is encrypted, the Pass Phrase dialog is forced at startup time. This directive can be used up to two times (referencing different filenames) when both a RSA and a DSA based private key is used in parallel.

### Example

```
SSLCertificateKeyFile  
/usr/local/apache2/conf/ssl.key/server.key
```



<b>Description:</b>	Cipher Suite available for negotiation in SSL handshake
<b>Syntax:</b>	SSLCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This complex directive uses a colon-separated *cipher-spec* string consisting of OpenSSL cipher specifications to configure the Cipher Suite the client is permitted to negotiate in the SSL handshake phase. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured Cipher Suite after the HTTP request was read but before the HTTP response is sent.

An SSL cipher specification in *cipher-spec* is composed of 4 major attributes plus a few extra minor ones:

- *Key Exchange Algorithm:*  
RSA or Diffie-Hellman variants.
- *Authentication Algorithm:*  
RSA, Diffie-Hellman, DSS or none.
- *Cipher/Encryption Algorithm:*  
DES, Triple-DES, RC4, RC2, IDEA or none.
- *MAC Digest Algorithm:*  
MD5, SHA or SHA1.

An SSL cipher can also be an export cipher and is either a SSLv2 or SSLv3/TLSv1 cipher (here TLSv1 is equivalent to SSLv3). To

specify which ciphers to use, one can either specify all the Ciphers, one at a time, or use aliases to specify the preference and order for the ciphers (see [Table 1](#)).

<b>Tag</b>	<b>Description</b>
<i>Key Exchange Algorithm:</i>	
kRSA	RSA key exchange
kDHR	Diffie-Hellman key exchange with RSA key
kDHd	Diffie-Hellman key exchange with DSA key
kEDH	Ephemeral (temp.key) Diffie-Hellman key exchange (no cert)
<i>Authentication Algorithm:</i>	
aNULL	No authentication
aRSA	RSA authentication
aDSS	DSS authentication
aDH	Diffie-Hellman authentication
<i>Cipher Encoding Algorithm:</i>	
eNULL	No encoding
DES	DES encoding
3DES	Triple-DES encoding
RC4	RC4 encoding
RC2	RC2 encoding
IDEA	IDEA encoding
<i>MAC Digest Algorithm:</i>	
MD5	MD5 hash function
SHA1	SHA1 hash function
SHA	SHA hash function
<i>Aliases:</i>	
SSLv2	all SSL version 2.0 ciphers
SSLv3	all SSL version 3.0 ciphers

TLSv1	all TLS version 1.0 ciphers
EXP	all export ciphers
EXPORT40	all 40-bit export ciphers only
EXPORT56	all 56-bit export ciphers only
LOW	all low strength ciphers (no export, single DES)
MEDIUM	all ciphers with 128 bit encryption
HIGH	all ciphers using Triple-DES
RSA	all ciphers using RSA key exchange
DH	all ciphers using Diffie-Hellman key exchange
EDH	all ciphers using Ephemeral Diffie-Hellman key exchange
ADH	all ciphers using Anonymous Diffie-Hellman key exchange
DSS	all ciphers using DSS authentication
NULL	all ciphers using no encryption

Now where this becomes interesting is that these can be put together to specify the order and ciphers you wish to use. To speed this up there are also aliases (SSLv2, SSLv3, TLSv1, EXP, LOW, MEDIUM, HIGH) for certain groups of ciphers. These tags can be joined together with prefixes to form the *cipher-spec*. Available prefixes are:

- none: add cipher to list
- +: move matching ciphers to the current location in list
- -: remove cipher from list (can be added later again)
- !: kill cipher from list completely (can **not** be added later again)

A simpler way to look at all of this is to use the `openssl ciphers -v` command which provides a nice way to successively create the correct *cipher-spec* string. The default

*cipher-spec* string is

```
``ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP``
```

which means the following: first, remove from consideration any ciphers that do not authenticate, i.e. for SSL only the Anonymous Diffie-Hellman ciphers. Next, use ciphers using RC4 and RSA. Next include the high, medium and then the low security ciphers. Finally *pull* all SSLv2 and export ciphers to the end of the list.

```
$ openssl ciphers -v 'ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:
NULL-SHA          SSLv3 Kx=RSA      Au=RSA  Enc=None      M
NULL-MD5          SSLv3 Kx=RSA      Au=RSA  Enc=None      M
EDH-RSA-DES-CBC3-SHA  SSLv3 Kx=DH        Au=RSA  Enc=3DES(168) M
...
EXP-RC4-MD5       SSLv3 Kx=RSA(512) Au=RSA  Enc=RC4(40)  M
EXP-RC2-CBC-MD5   SSLv2 Kx=RSA(512) Au=RSA  Enc=RC2(40)  M
EXP-RC4-MD5       SSLv2 Kx=RSA(512) Au=RSA  Enc=RC4(40)  M
```

The complete list of particular RSA & DH ciphers for SSL is given in [Table 2](#).

```
Example
SSLCipherSuite RSA:!EXP:!NULL:+HIGH:+MEDIUM:-LOW
```

Cipher-Tag	Protocol	Key Ex.	Auth.	Enc.	MAC	Type
<i>RSA Ciphers:</i>						
DES-CBC3-SHA	SSLv3	RSA	RSA	3DES(168)	SHA1	
DES-CBC3-MD5	SSLv2	RSA	RSA	3DES(168)	MD5	
IDEA-CBC-SHA	SSLv3	RSA	RSA	IDEA(128)	SHA1	
RC4-SHA	SSLv3	RSA	RSA	RC4(128)	SHA1	
RC4-MD5	SSLv3	RSA	RSA	RC4(128)	MD5	

IDEA - CBC - MD5	SSLv2	RSA	RSA	IDEA(128)	MD5	
RC2 - CBC - MD5	SSLv2	RSA	RSA	RC2(128)	MD5	
RC4 - MD5	SSLv2	RSA	RSA	RC4(128)	MD5	
DES - CBC - SHA	SSLv3	RSA	RSA	DES(56)	SHA1	
RC4 - 64 - MD5	SSLv2	RSA	RSA	RC4(64)	MD5	
DES - CBC - MD5	SSLv2	RSA	RSA	DES(56)	MD5	
EXP - DES - CBC - SHA	SSLv3	RSA(512)	RSA	DES(40)	SHA1	export
EXP - RC2 - CBC - MD5	SSLv3	RSA(512)	RSA	RC2(40)	MD5	export
EXP - RC4 - MD5	SSLv3	RSA(512)	RSA	RC4(40)	MD5	export
EXP - RC2 - CBC - MD5	SSLv2	RSA(512)	RSA	RC2(40)	MD5	export
EXP - RC4 - MD5	SSLv2	RSA(512)	RSA	RC4(40)	MD5	export
NULL - SHA	SSLv3	RSA	RSA	None	SHA1	
NULL - MD5	SSLv3	RSA	RSA	None	MD5	
<i>Diffie-Hellman Ciphers:</i>						
ADH - DES - CBC3 - SHA	SSLv3	DH	None	3DES(168)	SHA1	
ADH - DES - CBC - SHA	SSLv3	DH	None	DES(56)	SHA1	
ADH - RC4 - MD5	SSLv3	DH	None	RC4(128)	MD5	

EDH-RSA-DES-CBC3-SHA	SSLv3	DH	RSA	3DES(168)	SHA1	
EDH-DSS-DES-CBC3-SHA	SSLv3	DH	DSS	3DES(168)	SHA1	
EDH-RSA-DES-CBC-SHA	SSLv3	DH	RSA	DES(56)	SHA1	
EDH-DSS-DES-CBC-SHA	SSLv3	DH	DSS	DES(56)	SHA1	
EXP-EDH-RSA-DES-CBC-SHA	SSLv3	DH(512)	RSA	DES(40)	SHA1	export
EXP-EDH-DSS-DES-CBC-SHA	SSLv3	DH(512)	DSS	DES(40)	SHA1	export
EXP-ADH-DES-CBC-SHA	SSLv3	DH(512)	None	DES(40)	SHA1	export
EXP-ADH-RC4-MD5	SSLv3	DH(512)	None	RC4(40)	MD5	export



<b>Description:</b>	SSL Engine Operation Switch
<b>Syntax:</b>	SSLEngine on off
<b>Default:</b>	SSLEngine off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for both the main server and all configured virtual hosts.

### Example

```
<VirtualHost _default_:443>  
SSLEngine on  
...  
</VirtualHost>
```



<b>Description:</b>	Option to prefer the server's cipher preference order
<b>Syntax:</b>	SSLHonorCipherOrder <i>flag</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in Apache 2.0.65 and later, if using OpenSSL 0.9.7 or later

When choosing a cipher during an SSLv3 or TLSv1 handshake, normally the client's preference is used. If this directive is enabled, the server's preference will be used instead.

### Example

```
SSLHonorCipherOrder on
```



<b>Description:</b>	Option to enable support for insecure renegotiation
<b>Syntax:</b>	SSLInsecureRenegotiation <i>flag</i>
<b>Default:</b>	SSLInsecureRenegotiation off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in httpd 2.0.64 and later, if using OpenSSL 0.9.8m or later

As originally specified, all versions of the SSL and TLS protocols (up to and including TLS/1.2) were vulnerable to a Man-in-the-Middle attack ([CVE-2009-3555](#)) during a renegotiation. This vulnerability allowed an attacker to "prefix" a chosen plaintext to the HTTP request as seen by the web server. A protocol extension was developed which fixed this vulnerability if supported by both client and server.

If [mod\\_ssl](#) is linked against OpenSSL version 0.9.8m or later, by default renegotiation is only supported with clients supporting the new protocol extension. If this directive is enabled, renegotiation will be allowed with old (unpatched) clients, albeit insecurely.

### Security warning

If this directive is enabled, SSL connections will be vulnerable to the Man-in-the-Middle prefix attack as described in [CVE-2009-3555](#).

### Example

```
SSLInsecureRenegotiation on
```

The `SSL_SECURE_RENEG` environment variable can be used from an SSI or CGI script to determine whether secure renegotiation is supported for a given SSL connection.



<b>Description:</b>	Semaphore for internal mutual exclusion of operations
<b>Syntax:</b>	SSLMutex <i>type</i>
<b>Default:</b>	SSLMutex none
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures the SSL engine's semaphore (aka. lock) which is used for mutual exclusion of operations which have to be done in a synchronized way between the pre-forked Apache server processes. This directive can only be used in the global server context because it's only useful to have one global mutex. This directive is designed to closely match the [AcceptMutex](#) directive

The following Mutex *types* are available:

- none | no  
This is the default where no Mutex is used at all. Use it at your own risk. But because currently the Mutex is mainly used for synchronizing write access to the SSL Session Cache you can live without it as long as you accept a sometimes garbled Session Cache. So it's not recommended to leave this the default. Instead configure a real Mutex.
- posixsem  
This is an elegant Mutex variant where a Posix Semaphore is used when possible. It is only available when the underlying platform and APR supports it.
- sysvsem  
This is a somewhat elegant Mutex variant where a SystemV IPC Semaphore is used when possible. It is possible to "leak"

SysV semaphores if processes crash before the semaphore is removed. It is only available when the underlying platform and APR supports it.

- `sem`

This directive tells the SSL Module to pick the "best" semaphore implementation available to it, choosing between Posix and SystemV IPC, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- `pthread`

This directive tells the SSL Module to use Posix thread mutexes. It is only available if the underlying platform and APR supports it.

- `fcntl:/path/to/mutex`

This is a portable Mutex variant where a physical (lock-)file and the `fcntl()` function are used as the Mutex. Always use a local disk filesystem for `/path/to/mutex` and never a file residing on a NFS- or AFS-filesystem. It is only available when the underlying platform and APR supports it. Note: Internally, the Process ID (PID) of the Apache parent process is automatically appended to `/path/to/mutex` to make it unique, so you don't have to worry about conflicts yourself. Notice that this type of mutex is not available under the Win32 environment. There you *have* to use the semaphore mutex.

- `flock:/path/to/mutex`

This is similar to the `fcntl:/path/to/mutex` method with the exception that the `flock()` function is used to provide file locking. It is only available when the underlying platform and APR supports it.

- `file:/path/to/mutex`

This directive tells the SSL Module to pick the "best" file locking implementation available to it, choosing between `fcntl` and `flock`, in that order. It is only available when the underlying platform and APR supports at least one of the 2.

- `default` | `yes`

This directive tells the SSL Module to pick the default locking implementation as determined by the platform and APR.

### Example

```
SSLMutex file:/usr/local/apache/logs/ssl_mutex
```



<b>Description:</b>	Configure various SSL engine run-time options
<b>Syntax:</b>	SSLOptions [+ -] <i>option</i> ...
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control various run-time options on a per-directory basis. Normally, if multiple `SSLOptions` could apply to a directory, then the most specific one is taken completely; the options are not merged. However if *all* the options on the `SSLOptions` directive are preceded by a plus (+) or minus (-) symbol, the options are merged. Any options preceded by a + are added to the options currently in force, and any options preceded by a - are removed from the options currently in force.

The available *options* are:

- `StdEnvVars`  
When this option is enabled, the standard set of SSL related CGI/SSI environment variables are created. This per default is disabled for performance reasons, because the information extraction step is a rather expensive operation. So one usually enables this option for CGI and SSI requests only.
- `CompatEnvVars`  
When this option is enabled, additional CGI/SSI environment variables are created for backward compatibility to other Apache SSL solutions. Look in the [Compatibility](#) chapter for details on the particular variables generated.
- `ExportCertData`  
When this option is enabled, additional CGI/SSI environment

variables are created: `SSL_SERVER_CERT`, `SSL_CLIENT_CERT` and `SSL_CLIENT_CERT_CHAIN $n$`  (with  $n = 0,1,2,\dots$ ). These contain the PEM-encoded X.509 Certificates of server and client for the current HTTPS connection and can be used by CGI scripts for deeper Certificate checking. Additionally all other certificates of the client certificate chain are provided, too. This bloats up the environment a little bit which is why you have to use this option to enable it on demand.

- **FakeBasicAuth**

When this option is enabled, the Subject Distinguished Name (DN) of the Client X509 Certificate is translated into a HTTP Basic Authorization username. This means that the standard Apache authentication methods can be used for access control. The user name is just the Subject of the Client's X509 Certificate (can be determined by running OpenSSL's `openssl x509` command: `openssl x509 -noout -subject -in certificate.crt`). Note that no password is obtained from the user. Every entry in the user file needs this password: ``xj31ZMTZzkVA"`, which is the DES-encrypted version of the word ``password"`. Those who live under MD5-based encryption (for instance under FreeBSD or BSD/OS, etc.) should use the following MD5 hash of the same word: ``$1$0XLYS...$0wx8s2/m9/gfkcRVXzgoE/"`.

- **StrictRequire**

This *forces* forbidden access when `SSLRequireSSL` or `SSLRequire` successfully decided that access should be forbidden. Usually the default is that in the case where a ``Satisfy any"` directive is used, and other access restrictions are passed, denial of access due to `SSLRequireSSL` or `SSLRequire` is overridden (because that's how the Apache `Satisfy` mechanism should work.)

But for strict access restriction you can use `SSLRequireSSL` and/or `SSLRequire` in combination with an `SSLOptions +StrictRequire`". Then an additional `Satisfy Any`" has no chance once `mod_ssl` has decided to deny access.

- `OptRenegotiate`

This enables optimized SSL connection renegotiation handling when SSL directives are used in per-directory context. By default a strict scheme is enabled where *every* per-directory reconfiguration of SSL parameters causes a *full* SSL renegotiation handshake. When this option is used `mod_ssl` tries to avoid unnecessary handshakes by doing more granular (but still safe) parameter checks. Nevertheless these granular checks sometimes maybe not what the user expects, so enable this on a per-directory basis only, please.

### Example

```
SSLOptions +FakeBasicAuth -StrictRequire
<Files ~ "\.(cgi|html)$">
SSLOptions +StdEnvVars +CompatEnvVars -ExportCertData
<Files>
```



<b>Description:</b>	Type of pass phrase dialog for encrypted private keys
<b>Syntax:</b>	SSLPassPhraseDialog <i>type</i>
<b>Default:</b>	SSLPassPhraseDialog builtin
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

When Apache starts up it has to read the various Certificate (see [SSLCertificateFile](#)) and Private Key (see [SSLCertificateKeyFile](#)) files of the SSL-enabled virtual servers. Because for security reasons the Private Key files are usually encrypted, mod\_ssl needs to query the administrator for a Pass Phrase in order to decrypt those files. This query can be done in two ways which can be configured by *type*:

- **builtin**  
This is the default where an interactive terminal dialog occurs at startup time just before Apache detaches from the terminal. Here the administrator has to manually enter the Pass Phrase for each encrypted Private Key file. Because a lot of SSL-enabled virtual hosts can be configured, the following reuse-scheme is used to minimize the dialog: When a Private Key file is encrypted, all known Pass Phrases (at the beginning there are none, of course) are tried. If one of those known Pass Phrases succeeds no dialog pops up for this particular Private Key file. If none succeeded, another Pass Phrase is queried on the terminal and remembered for the next round (where it perhaps can be reused).

This scheme allows mod\_ssl to be maximally flexible (because for N encrypted Private Key files you *can* use N different Pass Phrases - but then you have to enter all of

them, of course) while minimizing the terminal dialog (i.e. when you use a single Pass Phrase for all N Private Key files this Pass Phrase is queried only once).

- `exec:/path/to/program`

Here an external program is configured which is called at startup for each encrypted Private Key file. It is called with two arguments (the first is of the form `servername:portnumber`", the second is either `RSA`" or `DSA`"), which indicate for which server and algorithm it has to print the corresponding Pass Phrase to `stdout`. The intent is that this external program first runs security checks to make sure that the system is not compromised by an attacker, and only when these checks were passed successfully it provides the Pass Phrase.

Both these security checks, and the way the Pass Phrase is determined, can be as complex as you like. `Mod_ssl` just defines the interface: an executable program which provides the Pass Phrase on `stdout`. Nothing more or less! So, if you're really paranoid about security, here is your interface. Anything else has to be left as an exercise to the administrator, because local security requirements are so different.

The reuse-algorithm above is used here, too. In other words: The external program is called only once per unique Pass Phrase.

Example:

```
SSLPassPhraseDialog exec:/usr/local/apache/sbin/pp-filter
```



<b>Description:</b>	Configure usable SSL protocol flavors
<b>Syntax:</b>	SSLProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProtocol all
<b>Context:</b>	server config, virtual host
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control the SSL protocol flavors mod\_ssl should use when establishing its server environment. Clients then can only connect with one of the provided protocols.

The available (case-insensitive) *protocols* are:

- SSLv2  
This is the Secure Sockets Layer (SSL) protocol, version 2.0. It is the original SSL protocol as designed by Netscape Corporation.
- SSLv3  
This is the Secure Sockets Layer (SSL) protocol, version 3.0. It is the successor to SSLv2 and the currently (as of February 1999) de-facto standardized SSL protocol from Netscape Corporation. It's supported by almost all popular browsers.
- TLSv1  
This is the Transport Layer Security (TLS) protocol, version 1.0. It is the successor to SSLv3 and currently (as of February 1999) still under construction by the Internet Engineering Task Force (IETF). It's still not supported by any popular browsers.
- All  
This is a shortcut for "+SSLv2 +SSLv3 +TLSv1" and a

convenient way for enabling all protocols except one when used in combination with the minus sign on a protocol as the example above shows.

### Example

```
# enable SSLv3 and TLSv1, but not SSLv2  
SSLProtocol all -SSLv2
```



<b>Description:</b>	File of concatenated PEM-encoded CA Certificates for Remote Server Auth
<b>Syntax:</b>	SSLProxyCACertificateFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificates of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded Certificate files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCACertificatePath](#).

### Example

```
SSLProxyCACertificateFile /usr/local/apache2/conf/ssl.crt/ca-bundle-remote-server.crt
```



<b>Description:</b>	Directory of PEM-encoded CA Certificates for Remote Server Auth
<b>Syntax:</b>	SSLProxyCACertificatePath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificates of Certification Authorities (CAs) whose remote servers you deal with. These are used to verify the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you can't just place the Certificate files there: you also have to create symbolic links named *hash-value*.N. And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with `mod_ssl` to accomplish this task.

### Example

```
SSLProxyCACertificatePath /usr/local/apache2/conf/ssl.crt/
```



<b>Description:</b>	File of concatenated PEM-encoded CA CRLs for Remote Server Auth
<b>Syntax:</b>	SSLProxyCAREvocationFile <i>file-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the *all-in-one* file where you can assemble the Certificate Revocation Lists (CRL) of Certification Authorities (CA) whose *remote servers* you deal with. These are used for Remote Server Authentication. Such a file is simply the concatenation of the various PEM-encoded CRL files, in order of preference. This can be used alternatively and/or additionally to [SSLProxyCAREvocationPath](#).

### Example

```
SSLProxyCAREvocationFile /usr/local/apache2/conf/ssl.crl/ca-  
bundle-remote-server.crl
```



<b>Description:</b>	Directory of PEM-encoded CA CRLs for Remote Server Auth
<b>Syntax:</b>	SSLProxyCAREvocationPath <i>directory-path</i>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the Certificate Revocation Lists (CRL) of Certification Authorities (CAs) whose remote servers you deal with. These are used to revoke the remote server certificate on Remote Server Authentication.

The files in this directory have to be PEM-encoded and are accessed through hash filenames. So usually you have not only to place the CRL files there. Additionally you have to create symbolic links named *hash-value*.rN. And you should always make sure this directory contains the appropriate symbolic links. Use the `Makefile` which comes with [mod\\_ssl](#) to accomplish this task.

### Example

```
SSLProxyCAREvocationPath /usr/local/apache2/conf/ssl.crl/
```



<b>Description:</b>	Cipher Suite available for negotiation in SSL proxy h
<b>Syntax:</b>	SSLProxyCipherSuite <i>cipher-spec</i>
<b>Default:</b>	SSLProxyCipherSuite ALL:!ADH:RC4+RSA:+HIGH:+MEDIUM:+LOW:+S
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

Equivalent to SSLCipherSuite, but for the proxy connection.  
Please refer to [SSLCipherSuite](#) for additional information.



## SSLProxyEngine Directive

<b>Description:</b>	SSL Proxy Engine Operation Switch
<b>Syntax:</b>	SSLProxyEngine on off
<b>Default:</b>	SSLProxyEngine off
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive toggles the usage of the SSL/TLS Protocol Engine for proxy. This is usually used inside a `<VirtualHost>` section to enable SSL/TLS for proxy usage in a particular virtual host. By default the SSL/TLS Protocol Engine is disabled for proxy image both for the main server and all configured virtual hosts.

### Example

```
<VirtualHost _default_:443>  
SSLProxyEngine on  
...  
</VirtualHost>
```



<b>Description:</b>	File of concatenated PEM-encoded client certificates and keys to be used by the proxy
<b>Syntax:</b>	SSLProxyMachineCertificateFile <i>filename</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the all-in-one file where you keep the certificates and keys used for authentication of the proxy server to remote servers.

This referenced file is simply the concatenation of the various PEM-encoded certificate files, in order of preference. Use this directive alternatively or additionally to SSLProxyMachineCertificatePath.

Currently there is no support for encrypted private keys

Example:

```
SSLProxyMachineCertificateFile  
/usr/local/apache2/conf/ssl.crt/proxy.pem
```



<b>Description:</b>	Directory of PEM-encoded client certificates and keys to be used by the proxy
<b>Syntax:</b>	SSLProxyMachineCertificatePath <i>directory</i>
<b>Context:</b>	server config
<b>Override:</b>	Not applicable
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the directory where you keep the certificates and keys used for authentication of the proxy server to remote servers.

The files in this directory must be PEM-encoded and are accessed through hash filenames. Additionally, you must create symbolic links named *hash-value.N*. And you should always make sure this directory contains the appropriate symbolic links. Use the Makefile which comes with mod\_ssl to accomplish this task.

Currently there is no support for encrypted private keys

Example:

```
SSLProxyMachineCertificatePath  
/usr/local/apache2/conf/proxy.crt/
```



<b>Description:</b>	Configure usable SSL protocol flavors for proxy usage
<b>Syntax:</b>	SSLProxyProtocol [+ -] <i>protocol</i> ...
<b>Default:</b>	SSLProxyProtocol all
<b>Context:</b>	server config, virtual host
<b>Override:</b>	Options
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive can be used to control the SSL protocol flavors mod\_ssl should use when establishing its server environment for proxy . It will only connect to servers using one of the provided protocols.

Please refer to [SSLProtocol](#) for additional information.



<b>Description:</b>	Type of remote server Certificate verification
<b>Syntax:</b>	SSLProxyVerify <i>level</i>
<b>Default:</b>	SSLProxyVerify none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the Certificate verification level for the remote server Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the remote server authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured remote server verification level after the HTTP request was read but before the HTTP response is sent.

The following levels are available for *level*:

- **none**: no remote server Certificate is required at all
- **optional**: the remote server *may* present a valid Certificate
- **require**: the remote server *has to* present a valid Certificate
- **optional\_no\_ca**: the remote server may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all servers and level **optional\_no\_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

### Example

```
SSLProxyVerify require
```



## SSLProxyVerifyDepth Directive

<b>Description:</b>	Maximum depth of CA Certificates in Remote Server Certificate verification
<b>Syntax:</b>	SSLProxyVerifyDepth <i>number</i>
<b>Default:</b>	SSLProxyVerifyDepth 1
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets how deeply mod\_ssl should verify before deciding that the remote server does not have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured remote server verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the remote server certificate. A depth of 0 means that self-signed remote server certificates are accepted only, the default depth of 1 means the remote server certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under [SSLProxyCACertificatePath](#)), etc.

**Example**

```
SSLProxyVerifyDepth 10
```



<b>Description:</b>	Pseudo Random Number Generator (PRNG) seeding source
<b>Syntax:</b>	SSLRandomSeed <i>context source [bytes]</i>
<b>Context:</b>	server config
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This configures one or more sources for seeding the Pseudo Random Number Generator (PRNG) in OpenSSL at startup time (*context* is `start up`) and/or just before a new SSL connection is established (*context* is `connect`). This directive can only be used in the global server context because the PRNG is a global facility.

The following *source* variants are available:

- `builtin`  
This is the always available builtin seeding source. It's usage consumes minimum CPU cycles under runtime and hence can be always used without drawbacks. The source used for seeding the PRNG contains of the current time, the current process id and (when applicable) a randomly chosen 1KB extract of the inter-process scoreboard structure of Apache. The drawback is that this is not really a strong source and at startup time (where the scoreboard is still not available) this source just produces a few bytes of entropy. So you should always, at least for the startup, use an additional seeding source.
- `file:/path/to/source`  
This variant uses an external file `/path/to/source` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of the file form the entropy (and *bytes* is given to `/path/to/source` as the first

argument). When *bytes* is not specified the whole file forms the entropy (and  $\emptyset$  is given to `/path/to/source` as the first argument). Use this especially at startup time, for instance with an available `/dev/random` and/or `/dev/urandom` devices (which usually exist on modern Unix derivatives like FreeBSD and Linux).

*But be careful:* Usually `/dev/random` provides only as much entropy data as it actually has, i.e. when you request 512 bytes of entropy, but the device currently has only 100 bytes available two things can happen: On some platforms you receive only the 100 bytes while on other platforms the read blocks until enough bytes are available (which can take a long time). Here using an existing `/dev/urandom` is better, because it never blocks and actually gives the amount of requested data. The drawback is just that the quality of the received data may not be the best.

On some platforms like FreeBSD one can even control how the entropy is actually generated, i.e. by which system interrupts. More details one can find under `rndcontrol(8)` on those platforms. Alternatively, when your system lacks such a random device, you can use tool like [EGD](#) (Entropy Gathering Daemon) and run it's client program with the `exec:/path/to/program/variant` (see below) or use `egd:/path/to/egd-socket` (see below).

- `exec:/path/to/program`  
This variant uses an external executable `/path/to/program` as the source for seeding the PRNG. When *bytes* is specified, only the first *bytes* number of bytes of its `stdout` contents form the entropy. When *bytes* is not specified, the entirety of the data produced on `stdout` form the entropy. Use this only at startup time when you need a

very strong seeding with the help of an external program (for instance as in the example above with the `truerand` utility you can find in the `mod_ssl` distribution which is based on the AT&T *truerand* library). Using this in the connection context slows down the server too dramatically, of course. So usually you should avoid using external programs in that context.

- `egd:/path/to/egd-socket` (Unix only)  
This variant uses the Unix domain socket of the external Entropy Gathering Daemon (EGD) (see <http://www.lothar.com/tech/crypto/>) to seed the PRNG. Use this if no random device exists on your platform.

### Example

```
SSLRandomSeed startup builtin
SSLRandomSeed startup file:/dev/random
SSLRandomSeed startup file:/dev/urandom 1024
SSLRandomSeed startup exec:/usr/local/bin/truerand 16
SSLRandomSeed connect builtin
SSLRandomSeed connect file:/dev/random
SSLRandomSeed connect file:/dev/urandom 1024
```



**Description:** Allow access only when an arbitrarily complex boolean expression is true

**Syntax:** `SSLRequire expression`

**Context:** directory, .htaccess

**Override:** AuthConfig

**Status:** Extension

**Module:** mod\_ssl

This directive specifies a general access requirement which has to be fulfilled in order to allow access. It's a very powerful directive because the requirement specification is an arbitrarily complex boolean expression containing any number of access checks.

The *expression* must match the following syntax (given as a BNF grammar notation):

```
expr ::= "true" | "false"
      | "!" expr
      | expr "&&" expr
      | expr "||" expr
      | "(" expr ")"
      | comp
```

```
comp ::= word "==" word | word "eq" word
      | word "!=" word | word "ne" word
      | word "<" word | word "lt" word
      | word "<=" word | word "le" word
      | word ">" word | word "gt" word
      | word ">=" word | word "ge" word
      | word "in" "{" wordlist "}"
      | word "=~" regex
      | word "!~" regex
```

```
wordlist ::= word
```

```

        | wordlist "," word

word ::= digit
      | cstring
      | variable
      | function

digit ::= [0-9]+
cstring ::= "... "
variable ::= "%{" varname "}"
function ::= funcname "(" funcargs ")"

```

while for varname any variable from [Table 3](#) can be used. Finally for funcname the following functions are available:

- `file(filename)`  
This function takes one string argument and expands to the contents of the file. This is especially useful for matching this contents against a regular expression, etc.

Notice that *expression* is first parsed into an internal machine representation and then evaluated in a second step. Actually, in Global and Per-Server Class context *expression* is parsed at startup time and at runtime only the machine representation is executed. For Per-Directory context this is different: here *expression* has to be parsed and immediately executed for every request.

### Example

```

SSLRequire ( %{SSL_CIPHER} !~ m/^(EXP|NULL)-/ \
and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20 ) \
or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/

```

*Standard CGI/1.0 and Apache variables:*

HTTP_USER_AGENT	PATH_INFO	AUTH_
HTTP_REFERER	QUERY_STRING	SERV
HTTP_COOKIE	REMOTE_HOST	API_
HTTP_FORWARDED	REMOTE_IDENT	TIME_
HTTP_HOST	IS_SUBREQ	TIME_
HTTP_PROXY_CONNECTION	DOCUMENT_ROOT	TIME_
HTTP_ACCEPT	SERVER_ADMIN	TIME_
HTTP:headername	SERVER_NAME	TIME_
THE_REQUEST	SERVER_PORT	TIME_
REQUEST_METHOD	SERVER_PROTOCOL	TIME_
REQUEST_SCHEME	REMOTE_ADDR	TIME_
REQUEST_URI	REMOTE_USER	ENV:'
REQUEST_FILENAME		

*SSL-related variables:*

HTTPS	SSL_CLIENT_M_VERSION	SSL_
	SSL_CLIENT_M_SERIAL	SSL_
SSL_PROTOCOL	SSL_CLIENT_V_START	SSL_
SSL_SESSION_ID	SSL_CLIENT_V_END	SSL_
SSL_CIPHER	SSL_CLIENT_S_DN	SSL_
SSL_CIPHER_EXPORT	SSL_CLIENT_S_DN_C	SSL_
SSL_CIPHER_ALGKEYSIZE	SSL_CLIENT_S_DN_ST	SSL_
SSL_CIPHER_USEKEYSIZE	SSL_CLIENT_S_DN_L	SSL_
SSL_VERSION_LIBRARY	SSL_CLIENT_S_DN_O	SSL_
SSL_VERSION_INTERFACE	SSL_CLIENT_S_DN_OU	SSL_
	SSL_CLIENT_S_DN_CN	SSL_
	SSL_CLIENT_S_DN_T	SSL_
	SSL_CLIENT_S_DN_I	SSL_
	SSL_CLIENT_S_DN_G	SSL_
	SSL_CLIENT_S_DN_S	SSL_
	SSL_CLIENT_S_DN_D	SSL_
	SSL_CLIENT_S_DN_UID	SSL_
	SSL_CLIENT_S_DN_Email	SSL_
	SSL_CLIENT_I_DN	SSL_
	SSL_CLIENT_I_DN_C	SSL_
	SSL_CLIENT_I_DN_ST	SSL_

SSL_CLIENT_I_DN_L	SSL.
SSL_CLIENT_I_DN_O	SSL.
SSL_CLIENT_I_DN_OU	SSL.
SSL_CLIENT_I_DN_CN	SSL.
SSL_CLIENT_I_DN_T	SSL.
SSL_CLIENT_I_DN_I	SSL.
SSL_CLIENT_I_DN_G	SSL.
SSL_CLIENT_I_DN_S	SSL.
SSL_CLIENT_I_DN_D	SSL.
SSL_CLIENT_I_DN_UID	SSL.
SSL_CLIENT_I_DN_Email	SSL.
SSL_CLIENT_A_SIG	SSL.
SSL_CLIENT_A_KEY	SSL.
SSL_CLIENT_CERT	SSL.
SSL_CLIENT_CERT_CHAINn	
SSL_CLIENT_VERIFY	



<b>Description:</b>	Deny access when SSL is not used for the HTTP request
<b>Syntax:</b>	SSLRequireSSL
<b>Context:</b>	directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive forbids access unless HTTP over SSL (i.e. HTTPS) is enabled for the current connection. This is very handy inside the SSL-enabled virtual host or directories for defending against configuration errors that expose stuff that should be protected. When this directive is present all requests are denied which are not using SSL.

### Example

```
SSLRequireSSL
```



**Description:** Type of the global/inter-process SSL Session Cache

**Syntax:** SSLSessionCache *type*

**Default:** SSLSessionCache none

**Context:** server config

**Status:** Extension

**Module:** mod\_ssl

This configures the storage type of the global/inter-process SSL Session Cache. This cache is an optional facility which speeds up parallel request processing. For requests to the same server process (via HTTP keep-alive), OpenSSL already caches the SSL session information locally. But because modern clients request inlined images and other data via parallel requests (usually up to four parallel requests are common) those requests are served by *different* pre-forked server processes. Here an inter-process cache helps to avoid unnecessary session handshakes.

The following two storage *types* are currently supported:

- none  
This is the default and just disables the global/inter-process Session Cache. There is no drawback in functionality, but a noticeable speed penalty can be observed.
- dbm:/path/to/datafile  
This makes use of a DBM hashfile on the local disk to synchronize the local OpenSSL memory caches of the server processes. The slight increase in I/O on the server results in a visible request speedup for your clients, so this type of storage is generally recommended.
- shm:/path/to/datafile[(size)]

This makes use of a high-performance hash table (approx. *size* bytes in size) inside a shared memory segment in RAM (established via `/path/to/datafile`) to synchronize the local OpenSSL memory caches of the server processes. This storage type is not available on all platforms.

### Examples

```
SSLSessionCache dbm:/usr/local/apache/logs/ssl_gcache_data
SSLSessionCache
shm:/usr/local/apache/logs/ssl_gcache_data(512000)
```



<b>Description:</b>	Number of seconds before an SSL session expires in the Session Cache
<b>Syntax:</b>	SSLSessionCacheTimeout <i>seconds</i>
<b>Default:</b>	SSLSessionCacheTimeout 300
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the timeout in seconds for the information stored in the global/inter-process SSL Session Cache and the OpenSSL internal memory cache. It can be set as low as 15 for testing, but should be set to higher values like 300 in real life.

### Example

```
SSLSessionCacheTimeout 600
```



<b>Description:</b>	Variable name to determine user name
<b>Syntax:</b>	SSLUserName <i>varname</i>
<b>Context:</b>	server config, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl
<b>Compatibility:</b>	Available in Apache 2.0.51 and later

This directive sets the "user" field in the Apache request object. This is used by lower modules to identify the user with a character string. In particular, this may cause the environment variable REMOTE\_USER to be set. The *varname* can be any of the [SSL environment variables](#).

### Example

```
SSLUserName SSL_CLIENT_S_DN_CN
```



<b>Description:</b>	Type of Client Certificate verification
<b>Syntax:</b>	SSLVerifyClient <i>level</i>
<b>Default:</b>	SSLVerifyClient none
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

This directive sets the Certificate verification level for the Client Authentication. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification level after the HTTP request was read but before the HTTP response is sent.

The following levels are available for *level*:

- **none**: no client Certificate is required at all
- **optional**: the client *may* present a valid Certificate
- **require**: the client *has to* present a valid Certificate
- **optional\_no\_ca**: the client may present a valid Certificate but it need not to be (successfully) verifiable.

In practice only levels **none** and **require** are really interesting, because level **optional** doesn't work with all browsers and level **optional\_no\_ca** is actually against the idea of authentication (but can be used to establish SSL test pages, etc.)

### Example

```
SSLVerifyClient require
```



<b>Description:</b>	Maximum depth of CA Certificates in Client Certificate verification
<b>Syntax:</b>	SSLVerifyDepth <i>number</i>
<b>Default:</b>	SSLVerifyDepth 1
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	AuthConfig
<b>Status:</b>	Extension
<b>Module:</b>	mod_ssl

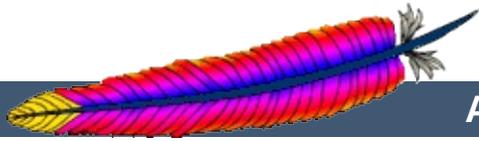
This directive sets how deeply mod\_ssl should verify before deciding that the clients don't have a valid certificate. Notice that this directive can be used both in per-server and per-directory context. In per-server context it applies to the client authentication process used in the standard SSL handshake when a connection is established. In per-directory context it forces a SSL renegotiation with the reconfigured client verification depth after the HTTP request was read but before the HTTP response is sent.

The depth actually is the maximum number of intermediate certificate issuers, i.e. the number of CA certificates which are max allowed to be followed while verifying the client certificate. A depth of 0 means that self-signed client certificates are accepted only, the default depth of 1 means the client certificate can be self-signed or has to be signed by a CA which is directly known to the server (i.e. the CA's certificate is under [SSLCACertificatePath](#)), etc.

### Example

```
SSLVerifyDepth 10
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_status

<b>Description:</b>	Provides information on server activity and performance
<b>Status:</b>	Base
<b>Module Identifier:</b>	status_module
<b>Source File:</b>	mod_status.c

### Summary

The Status module allows a server administrator to find out how well their server is performing. A HTML page is presented that gives the current server statistics in an easily readable form. If required this page can be made to automatically refresh (given a compatible browser). Another page gives a simple machine-readable list of the current server state.

The details given are:

- The number of worker serving requests
- The number of idle worker
- The status of each worker, the number of requests that worker has performed and the total number of bytes served by the worker (\*)
- A total number of accesses and byte count served (\*)
- The time the server was started/restarted and the time it has been running for
- Averages giving the number of requests per second, the number of bytes served per second and the average number of bytes per request (\*)
- The current percentage CPU used by each worker and in total by Apache (\*)
- The current hosts and requests being processed (\*)

The lines marked "\*" are only available if ExtendedStatus is On.



## Enabling Status Support

To enable status reports only for browsers from the foo.com domain add this code to your httpd.conf configuration file

```
<Location /server-status>  
SetHandler server-status  
  
Order Deny,Allow  
Deny from all  
Allow from .foo.com  
</Location>
```

You can now access server statistics by using a Web browser to access the page `http://your.server.name/server-status`



## Automatic Updates

You can get the status page to update itself automatically if you have a browser that supports "refresh". Access the page `http://your.server.name/server-status?refresh=N` to refresh the page every N seconds.



## MACHINE-READABLE STATUS FILE

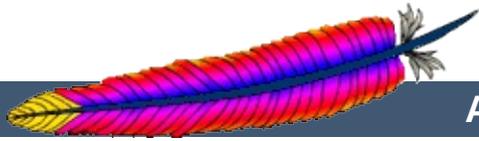
A machine-readable version of the status file is available by accessing the page `http://your.server.name/server-status?auto`. This is useful when automatically run, see the Perl program in the `/support` directory of Apache, `log_server_status`.

**It should be noted that if `mod_status` is compiled into the server, its handler capability is available in *all* configuration files, including *per-directory* files (e.g., `.htaccess`). This may have security-related ramifications for your site.**



<b>Description:</b>	Keep track of extended status information for each request
<b>Syntax:</b>	ExtendedStatus On Off
<b>Default:</b>	ExtendedStatus Off
<b>Context:</b>	server config
<b>Status:</b>	Base
<b>Module:</b>	mod_status
<b>Compatibility:</b>	ExtendedStatus is only available in Apache 1.3.2 and later.

This setting applies to the entire server, and cannot be enabled or disabled on a virtualhost-by-virtualhost basis. The collection of extended status information can slow down the server.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_suexec

<b>Description:</b>	Allows CGI scripts to run as a specified user and Group
<b>Status:</b>	Extension
<b>Module Identifier:</b>	suexec_module
<b>Source File:</b>	mod_suexec.c
<b>Compatibility:</b>	Available in Apache 2.0 and later

## Summary

This module, in combination with the [suexec](#) support program allows CGI scripts to run as a specified user and Group.

## See also

[SuEXEC support](#)



## SuexecUserGroup Directive

<b>Description:</b>	User and group for CGI programs to run as
<b>Syntax:</b>	<code>SuexecUserGroup <i>User Group</i></code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	mod_suexec
<b>Compatibility:</b>	SuexecUserGroup is only available in 2.0 and later.

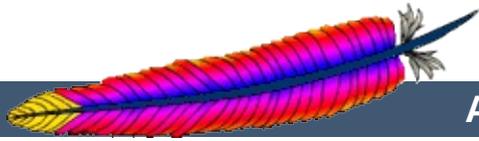
The `SuexecUserGroup` directive allows you to specify a user and group for CGI programs to run as. Non-CGI requests are still processed with the user specified in the `User` directive. This directive replaces the Apache 1.3 configuration of using the `User` and `Group` directives inside of `VirtualHosts`.

**Example**

```
SuexecUserGroup nobody nogroup
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_unique\_id

<b>Description:</b>	Provides an environment variable with a unique identifier for each request
<b>Status:</b>	Extension
<b>Module Identifier:</b>	unique_id_module
<b>Source File:</b>	mod_unique_id.c

### Summary

This module provides a magic token for each request which is guaranteed to be unique across "all" requests under very specific conditions. The unique identifier is even unique across multiple machines in a properly configured cluster of machines. The environment variable `UNIQUE_ID` is set to the identifier for each request. Unique identifiers are useful for various reasons which are beyond the scope of this document.



First a brief recap of how the Apache server works on Unix machines. This feature currently isn't supported on Windows NT. On Unix machines, Apache creates several children, the children process requests one at a time. Each child can serve multiple requests in its lifetime. For the purpose of this discussion, the children don't share any data with each other. We'll refer to the children as *httpd processes*.

Your website has one or more machines under your administrative control, together we'll call them a cluster of machines. Each machine can possibly run multiple instances of Apache. All of these collectively are considered "the universe", and with certain assumptions we'll show that in this universe we can generate unique identifiers for each request, without extensive communication between machines in the cluster.

The machines in your cluster should satisfy these requirements. (Even if you have only one machine you should synchronize its clock with NTP.)

- The machines' times are synchronized via NTP or other network time protocol.
- The machines' hostnames all differ, such that the module can do a hostname lookup on the hostname and receive a different IP address for each machine in the cluster.

As far as operating system assumptions go, we assume that pids (process ids) fit in 32-bits. If the operating system uses more than 32-bits for a pid, the fix is trivial but must be performed in the code.

Given those assumptions, at a single point in time we can identify any httpd process on any machine in the cluster from all other httpd processes. The machine's IP address and the pid of the httpd process are sufficient to do this. So in order to generate

unique identifiers for requests we need only distinguish between different points in time.

To distinguish time we will use a Unix timestamp (seconds since January 1, 1970 UTC), and a 16-bit counter. The timestamp has only one second granularity, so the counter is used to represent up to 65536 values during a single second. The quadruple ( *ip\_addr*, *pid*, *time\_stamp*, *counter* ) is sufficient to enumerate 65536 requests per second per httpd process. There are issues however with pid reuse over time, and the counter is used to alleviate this issue.

When an httpd child is created, the counter is initialized with ( current microseconds divided by 10 ) modulo 65536 (this formula was chosen to eliminate some variance problems with the low order bits of the microsecond timers on some systems). When a unique identifier is generated, the time stamp used is the time the request arrived at the web server. The counter is incremented every time an identifier is generated (and allowed to roll over).

The kernel generates a pid for each process as it forks the process, and pids are allowed to roll over (they're 16-bits on many Unixes, but newer systems have expanded to 32-bits). So over time the same pid will be reused. However unless it is reused within the same second, it does not destroy the uniqueness of our quadruple. That is, we assume the system does not spawn 65536 processes in a one second interval (it may even be 32768 processes on some Unixes, but even this isn't likely to happen).

Suppose that time repeats itself for some reason. That is, suppose that the system's clock is screwed up and it revisits a past time (or it is too far forward, is reset correctly, and then revisits the future time). In this case we can easily show that we can get pid and time stamp reuse. The choice of initializer for the counter is intended to help defeat this. Note that we really want a random number to

initialize the counter, but there aren't any readily available numbers on most systems (*i.e.*, you can't use `rand()` because you need to seed the generator, and can't seed it with the time because time, at least at one second resolution, has repeated itself). This is not a perfect defense.

How good a defense is it? Suppose that one of your machines serves at most 500 requests per second (which is a very reasonable upper bound at this writing, because systems generally do more than just shovel out static files). To do that it will require a number of children which depends on how many concurrent clients you have. But we'll be pessimistic and suppose that a single child is able to serve 500 requests per second. There are 1000 possible starting counter values such that two sequences of 500 requests overlap. So there is a 1.5% chance that if time (at one second resolution) repeats itself this child will repeat a counter value, and uniqueness will be broken. This was a very pessimistic example, and with real world values it's even less likely to occur. If your system is such that it's still likely to occur, then perhaps you should make the counter 32 bits (by editing the code).

You may be concerned about the clock being "set back" during summer daylight savings. However this isn't an issue because the times used here are UTC, which "always" go forward. Note that x86 based Unixes may need proper configuration for this to be true -- they should be configured to assume that the motherboard clock is on UTC and compensate appropriately. But even still, if you're running NTP then your UTC time will be correct very shortly after reboot.

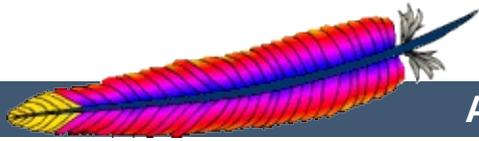
The `UNIQUE_ID` environment variable is constructed by encoding the 112-bit (32-bit IP address, 32 bit pid, 32 bit time stamp, 16 bit counter) quadruple using the alphabet `[A-Za-z0-9@-]` in a manner similar to MIME base64 encoding, producing 19 characters. The MIME base64 alphabet is actually `[A-Za-z0-`

9+/] however + and / need to be specially encoded in URLs, which makes them less desirable. All values are encoded in network byte ordering so that the encoding is comparable across architectures of different byte ordering. The actual ordering of the encoding is: time stamp, IP address, pid, counter. This ordering has a purpose, but it should be emphasized that applications should not dissect the encoding. Applications should treat the entire encoded UNIQUE\_ID as an opaque token, which can be compared against other UNIQUE\_IDs for equality only.

The ordering was chosen such that it's possible to change the encoding in the future without worrying about collision with an existing database of UNIQUE\_IDs. The new encodings should also keep the time stamp as the first element, and can otherwise use the same alphabet and bit length. Since the time stamps are essentially an increasing sequence, it's sufficient to have a *flag second* in which all machines in the cluster stop serving and request, and stop using the old encoding format. Afterwards they can resume requests and begin issuing the new encodings.

This we believe is a relatively portable solution to this problem. It can be extended to multithreaded systems like Windows NT, and can grow with future needs. The identifiers generated have essentially an infinite life-time because future identifiers can be made longer as required. Essentially no communication is required between machines in the cluster (only NTP synchronization is required, which is low overhead), and no communication between httpd processes is required (the communication is implicit in the pid value assigned by the kernel). In very specific situations the identifier can be shortened, but more information needs to be assumed (for example the 32-bit IP address is overkill for any site, but there is no portable shorter replacement for it).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_userdir

<b>Description:</b>	User-specific directories
<b>Status:</b>	Base
<b>Module Identifier:</b>	userdir_module
<b>Source File:</b>	mod_userdir.c

## Summary

This module allows user-specific directories to be accessed using the `http://example.com/~user/` syntax.

## See also

[Mapping URLs to the Filesystem](#)  
[public\\_html tutorial](#)



<b>Description:</b>	Location of the user-specific directories
<b>Syntax:</b>	UserDir <i>directory-filename</i>
<b>Default:</b>	UserDir public_html
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Base
<b>Module:</b>	mod_userdir

The `UserDir` directive sets the real directory in a user's home directory to use when a request for a document for a user is received. *Directory-filename* is one of the following:

- The name of a directory or a pattern such as those shown below.
- The keyword `disabled`. This turns off *all* username-to-directory translations except those explicitly named with the `enabled` keyword (see below).
- The keyword `disabled` followed by a space-delimited list of usernames. Usernames that appear in such a list will *never* have directory translation performed, even if they appear in an `enabled` clause.
- The keyword `enabled` followed by a space-delimited list of usernames. These usernames will have directory translation performed even if a global `disabled` is in effect, but not if they also appear in a `disabled` clause.

If neither the `enabled` nor the `disabled` keywords appear in the `UserDir` directive, the argument is treated as a filename pattern, and is used to turn the name into a directory specification. A request for `http://www.foo.com/~bob/one/two.html` will be translated to:

<b>UserDir directive used</b> <b>Translated path</b>
--

```
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
```

The following directives will send redirects to the client:

UserDir directive used	Translated path
UserDir	http://www.foo.com/users/bob/one/two.h
http://www.foo.com/users	
UserDir	http://www.foo.com/bob/usr/one/two.htm
http://www.foo.com/*/usr	
UserDir	http://www.foo.com/~bob/one/two.html
http://www.foo.com/~*/	

**Be careful when using this directive; for instance, "UserDir ./" would map "~/root" to "/" - which is probably undesirable. It is strongly recommended that your configuration include a "UserDir disabled root" declaration. See also the [Directory](#) directive and the [Security Tips](#) page for more information.**

Additional examples:

To allow a few users to have UserDir directories, but not anyone else, use the following:

```
UserDir disabled
UserDir enabled user1 user2 user3
```

To allow most users to have UserDir directories, but deny this to a few, use the following:

```
UserDir enabled
UserDir disabled user4 user5 user6
```

It is also possible to specify alternative user directories. If you use a command like:

```
Userdir public_html /usr/web http://www.foo.com/
```

With a request for `http://www.foo.com/~bob/one/two.html`, will try to find the page at `~bob/public_html/one/two.html` first, then `/usr/web/bob/one/two.html`, and finally it will send a redirect to `http://www.foo.com/bob/one/two.html`.

If you add a redirect, it must be the last alternative in the list. Apache cannot determine if the redirect succeeded or not, so if you have the redirect earlier in the list, that will always be the alternative that is used.

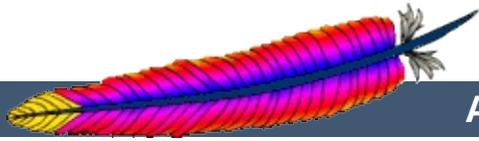
## See also

- [public\\_html tutorial](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

## Apache Module mod\_usertrack

<b>Description:</b>	<i>Clickstream</i> logging of user activity on a site
<b>Status:</b>	Extension
<b>Module Identifier:</b>	usertrack_module
<b>Source File:</b>	mod_usertrack.c

### Summary

Previous releases of Apache have included a module which generates a 'clickstream' log of user activity on a site using cookies. This was called the "cookies" module, mod\_cookies. In Apache 1.2 and later this module has been renamed the "user tracking" module, mod\_usertrack. This module has been simplified and new directives added.



## Logging

Previously, the cookies module (now the user tracking module) did its own logging, using the `CookieLog` directive. In this release, this module does no logging at all. Instead, a configurable log format file should be used to log user click-streams. This is possible because the logging module now allows multiple log files. The cookie itself is logged by using the text `%{cookie}`n in the log file format. For example:

```
CustomLog logs/clickstream "%{cookie}n %r %t"
```

For backward compatibility the configurable log module implements the old `CookieLog` directive, but this should be upgraded to the above `CustomLog` directive.



(the following is from message  
<022701bda43d\$9d32bbb0\$1201a8c0@christian.office.sane.com>  
in the new-httpd archives)

From: "Christian Allen" <christian@sane.com>  
Subject: Re: Apache Y2K bug in mod\_usertrack.c  
Date: Tue, 30 Jun 1998 11:41:56 -0400

Did some work with cookies and dug up some info t

True, Netscape claims that the correct format NOW  
four digit dates do in fact work... for Netscape  
is. However, 3.x and below do NOT accept them.  
originally had a 2-digit standard, and then with  
probably a few complaints, changed to a four digit  
Fortunately, 4.x also understands the 2-digit form  
ensure that your expiration date is legible to th  
use 2-digit dates.

However, this does not limit expiration dates to  
an expiration year of "13", for example, it is in  
1913! In fact, you can use an expiration year of  
understood as "2037" by both MSIE and Netscape ve  
about versions previous to those). Not sure why  
particular year as its cut-off point, but my gues  
to UNIX's 2038 problem. Netscape/MSIE 4.x seem t  
2-digit years beyond that, at least until "50" fo  
understand up until about "70", but not for sure)

Summary: Mozilla 3.x and up understands two digi  
(2037). Mozilla 4.x understands up until at leas  
form, but also understands 4-digit years, which c  
9999. Your best bet for sending a long-life cook  
time late in the year "37".



<b>Description:</b>	The domain to which the tracking cookie applies
<b>Syntax:</b>	CookieDomain <i>domain</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive controls the setting of the domain to which the tracking cookie applies. If not present, no domain is included in the cookie header field.

The domain string **must** begin with a dot, and **must** include at least one embedded dot. That is, `.foo.com` is legal, but `foo.bar.com` and `.com` are not.

Most browsers in use today will not allow cookies to be set for a two-part top level domain, such as `.co.uk`, although such a domain ostensibly fulfills the requirements above. These domains are equivalent to top level domains such as `.com`, and allowing such cookies may be a security risk. Thus, if you are under a two-part top level domain, you should still use your actual domain, as you would with any other top level domain (for example, use `.foo.co.uk`).



<b>Description:</b>	Expiry time for the tracking cookie
<b>Syntax:</b>	CookieExpires <i>expiry-period</i>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

When used, this directive sets an expiry time on the cookie generated by the usertrack module. The *expiry-period* can be given either as a number of seconds, or in the format such as "2 weeks 3 days 7 hours". Valid denominations are: years, months, weeks, days, hours, minutes and seconds. If the expiry time is in any format other than one number indicating the number of seconds, it must be enclosed by double quotes.

If this directive is not used, cookies last only for the current browser session.



<b>Description:</b>	Name of the tracking cookie
<b>Syntax:</b>	CookieName <i>token</i>
<b>Default:</b>	CookieName Apache
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive allows you to change the name of the cookie this module uses for its tracking purposes. By default the cookie is named "Apache".

You must specify a valid cookie name; results are unpredictable if you use a name containing unusual characters. Valid characters include A-Z, a-z, 0-9, "\_", and "-".



<b>Description:</b>	Format of the cookie header field
<b>Syntax:</b>	CookieStyle <i>Netscape   Cookie   Cookie2   RFC2109   RFC2961</i>
<b>Default:</b>	CookieStyle Netscape
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

This directive controls the format of the cookie header field. The three formats allowed are:

- **Netscape**, which is the original but now deprecated syntax. This is the default, and the syntax Apache has historically used.
- **Cookie** or **RFC2109**, which is the syntax that superseded the Netscape syntax.
- **Cookie2** or **RFC2965**, which is the most current cookie syntax.

Not all clients can understand all of these formats. but you should use the newest one that is generally acceptable to your users' browsers. At the time of writing, most browsers only fully support CookieStyle Netscape.



## CookieTracking Directive

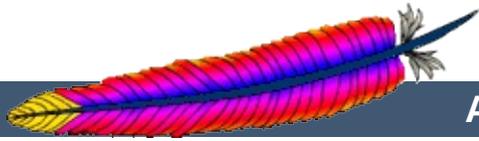
<b>Description:</b>	Enables tracking cookie
<b>Syntax:</b>	CookieTracking on off
<b>Default:</b>	CookieTracking off
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	FileInfo
<b>Status:</b>	Extension
<b>Module:</b>	mod_usertrack

When [mod\\_usertrack](#) is loaded, and `CookieTracking on` is set, Apache will send a user-tracking cookie for all new requests. This directive can be used to turn this behavior on or off on a per-server or per-directory basis. By default, enabling [mod\\_usertrack](#) will **not** activate cookies.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module mod\_version

<b>Description:</b>	Version dependent configuration
<b>Status:</b>	Extension
<b>Module Identifier:</b>	version_module
<b>Source File:</b>	mod_version.c
<b>Compatibility:</b>	Available in version 2.0.56 and later

## Summary

This module is designed for the use in test suites and large networks which have to deal with different httpd versions and different configurations. It provides a new container -- `<IfVersion>`, which allows a flexible version checking including numeric comparisons and regular expressions.

## Examples

```
<IfVersion 2.1.0>
  # current httpd version is exactly 2.1.0
</IfVersion>

<IfVersion >= 2.2>
  # use really new features :-)
</IfVersion>
```

See below for further possibilities.



<b>Description:</b>	contains version dependent configuration
<b>Syntax:</b>	<code>&lt;IfVersion [[!]operator] version&gt; ... &lt;/IfVersion&gt;</code>
<b>Context:</b>	server config, virtual host, directory, .htaccess
<b>Override:</b>	All
<b>Status:</b>	Extension
<b>Module:</b>	mod_version

The `<IfVersion>` section encloses configuration directives which are executed only if the `httpd` version matches the desired criteria. For normal (numeric) comparisons the `version` argument has the format `major[.minor[.patch]]`, e.g. `2.1.0` or `2.2.minor` and `patch` are optional. If these numbers are omitted, they are assumed to be zero. The following numerical *operators* are possible:

<i>operator</i>	<i>description</i>
<code>=</code> or <code>==</code>	httpd version is equal
<code>&gt;</code>	httpd version is greater than
<code>&gt;=</code>	httpd version is greater or equal
<code>&lt;</code>	httpd version is less than
<code>&lt;=</code>	httpd version is less or equal

### Example

```
<IfVersion >= 2.1>
  # this happens only in versions greater or
  # equal 2.1.0.
</IfVersion>
```

Besides the numerical comparison it is possible to match a regular expression against the `httpd` version. There are two ways to write it:

<i>operator</i>	<b>description</b>
= or ==	<i>version</i> has the form <i>/regex/</i>
~	<i>version</i> has the form <i>regex</i>

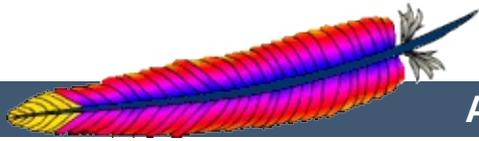
### Example

```
<IfVersion = /^2.1.[01234]$/>  
  # e.g. workaround for buggy versions  
</IfVersion>
```

In order to reverse the meaning, all operators can be preceded by an exclamation mark (!):

```
<IfVersion !~ ^2.1.[01234]$/>  
  # not for those versions  
</IfVersion>
```

If the *operator* is omitted, it is assumed to be =.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Modules](#)

# Apache Module `mod_vhost_alias`

<b>Description:</b>	Provides for dynamically configured mass virtual hosting
<b>Status:</b>	Extension
<b>Module Identifier:</b>	<code>vhost_alias_module</code>
<b>Source File:</b>	<code>mod_vhost_alias.c</code>

## Summary

This module creates dynamically configured virtual hosts, by allowing the IP address and/or the `Host` header of the HTTP request to be used as part of the pathname to determine what files to serve. This allows for easy use of a huge number of virtual hosts with similar configurations.

### Note

If `mod_alias` or `mod_userdir` are used for translating URIs to filenames, they will override the directives of `mod_vhost_alias` described below. For example, the following configuration will map `/cgi-bin/script.pl` to `/usr/local/apache2/cgi-bin/bin/script.pl` in all cases:

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/  
VirtualScriptAlias /never/found/%0/cgi-bin/
```

## See also

[UseCanonicalName](#)

[Dynamically configured mass virtual hosting](#)



## Directory Name Interpolation

All the directives in this module interpolate a string into a pathname. The interpolated string (henceforth called the "name") may be either the server name (see the [UseCanonicalName](#) directive for details on how this is determined) or the IP address of the virtual host on the server in dotted-quad format. The interpolation is controlled by specifiers inspired by `printf` which have a number of formats:

<code>%%</code>	insert a %
<code>%p</code>	insert the port number of the virtual host
<code>%N.M</code>	insert (part of) the name

N and M are used to specify substrings of the name. N selects from the dot-separated components of the name, and M selects characters within whatever N has selected. M is optional and defaults to zero if it isn't present; the dot must be present if and only if M is present. The interpretation is as follows:

<code>0</code>	the whole name
<code>1</code>	the first part
<code>2</code>	the second part
<code>-1</code>	the last part
<code>-2</code>	the penultimate part
<code>2+</code>	the second and all subsequent parts
<code>-2+</code>	the penultimate and all preceding parts
<code>1+</code> and <code>-1+</code>	the same as <code>0</code>

If N or M is greater than the number of parts available a single underscore is interpolated.



For simple name-based virtual hosts you might use the following directives in your server configuration file:

```
UseCanonicalName Off
VirtualDocumentRoot /usr/local/apache/vhosts/%0
```

A request for  
`http://www.example.com/directory/file.html` will be satisfied by the file  
`/usr/local/apache/vhosts/www.example.com/directory`

For a very large number of virtual hosts it is a good idea to arrange the files to reduce the size of the vhosts directory. To do this you might use the following in your configuration file:

```
UseCanonicalName Off
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

A request for  
`http://www.domain.example.com/directory/file.html`  
will be satisfied by the file  
`/usr/local/apache/vhosts/example.com/d/o/m/domain`

A more even spread of files can be achieved by hashing from the end of the name, for example:

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/n/i/a/domain`

Alternatively you might use:

```
VirtualDocumentRoot
/usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

The example request would come from  
`/usr/local/apache/vhosts/example.com/d/o/m/ain/di`

For IP-based virtual hosting you might use the following in your configuration file:

```
UseCanonicalName DNS
VirtualDocumentRootIP /usr/local/apache/vhosts/%1/%2/%3/%4/docs
VirtualScriptAliasIP /usr/local/apache/vhosts/%1/%2/%3/%4/cgi-
bin
```

A request for  
`http://www.domain.example.com/directory/file.html`  
would be satisfied by the file  
`/usr/local/apache/vhosts/10/20/30/40/docs/directo`  
if the IP address of `www.domain.example.com` were  
10.20.30.40. A request for  
`http://www.domain.example.com/cgi-bin/script.pl`  
would be satisfied by executing the program  
`/usr/local/apache/vhosts/10/20/30/40/cgi-`  
`bin/script.pl`.

If you want to include the `.` character in a  
`VirtualDocumentRoot` directive, but it clashes with a `%`  
directive, you can work around the problem in the following way:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%2.0.%3.0
```

A request for  
`http://www.domain.example.com/directory/file.html`  
will be satisfied by the file  
`/usr/local/apache/vhosts/domain.example/directory`

The LogFormat directives %V and %A are useful in conjunction with this module.



<b>Description:</b>	Dynamically configure the location of the document root for a given virtual host
<b>Syntax:</b>	<code>VirtualDocumentRoot <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualDocumentRoot none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualDocumentRoot` directive allows you to determine where Apache will find your documents based on the value of the server name. The result of expanding *interpolated-directory* is used as the root of the document tree in a similar manner to the `DocumentRoot` directive's argument. If *interpolated-directory* is none then `VirtualDocumentRoot` is turned off. This directive cannot be used in the same context as `VirtualDocumentRootIP`.



<b>Description:</b>	Dynamically configure the location of the document root for a given virtual host
<b>Syntax:</b>	<code>VirtualDocumentRootIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualDocumentRootIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualDocumentRootIP` directive is like the `VirtualDocumentRoot` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.



## VIRTUALSCRIPTALIAS DIRECTIVE

<b>Description:</b>	Dynamically configure the location of the CGI directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAlias <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAlias none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualScriptAlias` directive allows you to determine where Apache will find CGI scripts in a similar manner to `VirtualDocumentRoot` does for other documents. It matches requests for URIs starting `/cgi-bin/`, much like `ScriptAlias /cgi-bin/` would.



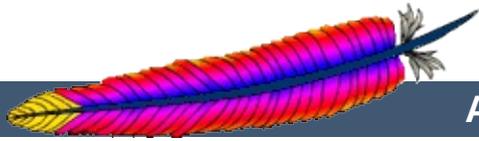
<b>Description:</b>	Dynamically configure the location of the cgi directory for a given virtual host
<b>Syntax:</b>	<code>VirtualScriptAliasIP <i>interpolated-directory</i> none</code>
<b>Default:</b>	<code>VirtualScriptAliasIP none</code>
<b>Context:</b>	server config, virtual host
<b>Status:</b>	Extension
<b>Module:</b>	<code>mod_vhost_alias</code>

The `VirtualScriptAliasIP` directive is like the `VirtualScriptAlias` directive, except that it uses the IP address of the server end of the connection for directory interpolation instead of the server name.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

# Apache 1.3 API notes

## Warning

This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

These are some notes on the Apache API and the data structures you have to deal with, *etc.* They are not yet nearly complete, but hopefully, they will help you get your bearings. Keep in mind that the API is still subject to change as we gain experience with it. (See the TODO file for what *might* be coming). However, it will be easy to adapt modules to any changes that are made. (We have more modules to adapt than you do).

A few notes on general pedagogical style here. In the interest of conciseness, all structure declarations here are incomplete -- the real ones have more slots that I'm not telling you about. For the most part, these are reserved to one component of the server core or another, and should be altered by modules with caution. However, in some cases, they really are things I just haven't gotten around to yet. Welcome to the bleeding edge.

Finally, here's an outline, to give you some bare idea of what's coming up, and in what order:

- [Basic concepts.](#)
  - [Handlers, Modules, and Requests](#)
  - [A brief tour of a module](#)
- [How handlers work](#)
  - [A brief tour of the request\\_rec](#)
  - [Where request\\_rec structures come from](#)

- [Handling requests, declining, and returning error codes](#)
- [Special considerations for response handlers](#)
- [Special considerations for authentication handlers](#)
- [Special considerations for logging handlers](#)
  
- [Resource allocation and resource pools](#)
- [Configuration, commands and the like](#)
  - [Per-directory configuration structures](#)
  - [Command handling](#)
  - [Side notes --- per-server configuration, virtual servers, etc.](#)



## Basic Concepts

We begin with an overview of the basic concepts behind the API, and how they are manifested in the code.

### Handlers, Modules, and Requests

Apache breaks down request handling into a series of steps, more or less the same way the Netscape server API does (although this API has a few more stages than NetSite does, as hooks for stuff I thought might be useful in the future). These are:

- URI -> Filename translation
- Auth ID checking [is the user who they say they are?]
- Auth access checking [is the user authorized *here*?]
- Access checking other than auth
- Determining MIME type of the object requested
- `Fixups' -- there aren't any of these yet, but the phase is intended as a hook for possible extensions like [SetEnv](#), which don't really fit well elsewhere.
- Actually sending a response back to the client.
- Logging the request

These phases are handled by looking at each of a succession of *modules*, looking to see if each of them has a handler for the phase, and attempting invoking it if so. The handler can typically do one of three things:

- *Handle* the request, and indicate that it has done so by returning the magic constant OK.
- *Decline* to handle the request, by returning the magic integer constant DECLINED. In this case, the server behaves in all respects as if the handler simply hadn't been there.
- Signal an error, by returning one of the HTTP error codes. This terminates normal handling of the request, although an `ErrorDocument` may be invoked to try to mop up, and it will be

logged in any case.

Most phases are terminated by the first module that handles them; however, for logging, `fixups', and non-access authentication checking, all handlers always run (barring an error). Also, the response phase is unique in that modules may declare multiple handlers for it, via a dispatch table keyed on the MIME type of the requested object. Modules may declare a response-phase handler which can handle *any* request, by giving it the key `*/*` (*i.e.*, a wildcard MIME type specification). However, wildcard handlers are only invoked if the server has already tried and failed to find a more specific response handler for the MIME type of the requested object (either none existed, or they all declined).

The handlers themselves are functions of one argument (a `request_rec` structure. *vide infra*), which returns an integer, as above.

## A brief tour of a module

At this point, we need to explain the structure of a module. Our candidate will be one of the messier ones, the CGI module -- this handles both CGI scripts and the [ScriptAlias](#) config file command. It's actually a great deal more complicated than most modules, but if we're going to have only one example, it might as well be the one with its fingers in every place.

Let's begin with handlers. In order to handle the CGI scripts, the module declares a response handler for them. Because of [ScriptAlias](#), it also has handlers for the name translation phase (to recognize [ScriptAliased](#) URIs), the type-checking phase (any [ScriptAliased](#) request is typed as a CGI script).

The module needs to maintain some per (virtual) server information, namely, the [ScriptAliases](#) in effect; the module

structure therefore contains pointers to a functions which builds these structures, and to another which combines two of them (in case the main server and a virtual server both have [ScriptAliases](#) declared).

Finally, this module contains code to handle the [ScriptAlias](#) command itself. This particular module only declares one command, but there could be more, so modules have *command tables* which declare their commands, and describe where they are permitted, and how they are to be invoked.

A final note on the declared types of the arguments of some of these commands: a *pool* is a pointer to a *resource pool* structure; these are used by the server to keep track of the memory which has been allocated, files opened, *etc.*, either to service a particular request, or to handle the process of configuring itself. That way, when the request is over (or, for the configuration pool, when the server is restarting), the memory can be freed, and the files closed, *en masse*, without anyone having to write explicit code to track them all down and dispose of them. Also, a *cmd\_parms* structure contains various information about the config file being read, and other status information, which is sometimes of use to the function which processes a config-file command (such as [ScriptAlias](#)). With no further ado, the module itself:

```
/* Declarations of handlers. */

int translate_scriptalias (request_rec *);
int type_scriptalias (request_rec *);
int cgi_handler (request_rec *);

/* Subsidiary dispatch table for response-phase
 * handlers, by MIME type */

handler_rec cgi_handlers[] = {
    { "application/x-httpd-cgi", cgi_handler },
    { NULL }
};
```

```

/* Declarations of routines to manipulate the
 * module's configuration info. Note that these are
 * returned, and passed in, as void *'s; the server
 * core keeps track of them, but it doesn't, and can't,
 * know their internal structure.
 */

void *make_cgi_server_config (pool *);
void *merge_cgi_server_config (pool *, void *, void *);

/* Declarations of routines to handle config-file commands */

extern char *script_alias(cmd_parms *, void *per_dir_config,
char *fake, char *real);

command_rec cgi_cmds[] = {
    { "ScriptAlias", script_alias, NULL, RSRC_CONF, TAKE2,
      "a fakename and a realname"},
    { NULL }
};

module cgi_module = {
    STANDARD_MODULE_STUFF,
    NULL, /* initializer */
    NULL, /* dir config creator */
    NULL, /* dir merger */
    make_cgi_server_config, /* server config */
    merge_cgi_server_config, /* merge server config */
    cgi_cmds, /* command table */
    cgi_handlers, /* handlers */
    translate_scriptalias, /* filename translation */
    NULL, /* check_user_id */
    NULL, /* check_auth */
    NULL, /* check_access */
    type_scriptalias, /* type_checker */
    NULL, /* fixups */
    NULL, /* logger */
    NULL /* header parser */
};

```



The sole argument to handlers is a `request_rec` structure. This structure describes a particular request which has been made to the server, on behalf of a client. In most cases, each connection to the client generates only one `request_rec` structure.

## A brief tour of the `request_rec`

The `request_rec` contains pointers to a resource pool which will be cleared when the server is finished handling the request; to structures containing per-server and per-connection information, and most importantly, information on the request itself.

The most important such information is a small set of character strings describing attributes of the object being requested, including its URI, filename, content-type and content-encoding (these being filled in by the translation and type-check handlers which handle the request, respectively).

Other commonly used data items are tables giving the MIME headers on the client's original request, MIME headers to be sent back with the response (which modules can add to at will), and environment variables for any subprocesses which are spawned off in the course of servicing the request. These tables are manipulated using the `ap_table_get` and `ap_table_set` routines.

Note that the Content - type header value *cannot* be set by module content-handlers using the `ap_table_*` ( ) routines. Rather, it is set by pointing the `content_type` field in the `request_rec` structure to an appropriate string. *e.g.*,

```
r->content_type = "text/html";
```

Finally, there are pointers to two data structures which, in turn, point to per-module configuration structures. Specifically, these hold pointers to the data structures which the module has built to describe the way it has been configured to operate in a given directory (via `.htaccess` files or [<Directory>](#) sections), for private data it has built in the course of servicing the request (so modules' handlers for one phase can pass `notes' to their handlers for other phases). There is another such configuration vector in the `server_rec` data structure pointed to by the `request_rec`, which contains per (virtual) server configuration data.

Here is an abridged declaration, giving the fields most commonly used:

```
struct request_rec {

    pool *pool;
    conn_rec *connection;
    server_rec *server;

    /* What object is being requested */

    char *uri;
    char *filename;
    char *path_info;
    char *args;          /* QUERY_ARGS, if any */
    struct stat finfo;   /* Set by server core;
                        * st_mode set to zero if no such file */

    char *content_type;
    char *content_encoding;

    /* MIME header environments, in and out. Also,
     * an array containing environment variables to
     * be passed to subprocesses, so people can write
     * modules to add to that environment.
     *
     * The difference between headers_out and
     * err_headers_out is that the latter are printed
     * even on error, and persist across internal
     * redirects (so the headers printed for
     * ErrorDocument handlers will have them).
```

```

*/

table *headers_in;
table *headers_out;
table *err_headers_out;
table *subprocess_env;

/* Info about the request itself... */

int header_only;      /* HEAD request, as opposed to GET */
char *protocol;      /* Protocol, as given to us, or HTTP/0.9 */
char *method;        /* GET, HEAD, POST, etc. */
int method_number;   /* M_GET, M_POST, etc. */

/* Info for logging */

char *the_request;
int bytes_sent;

/* A flag which modules can set, to indicate that
 * the data being returned is volatile, and clients
 * should be told not to cache it.
 */

int no_cache;

/* Various other config info which may change
 * with .htaccess files
 * These are config vectors, with one void*
 * pointer for each module (the thing pointed
 * to being the module's business).
 */

void *per_dir_config; /* Options set in config files, etc. */
void *request_config; /* Notes on *this* request */

};

```

## Where request\_rec structures come from

Most request\_rec structures are built by reading an HTTP request from a client, and filling in the fields. However, there are a few exceptions:

- If the request is to an imagemap, a type map (*i.e.*, a \*.var file), or a CGI script which returned a local `Location:', then the resource which the user requested is going to be ultimately located by some URI other than what the client originally supplied. In this case, the server does an *internal redirect*, constructing a new request\_rec for the new URI, and processing it almost exactly as if the client had requested the new URI directly.
- If some handler signaled an error, and an ErrorDocument is in scope, the same internal redirect machinery comes into play.
- Finally, a handler occasionally needs to investigate `what would happen if' some other request were run. For instance, the directory indexing module needs to know what MIME type would be assigned to a request for each directory entry, in order to figure out what icon to use.

Such handlers can construct a *sub-request*, using the functions ap\_sub\_req\_lookup\_file, ap\_sub\_req\_lookup\_uri, and ap\_sub\_req\_method\_uri; these construct a new request\_rec structure and processes it as you would expect, up to but not including the point of actually sending a response. (These functions skip over the access checks if the sub-request is for a file in the same directory as the original request).

(Server-side includes work by building sub-requests and then actually invoking the response handler for them, via the function ap\_run\_sub\_req).

## **Handling requests, declining, and returning error codes**

As discussed above, each handler, when invoked to handle a particular `request_rec`, has to return an `int` to indicate what happened. That can either be

- OK -- the request was handled successfully. This may or may not terminate the phase.
- DECLINED -- no erroneous condition exists, but the module declines to handle the phase; the server tries to find another.
- an HTTP error code, which aborts handling of the request.

Note that if the error code returned is REDIRECT, then the module should put a `Location` in the request's `headers_out`, to indicate where the client should be redirected to.

## Special considerations for response handlers

Handlers for most phases do their work by simply setting a few fields in the `request_rec` structure (or, in the case of access checkers, simply by returning the correct error code). However, response handlers have to actually send a request back to the client.

They should begin by sending an HTTP response header, using the function `ap_send_http_header`. (You don't have to do anything special to skip sending the header for HTTP/0.9 requests; the function figures out on its own that it shouldn't do anything). If the request is marked `header_only`, that's all they should do; they should return after that, without attempting any further output.

Otherwise, they should produce a request body which responds to the client as appropriate. The primitives for this are `ap_rputc` and `ap_rprintf`, for internally generated output, and `ap_send_fd`, to copy the contents of some `FILE` \* straight to the client.

At this point, you should more or less understand the following piece of code, which is the handler which handles GET requests which have no more specific handler; it also shows how conditional GETs can be handled, if it's desirable to do so in a particular response handler -- `ap_set_last_modified` checks against the `If-modified-since` value supplied by the client, if any, and returns an appropriate code (which will, if nonzero, be `USE_LOCAL_COPY`). No similar considerations apply for `ap_set_content_length`, but it returns an error code for symmetry.

```
int default_handler (request_rec *r)
{
    int errstatus;
    FILE *f;

    if (r->method_number != M_GET) return DECLINED;
    if (r->finfo.st_mode == 0) return NOT_FOUND;

    if ((errstatus = ap_set_content_length (r, r-
>finfo.st_size))
        || (errstatus = ap_set_last_modified (r, r-
>finfo.st_mtime)))
        return errstatus;

    f = fopen (r->filename, "r");

    if (f == NULL) {
        log_reason("file permissions deny server access", r-
>filename, r);
        return FORBIDDEN;
    }

    register_timeout ("send", r);
    ap_send_http_header (r);

    if (!r->header_only) send_fd (f, r);
    ap_pfclose (r->pool, f);
    return OK;
}
```

Finally, if all of this is too much of a challenge, there are a few

ways out of it. First off, as shown above, a response handler which has not yet produced any output can simply return an error code, in which case the server will automatically produce an error response. Secondly, it can punt to some other handler by invoking `ap_internal_redirect`, which is how the internal redirection machinery discussed above is invoked. A response handler which has internally redirected should always return OK.

(Invoking `ap_internal_redirect` from handlers which are *not* response handlers will lead to serious confusion).

## Special considerations for authentication handlers

Stuff that should be discussed here in detail:

- Authentication-phase handlers not invoked unless auth is configured for the directory.
- Common auth configuration stored in the core per-dir configuration; it has accessors `ap_auth_type`, `ap_auth_name`, and `ap_requires`.
- Common routines, to handle the protocol end of things, at least for HTTP basic authentication (`ap_get_basic_auth_pw`, which sets the `connection->user` structure field automatically, and `ap_note_basic_auth_failure`, which arranges for the proper `WWW-Authenticate:` header to be sent back).

## Special considerations for logging handlers

When a request has internally redirected, there is the question of what to log. Apache handles this by bundling the entire chain of redirects into a list of `request_rec` structures which are threaded through the `r->prev` and `r->next` pointers. The `request_rec` which is passed to the logging handlers in such cases is the one which was originally built for the initial request from the client; note

that the `bytes_sent` field will only be correct in the last request in the chain (the one for which a response was actually sent).



---

One of the problems of writing and designing a server-pool server is that of preventing leakage, that is, allocating resources (memory, open files, *etc.*), without subsequently releasing them. The resource pool machinery is designed to make it easy to prevent this from happening, by allowing resource to be allocated in such a way that they are *automatically* released when the server is done with them.

The way this works is as follows: the memory which is allocated, file opened, *etc.*, to deal with a particular request are tied to a *resource pool* which is allocated for the request. The pool is a data structure which itself tracks the resources in question.

When the request has been processed, the pool is *cleared*. At that point, all the memory associated with it is released for reuse, all files associated with it are closed, and any other clean-up functions which are associated with the pool are run. When this is over, we can be confident that all the resource tied to the pool have been released, and that none of them have leaked.

Server restarts, and allocation of memory and resources for per-server configuration, are handled in a similar way. There is a *configuration pool*, which keeps track of resources which were allocated while reading the server configuration files, and handling the commands therein (for instance, the memory that was allocated for per-server module configuration, log files and other files that were opened, and so forth). When the server restarts, and has to reread the configuration files, the configuration pool is cleared, and so the memory and file descriptors which were taken up by reading them the last time are made available for reuse.

It should be noted that use of the pool machinery isn't generally obligatory, except for situations like logging handlers, where you really need to register cleanups to make sure that the log file gets

closed when the server restarts (this is most easily done by using the function `ap_pfopen`, which also arranges for the underlying file descriptor to be closed before any child processes, such as for CGI scripts, are execed), or in case you are using the timeout machinery (which isn't yet even documented here). However, there are two benefits to using it: resources allocated to a pool never leak (even if you allocate a scratch string, and just forget about it); also, for memory allocation, `ap_palloc` is generally faster than `malloc`.

We begin here by describing how memory is allocated to pools, and then discuss how other resources are tracked by the resource pool machinery.

## Allocation of memory in pools

Memory is allocated to pools by calling the function `ap_palloc`, which takes two arguments, one being a pointer to a resource pool structure, and the other being the amount of memory to allocate (in chars). Within handlers for handling requests, the most common way of getting a resource pool structure is by looking at the `pool` slot of the relevant `request_rec`; hence the repeated appearance of the following idiom in module code:

```
int my_handler(request_rec *r)
{
    struct my_structure *foo;
    ...

    foo = (foo *)ap_palloc (r->pool, sizeof(my_structure));
}
```

Note that *there is no `ap_pfree`* -- `ap_palloc`d memory is freed only when the associated resource pool is cleared. This means that `ap_palloc` does not have to do as much accounting as `malloc()`; all it does in the typical case is to round up the size,

bump a pointer, and do a range check.

(It also raises the possibility that heavy use of `ap_palloc` could cause a server process to grow excessively large. There are two ways to deal with this, which are dealt with below; briefly, you can use `malloc`, and try to be sure that all of the memory gets explicitly freed, or you can allocate a sub-pool of the main pool, allocate your memory in the sub-pool, and clear it out periodically. The latter technique is discussed in the section on sub-pools below, and is used in the directory-indexing code, in order to avoid excessive storage allocation when listing directories with thousands of files).

## Allocating initialized memory

There are functions which allocate initialized memory, and are frequently useful. The function `ap_pcalloc` has the same interface as `ap_palloc`, but clears out the memory it allocates before it returns it. The function `ap_pstrdup` takes a resource pool and a `char *` as arguments, and allocates memory for a copy of the string the pointer points to, returning a pointer to the copy. Finally `ap_pstrcat` is a `varargs`-style function, which takes a pointer to a resource pool, and at least two `char *` arguments, the last of which must be `NULL`. It allocates enough memory to fit copies of each of the strings, as a unit; for instance:

```
ap_pstrcat (r->pool, "foo", "/", "bar", NULL);
```

returns a pointer to 8 bytes worth of memory, initialized to "foo/bar".

## Commonly-used pools in the Apache Web server

A pool is really defined by its lifetime more than anything else.

There are some static pools in `http_main` which are passed to various non-`http_main` functions as arguments at opportune times. Here they are:

### **permanent\_pool**

never passed to anything else, this is the ancestor of all pools

### **pconf**

- subpool of `permanent_pool`
- created at the beginning of a config "cycle"; exists until the server is terminated or restarts; passed to all config-time routines, either via `cmd->pool`, or as the "pool \*p" argument on those which don't take pools
- passed to the module `init()` functions

### **ptemp**

- sorry I lie, this pool isn't called this currently in 1.3, I renamed it this in my pthreads development. I'm referring to the use of `ptrans` in the parent... contrast this with the later definition of `ptrans` in the child.
- subpool of `permanent_pool`
- created at the beginning of a config "cycle"; exists until the end of config parsing; passed to config-time routines *via* `cmd->temp_pool`. Somewhat of a "bastard child" because it isn't available everywhere. Used for temporary scratch space which may be needed by some config routines but which is deleted at the end of config.

### **pchild**

- subpool of `permanent_pool`
- created when a child is spawned (or a thread is created); lives until that child (thread) is destroyed
- passed to the module `child_init` functions
- destruction happens right after the `child_exit` functions are called... (which may explain why I think `child_exit` is

redundant and unneeded)

### **ptrans**

- should be a subpool of pchild, but currently is a subpool of permanent\_pool, see above
- cleared by the child before going into the accept() loop to receive a connection
- used as connection->pool

### **r->pool**

- for the main request this is a subpool of connection->pool; for subrequests it is a subpool of the parent request's pool.
- exists until the end of the request (*i.e.*, ap\_destroy\_sub\_req, or in child\_main after process\_request has finished)
- note that r itself is allocated from r->pool; *i.e.*, r->pool is first created and then r is the first thing palloc()d from it

For almost everything folks do, r->pool is the pool to use. But you can see how other lifetimes, such as pchild, are useful to some modules... such as modules that need to open a database connection once per child, and wish to clean it up when the child dies.

You can also see how some bugs have manifested themselves, such as setting connection->user to a value from r->pool -- in this case connection exists for the lifetime of ptrans, which is longer than r->pool (especially if r->pool is a subrequest!). So the correct thing to do is to allocate from connection->pool.

And there was another interesting bug in [mod\\_include](#) / [mod\\_cgi](#). You'll see in those that they do this test to decide if they should use r->pool or r->main->pool. In this case the resource that they are registering for cleanup is a child process. If

it were registered in `r->pool`, then the code would `wait()` for the child when the subrequest finishes. With `mod_include` this could be any old `#include`, and the delay can be up to 3 seconds... and happened quite frequently. Instead the subprocess is registered in `r->main->pool` which causes it to be cleaned up when the entire request is done -- *i.e.*, after the output has been sent to the client and logging has happened.

## Tracking open files, etc.

As indicated above, resource pools are also used to track other sorts of resources besides memory. The most common are open files. The routine which is typically used for this is `ap_pfdopen`, which takes a resource pool and two strings as arguments; the strings are the same as the typical arguments to `fopen`, *e.g.*,

```
...
FILE *f = ap_pfdopen (r->pool, r->filename, "r");
if (f == NULL) { ... } else { ... }
```

There is also a `ap_popenf` routine, which parallels the lower-level open system call. Both of these routines arrange for the file to be closed when the resource pool in question is cleared.

Unlike the case for memory, there *are* functions to close files allocated with `ap_pfdopen`, and `ap_popenf`, namely `ap_pfdclose` and `ap_pclosef`. (This is because, on many systems, the number of files which a single process can have open is quite limited). It is important to use these functions to close files allocated with `ap_pfdopen` and `ap_popenf`, since to do otherwise could cause fatal errors on systems such as Linux, which react badly if the same `FILE*` is closed more than once.

(Using the `close` functions is not mandatory, since the file will

eventually be closed regardless, but you should consider it in cases where your module is opening, or could open, a lot of files).

## **Other sorts of resources -- cleanup functions**

More text goes here. Describe the cleanup primitives in terms of which the file stuff is implemented; also, `spawn_process`.

Pool cleanups live until `clear_pool()` is called:

`clear_pool(a)` recursively calls `destroy_pool()` on all subpools of `a`; then calls all the cleanups for `a`; then releases all the memory for `a`. `destroy_pool(a)` calls `clear_pool(a)` and then releases the pool structure itself. *i.e.*, `clear_pool(a)` doesn't delete `a`, it just frees up all the resources and you can start using it again immediately.

## **Fine control -- creating and dealing with sub-pools, with a note on sub-requests**

On rare occasions, too-free use of `ap_palloc()` and the associated primitives may result in undesirably profligate resource allocation. You can deal with such a case by creating a *sub-pool*, allocating within the sub-pool rather than the main pool, and clearing or destroying the sub-pool, which releases the resources which were associated with it. (This really *is* a rare situation; the only case in which it comes up in the standard module set is in case of listing directories, and then only with *very* large directories. Unnecessary use of the primitives discussed here can hair up your code quite a bit, with very little gain).

The primitive for creating a sub-pool is `ap_make_sub_pool`, which takes another pool (the parent pool) as an argument. When the main pool is cleared, the sub-pool will be destroyed. The sub-pool may also be cleared or destroyed at any time, by calling the functions `ap_clear_pool` and `ap_destroy_pool`, respectively.

(The difference is that `ap_clear_pool` frees resources associated with the pool, while `ap_destroy_pool` also deallocates the pool itself. In the former case, you can allocate new resources within the pool, and clear it again, and so forth; in the latter case, it is simply gone).

One final note -- sub-requests have their own resource pools, which are sub-pools of the resource pool for the main request. The polite way to reclaim the resources associated with a sub request which you have allocated (using the `ap_sub_req_...` functions) is `ap_destroy_sub_req`, which frees the resource pool. Before calling this function, be sure to copy anything that you care about which might be allocated in the sub-request's resource pool into someplace a little less volatile (for instance, the filename in its `request_rec` structure).

(Again, under most circumstances, you shouldn't feel obliged to call this function; only 2K of memory or so are allocated for a typical sub request, and it will be freed anyway when the main request pool is cleared. It is only when you are allocating many, many sub-requests for a single main request that you should seriously consider the `ap_destroy_...` functions).



## Configuration, Commands and the like

One of the design goals for this server was to maintain external compatibility with the NCSA 1.3 server --- that is, to read the same configuration files, to process all the directives therein correctly, and in general to be a drop-in replacement for NCSA. On the other hand, another design goal was to move as much of the server's functionality into modules which have as little as possible to do with the monolithic server core. The only way to reconcile these goals is to move the handling of most commands from the central server into the modules.

However, just giving the modules command tables is not enough to divorce them completely from the server core. The server has to remember the commands in order to act on them later. That involves maintaining data which is private to the modules, and which can be either per-server, or per-directory. Most things are per-directory, including in particular access control and authorization information, but also information on how to determine file types from suffixes, which can be modified by [AddType](#) and [DefaultType](#) directives, and so forth. In general, the governing philosophy is that anything which *can* be made configurable by directory should be; per-server information is generally used in the standard set of modules for information like [Aliases](#) and [Redirects](#) which come into play before the request is tied to a particular place in the underlying file system.

Another requirement for emulating the NCSA server is being able to handle the per-directory configuration files, generally called `.htaccess` files, though even in the NCSA server they can contain directives which have nothing at all to do with access control. Accordingly, after URI -> filename translation, but before performing any other phase, the server walks down the directory hierarchy of the underlying filesystem, following the translated pathname, to read any `.htaccess` files which might be present.

The information which is read in then has to be *merged* with the applicable information from the server's own config files (either from the `<Directory>` sections in `access.conf`, or from defaults in `srml.conf`, which actually behaves for most purposes almost exactly like `<Directory />`).

Finally, after having served a request which involved reading `.htaccess` files, we need to discard the storage allocated for handling them. That is solved the same way it is solved wherever else similar problems come up, by tying those structures to the per-transaction resource pool.

## Per-directory configuration structures

Let's look out how all of this plays out in `mod_mime.c`, which defines the file typing handler which emulates the NCSA server's behavior of determining file types from suffixes. What we'll be looking at, here, is the code which implements the `AddType` and `AddEncoding` commands. These commands can appear in `.htaccess` files, so they must be handled in the module's private per-directory data, which in fact, consists of two separate tables for MIME types and encoding information, and is declared as follows:

```
typedef struct {
    table *forced_types;      /* Additional AddTyped stuff */
    table *encoding_types;   /* Added with AddEncoding... */
} mime_dir_config;
```

When the server is reading a configuration file, or `<Directory>` section, which includes one of the MIME module's commands, it needs to create a `mime_dir_config` structure, so those commands have something to act on. It does this by invoking the function it finds in the module's 'create per-dir config slot', with two arguments: the name of the directory to which this configuration

information applies (or NULL for `srm.conf`), and a pointer to a resource pool in which the allocation should happen.

(If we are reading a `.htaccess` file, that resource pool is the per-request resource pool for the request; otherwise it is a resource pool which is used for configuration data, and cleared on restarts. Either way, it is important for the structure being created to vanish when the pool is cleared, by registering a cleanup on the pool if necessary).

For the MIME module, the per-dir config creation function just `ap_pallocs` the structure above, and creates a couple of tables to fill it. That looks like this:

```
void *create_mime_dir_config (pool *p, char *dummy)
{
    mime_dir_config *new =
        (mime_dir_config *) ap_palloc (p,
        sizeof(mime_dir_config));

    new->forced_types = ap_make_table (p, 4);
    new->encoding_types = ap_make_table (p, 4);

    return new;
}
```

Now, suppose we've just read in a `.htaccess` file. We already have the per-directory configuration structure for the next directory up in the hierarchy. If the `.htaccess` file we just read in didn't have any [AddType](#) or [AddEncoding](#) commands, its per-directory config structure for the MIME module is still valid, and we can just use it. Otherwise, we need to merge the two structures somehow.

To do that, the server invokes the module's per-directory config merge function, if one is present. That function takes three arguments: the two structures being merged, and a resource pool in which to allocate the result. For the MIME module, all that needs

to be done is overlay the tables from the new per-directory config structure with those from the parent:

```
void *merge_mime_dir_configs (pool *p, void *parent_dirv, void
*subdirv)
{
    mime_dir_config *parent_dir = (mime_dir_config
*)parent_dirv;
    mime_dir_config *subdir = (mime_dir_config *)subdirv;
    mime_dir_config *new =
        (mime_dir_config *)ap_palloc (p, sizeof(mime_dir_config));

    new->forced_types = ap_overlay_tables (p, subdir-
>forced_types,
        parent_dir->forced_types);
    new->encoding_types = ap_overlay_tables (p, subdir-
>encoding_types,
        parent_dir->encoding_types);

    return new;
}
```

As a note -- if there is no per-directory merge function present, the server will just use the subdirectory's configuration info, and ignore the parent's. For some modules, that works just fine (e.g., for the includes module, whose per-directory configuration information consists solely of the state of the XBITHACK), and for those modules, you can just not declare one, and leave the corresponding structure slot in the module itself NULL.

## Command handling

Now that we have these structures, we need to be able to figure out how to fill them. That involves processing the actual [AddType](#) and [AddEncoding](#) commands. To find commands, the server looks in the module's command table. That table contains information on how many arguments the commands take, and in what formats, where it is permitted, and so forth. That information is sufficient to allow the server to invoke most command-handling functions with pre-parsed arguments. Without further ado, let's

look at the [AddType](#) command handler, which looks like this (the [AddEncoding](#) command looks basically the same, and won't be shown here):

```
char *add_type(cmd_parms *cmd, mime_dir_config *m, char *ct,
char *ext)
{
    if (*ext == '.') ++ext;
    ap_table_set (m->forced_types, ext, ct);
    return NULL;
}
```

This command handler is unusually simple. As you can see, it takes four arguments, two of which are pre-parsed arguments, the third being the per-directory configuration structure for the module in question, and the fourth being a pointer to a `cmd_parms` structure. That structure contains a bunch of arguments which are frequently of use to some, but not all, commands, including a resource pool (from which memory can be allocated, and to which cleanups should be tied), and the (virtual) server being configured, from which the module's per-server configuration data can be obtained if required.

Another way in which this particular command handler is unusually simple is that there are no error conditions which it can encounter. If there were, it could return an error message instead of `NULL`; this causes an error to be printed out on the server's `stderr`, followed by a quick exit, if it is in the main config files; for a `.htaccess` file, the syntax error is logged in the server error log (along with an indication of where it came from), and the request is bounced with a server error response (HTTP error status, code 500).

The MIME module's command table has entries for these commands, which look like this:

```
command_rec mime_cmds[] = {
    { "AddType", add_type, NULL, OR_FILEINFO, TAKE2,
      "a mime type followed by a file extension" },
    { "AddEncoding", add_encoding, NULL, OR_FILEINFO, TAKE2,
      "an encoding (e.g., gzip), followed by a file extension"
    },
    { NULL }
};
```

The entries in these tables are:

- The name of the command
- The function which handles it
- a (`void *`) pointer, which is passed in the `cmd_parms` structure to the command handler --- this is useful in case many similar commands are handled by the same function.
- A bit mask indicating where the command may appear. There are mask bits corresponding to each `AllowOverride` option, and an additional mask bit, `RSRC_CONF`, indicating that the command may appear in the server's own config files, but *not* in any `.htaccess` file.
- A flag indicating how many arguments the command handler wants pre-parsed, and how they should be passed in. `TAKE2` indicates two pre-parsed arguments. Other options are `TAKE1`, which indicates one pre-parsed argument, `FLAG`, which indicates that the argument should be `On` or `Off`, and is passed in as a boolean flag, `RAW_ARGS`, which causes the server to give the command the raw, unparsed arguments (everything but the command name itself). There is also `ITERATE`, which means that the handler looks the same as `TAKE1`, but that if multiple arguments are present, it should be called multiple times, and finally `ITERATE2`, which indicates that the command handler looks like a `TAKE2`, but if more arguments are present, then it should be called multiple times, holding the first argument constant.

- Finally, we have a string which describes the arguments that should be present. If the arguments in the actual config file are not as required, this string will be used to help give a more specific error message. (You can safely leave this NULL).

Finally, having set this all up, we have to use it. This is ultimately done in the module's handlers, specifically for its file-typing handler, which looks more or less like this; note that the per-directory configuration structure is extracted from the request\_rec's per-directory configuration vector by using the ap\_get\_module\_config function.

```
int find_ct(request_rec *r)
{
    int i;
    char *fn = ap_pstrdup (r->pool, r->filename);
    mime_dir_config *conf = (mime_dir_config *)
        ap_get_module_config(r->per_dir_config, &mime_module);
    char *type;

    if (S_ISDIR(r->finfo.st_mode)) {
        r->content_type = DIR_MAGIC_TYPE;
        return OK;
    }

    if((i=ap_rind(fn, '.')) < 0) return DECLINED;
    ++i;

    if ((type = ap_table_get (conf->encoding_types, &fn[i])))
    {
        r->content_encoding = type;

        /* go back to previous extension to try to use it as a
        type */
        fn[i-1] = '\0';
        if((i=ap_rind(fn, '.')) < 0) return OK;
        ++i;
    }

    if ((type = ap_table_get (conf->forced_types, &fn[i])))
    {
        r->content_type = type;
    }
}
```

```
    return OK;
}
```

## Side notes -- per-server configuration, virtual servers, etc.

The basic ideas behind per-server module configuration are basically the same as those for per-directory configuration; there is a creation function and a merge function, the latter being invoked where a virtual server has partially overridden the base server configuration, and a combined structure must be computed. (As with per-directory configuration, the default if no merge function is specified, and a module is configured in some virtual server, is that the base configuration is simply ignored).

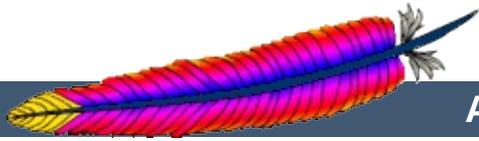
The only substantial difference is that when a command needs to configure the per-server private module data, it needs to go to the `cmd_parms` data to get at it. Here's an example, from the `alias` module, which also indicates how a syntax error can be returned (note that the per-directory configuration argument to the command handler is declared as a dummy, since the module doesn't actually have per-directory config data):

```
char *add_redirect(cmd_parms *cmd, void *dummy, char *f, char
*url)
{
    server_rec *s = cmd->server;
    alias_server_conf *conf = (alias_server_conf *)
        ap_get_module_config(s->module_config,&alias_module);
    alias_entry *new = ap_push_array (conf->redirects);

    if (!ap_is_url (url)) return "Redirect to non-URL";

    new->fake = f; new->real = url;
    return NULL;
}
```

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

## Debugging Memory Allocation in APR

The allocation mechanism's within APR have a number of debugging modes that can be used to assist in finding memory problems. This document describes the modes available and gives instructions on activating them.



### Allocation Debugging - `ALLOC_DEBUG`

Debugging support: Define this to enable code which helps detect re-use of `free()`d memory and other such nonsense.

The theory is simple. The `FILL_BYTE` (`0xa5`) is written over all `malloc`'d memory as we receive it, and is written over everything that we free up during a `clear_pool`. We check that blocks on the free list always have the `FILL_BYTE` in them, and we check during `palloc()` that the bytes still have `FILL_BYTE` in them. If you ever see garbage URLs or whatnot containing lots of `0xa5`s then you know something used data that's been freed or uninitialized.

### Malloc Support - `ALLOC_USE_MALLOC`

If defined all allocations will be done with `malloc()` and `free()`d appropriately at the end.

This is intended to be used with something like Electric Fence or Purify to help detect memory problems. Note that if you're using `efence` then you should also add in `ALLOC_DEBUG`. But don't add in `ALLOC_DEBUG` if you're using Purify because `ALLOC_DEBUG` would hide all the uninitialized read errors that Purify can diagnose.

### Pool Debugging - `POOL_DEBUG`

This is intended to detect cases where the wrong pool is used when assigning data to an object in another pool.

In particular, it causes the `table_{set, add, merge}` routines to check that their arguments are safe for the `apr_table_t` they're being placed in. It currently only works with the unix multiprocess model, but could be extended to others.

## Table Debugging - MAKE\_TABLE\_PROFILE

Provide diagnostic information about `make_table()` calls which are possibly too small.

This requires a recent gcc which supports `__builtin_return_address()`. The `error_log` output will be a message such as:

```
table_push: apr_table_t created by 0x804d874 hit limit of 10
```

Use `l *0x804d874` to find the source that corresponds to. It indicates that a `apr_table_t` allocated by a call at that address has possibly too small an initial `apr_table_t` size guess.

## Allocation Statistics - ALLOC\_STATS

Provide some statistics on the cost of allocations.

This requires a bit of an understanding of how `alloc.c` works.



## ALLOCATION COMBINATIONS

Not all the options outlined above can be activated at the same time. the following table gives more information.

	<b>ALLOC DEBUG</b>	<b>ALLOC USE MALLOC</b>	<b>POOL DEBUG</b>	<b>MAKE TABLE PROFILE</b>	<b>ALLOC STATS</b>
<b>ALLOC DEBUG</b>	-	No	Yes	Yes	Yes
<b>ALLOC USE MALLOC</b>	No	-	No	No	No
<b>POOL DEBUG</b>	Yes	No	-	Yes	Yes
<b>MAKE TABLE PROFILE</b>	Yes	No	Yes	-	Yes
<b>ALLOC STATS</b>	Yes	No	Yes	Yes	-

Additionally the debugging options are not suitable for multi-threaded versions of the server. When trying to debug with these options the server should be started in single process mode.



## Remaining Debugging Options

The various options for debugging memory are now enabled in the `apr_general.h` header file in APR. The various options are enabled by uncommenting the define for the option you wish to use. The section of the code currently looks like this (*contained in `src/lib/apr/include/apr_pools.h`*)

```
/*
#define ALLOC_DEBUG
#define POOL_DEBUG
#define ALLOC_USE_MALLOC
#define MAKE_TABLE_PROFILE
#define ALLOC_STATS
*/

typedef struct ap_pool_t {
    union block_hdr *first;
    union block_hdr *last;
    struct cleanup *cleanups;
    struct process_chain *subprocesses;
    struct ap_pool_t *sub_pools;
    struct ap_pool_t *sub_next;
    struct ap_pool_t *sub_prev;
    struct ap_pool_t *parent;
    char *free_first_avail;
#ifdef ALLOC_USE_MALLOC
    void *allocation_list;
#endif
#ifdef POOL_DEBUG
    struct ap_pool_t *joined;
#endif
    int (*apr_abort)(int retcode);
    struct datastruct *prog_data;
} ap_pool_t;
```

To enable allocation debugging simply move the `#define ALLOC_DEBUG` above the start of the comments block and rebuild the server.

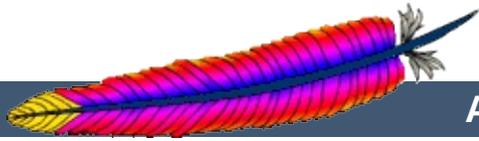
### Note

In order to use the various options the server **must** be rebuilt after editing the header file.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

## Documenting Apache 2.0

Apache 2.0 uses [Doxygen](#) to document the APIs and global variables in the code. This will explain the basics of how to document using Doxygen.



To start a documentation block, use `/**`

To end a documentation block, use `*/`

In the middle of the block, there are multiple tags we can use:

```
Description of this functions purpose
@param parameter_name description
@return description
@deffunc signature of the function
```

The `deffunc` is not always necessary. Doxygen does not have a full parser in it, so any prototype that use a macro in the return type declaration is too complex for scandoc. Those functions require a `deffunc`. An example (using `&gt;` rather than `>`):

```
/**
 * return the final element of the pathname
 * @param pathname The path to get the final element of
 * @return the final element of the path
 * @tip Examples:
 * <pre>
 * "/foo/bar/gum" -&gt; "gum"
 * "/foo/bar/gum/" -&gt; ""
 * "gum" -&gt; "gum"
 * "wi\\n32\\stuff" -&gt; "stuff"
 * </pre>
 * @deffunc const char * ap_filename_of_pathname(const char
 *pathname)
 */
```

At the top of the header file, always include:

```
/**
 * @package Name of library header
 */
```

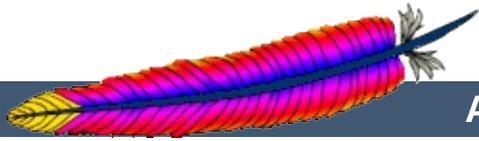
Doxygen uses a new HTML file for each package. The HTML files are named `{Name_of_library_header}.html`, so try to be concise with your names.

For a further discussion of the possibilities please refer to [the Doxygen site](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

# Apache 2.0 Hook Functions

## Warning

This document is still in development and may be partially out of date.

In general, a hook function is one that Apache will call at some point during the processing of a request. Modules can provide functions that are called, and specify when they get called in comparison to other modules.



## Creating a hook function

In order to create a new hook, four things need to be done:

### Declare the hook function

Use the `AP_DECLARE_HOOK` macro, which needs to be given the return type of the hook function, the name of the hook, and the arguments. For example, if the hook returns an `int` and takes a `request_rec *` and an `int` and is called `do_something`, then declare it like this:

```
AP_DECLARE_HOOK(int, do_something, (request_rec *r, int n))
```

This should go in a header which modules will include if they want to use the hook.

### Create the hook structure

Each source file that exports a hook has a private structure which is used to record the module functions that use the hook. This is declared as follows:

```
APR_HOOK_STRUCT(  
    APR_HOOK_LINK(do_something)  
    ...  
)
```

### Implement the hook caller

The source file that exports the hook has to implement a function that will call the hook. There are currently three possible ways to do this. In all cases, the calling function is called `ap_run_hookname()`.

### Void hooks

If the return value of a hook is `void`, then all the hooks are called,

and the caller is implemented like this:

```
AP_IMPLEMENT_HOOK_VOID(do_something, (request_rec *r, int n),
(r, n))
```

The second and third arguments are the dummy argument declaration and the dummy arguments as they will be used when calling the hook. In other words, this macro expands to something like this:

```
void ap_run_do_something(request_rec *r, int n)
{
    ...
    do_something(r, n);
}
```

### Hooks that return a value

If the hook returns a value, then it can either be run until the first hook that does something interesting, like so:

```
AP_IMPLEMENT_HOOK_RUN_FIRST(int, do_something, (request_rec *r,
int n), (r, n), DECLINED)
```

The first hook that does *not* return DECLINED stops the loop and its return value is returned from the hook caller. Note that DECLINED is the tradition Apache hook return meaning "I didn't do anything", but it can be whatever suits you.

Alternatively, all hooks can be run until an error occurs. This boils down to permitting *two* return values, one of which means "I did something, and it was OK" and the other meaning "I did nothing". The first function that returns a value other than one of those two stops the loop, and its return is the return value. Declare these like so:

```
AP_IMPLEMENT_HOOK_RUN_ALL(int, do_something, (request_rec *r,
```

```
int n), (r, n), OK, DECLINED)
```

Again, OK and DECLINED are the traditional values. You can use what you want.

## Call the hook callers

At appropriate moments in the code, call the hook caller, like so:

```
int n, ret;  
request_rec *r;  
  
ret=ap_run_do_something(r, n);
```



A module that wants a hook to be called needs to do two things.

## Implement the hook function

Include the appropriate header, and define a static function of the correct type:

```
static int my_something_doer(request_rec *r, int n)
{
    ...
    return OK;
}
```

## Add a hook registering function

During initialisation, Apache will call each modules hook registering function, which is included in the module structure:

```
static void my_register_hooks()
{
    ap_hook_do_something(my_something_doer, NULL, NULL,
        HOOK_MIDDLE);
}

module MODULE_VAR_EXPORT my_module =
{
    ...
    my_register_hooks /* register hooks */
};
```

## Controlling hook calling order

In the example above, we didn't use the three arguments in the hook registration function that control calling order. There are two mechanisms for doing this. The first, rather crude, method, allows us to specify roughly where the hook is run relative to other modules. The final argument control this. There are three possible values: HOOK\_FIRST, HOOK\_MIDDLE and HOOK\_LAST.

All modules using any particular value may be run in any order relative to each other, but, of course, all modules using HOOK\_FIRST will be run before HOOK\_MIDDLE which are before HOOK\_LAST. Modules that don't care when they are run should use HOOK\_MIDDLE. *(I spaced these out so people could do stuff like HOOK\_FIRST-2 to get in slightly earlier, but is this wise? - Ben)*

Note that there are two more values, HOOK\_REALLY\_FIRST and HOOK\_REALLY\_LAST. These should only be used by the hook exporter.

The other method allows finer control. When a module knows that it must be run before (or after) some other modules, it can specify them by name. The second (third) argument is a NULL-terminated array of strings consisting of the names of modules that must be run before (after) the current module. For example, suppose we want "mod\_xyz.c" and "mod\_abc.c" to run before we do, then we'd hook as follows:

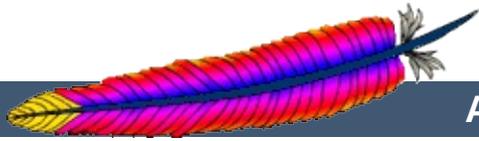
```
static void register_hooks()
{
    static const char * const aszPre[] = { "mod_xyz.c",
        "mod_abc.c", NULL };

    ap_hook_do_something(my_something_doer, aszPre, NULL,
        HOOK_MIDDLE);
}
```

Note that the sort used to achieve this is stable, so ordering set by HOOK\_ORDER is preserved, as far as is possible.

*Ben Laurie, 15th August 1999*

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

## Converting Modules from Apache 1.3 to Apache 2.0

This is a first attempt at writing the lessons I learned when trying to convert the `mod_mmap_static` module to Apache 2.0. It's by no means definitive and probably won't even be correct in some ways, but it's a start.



## Cleanup Routines

These now need to be of type `apr_status_t` and return a value of that type. Normally the return value will be `APR_SUCCESS` unless there is some need to signal an error in the cleanup. Be aware that even though you signal an error not all code yet checks and acts upon the error.

## Initialisation Routines

These should now be renamed to better signify where they sit in the overall process. So the name gets a small change from `mmap_init` to `mmap_post_config`. The arguments passed have undergone a radical change and now look like

- `apr_pool_t *p`
- `apr_pool_t *plog`
- `apr_pool_t *ptemp`
- `server_rec *s`

## Data Types

A lot of the data types have been moved into the [APR](#). This means that some have had a name change, such as the one shown above. The following is a brief list of some of the changes that you are likely to have to make.

- `pool` becomes `apr_pool_t`
- `table` becomes `apr_table_t`



## Register Hooks

The new architecture uses a series of hooks to provide for calling your functions. These you'll need to add to your module by way of a new function, `static void register_hooks(void)`. The function is really reasonably straightforward once you understand what needs to be done. Each function that needs calling at some stage in the processing of a request needs to be registered, handlers do not. There are a number of phases where functions can be added, and for each you can specify with a high degree of control the relative order that the function will be called in.

This is the code that was added to `mod_mmap_static`:

```
static void register_hooks(void)
{
    static const char * const aszPre[]={ "http_core.c",NULL };
    ap_hook_post_config(mmap_post_config,NULL,NULL,HOOK_MIDDLE);
    ap_hook_translate_name(mmap_static_xlat,aszPre,NULL,HOOK_LAST);
};
```

This registers 2 functions that need to be called, one in the `post_config` stage (virtually every module will need this one) and one for the `translate_name` phase. note that while there are different function names the format of each is identical. So what is the format?

```
ap_hook_phase_name(function_name, predecessors, successors,
position);
```

There are 3 hook positions defined...

- HOOK\_FIRST
- HOOK\_MIDDLE
- HOOK\_LAST

To define the position you use the position and then modify it with the predecessors and successors. Each of the modifiers can be a list of functions that should be called, either before the function is run (predecessors) or after the function has run (successors).

In the `mod_mmap_static` case I didn't care about the `post_config` stage, but the `mmap_static_xlat` **must** be called after the core module had done its name translation, hence the use of the `aszPre` to define a modifier to the position `HOOK_LAST`.

## Module Definition

There are now a lot fewer stages to worry about when creating your module definition. The old definition looked like

```
module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD_MODULE_STUFF,
    /* initializer */
    /* dir config creator */
    /* dir merger --- default is to override */
    /* server config */
    /* merge server config */
    /* command handlers */
    /* handlers */
    /* filename translation */
    /* check_user_id */
    /* check auth */
    /* check access */
    /* type_checker */
    /* fixups */
    /* logger */
    /* header parser */
    /* child_init */
    /* child_exit */
    /* post read-request */
};
```

The new structure is a great deal simpler...

```

module MODULE_VAR_EXPORT module_name_module =
{
    STANDARD20_MODULE_STUFF,
    /* create per-directory config structures */
    /* merge per-directory config structures */
    /* create per-server config structures */
    /* merge per-server config structures */
    /* command handlers */
    /* handlers */
    /* register hooks */
};

```

Some of these read directly across, some don't. I'll try to summarise what should be done below.

The stages that read directly across :

```

/* dir config creator */
    /* create per-directory config structures */
/* server config */
    /* create per-server config structures */
/* dir merger */
    /* merge per-directory config structures */
/* merge server config */
    /* merge per-server config structures */
/* command table */
    /* command apr_table_t */
/* handlers */
    /* handlers */

```

The remainder of the old functions should be registered as hooks. There are the following hook stages defined so far...

### **ap\_hook\_post\_config**

this is where the old `_init` routines get registered

### **ap\_hook\_http\_method**

retrieve the http method from a request. (legacy)

**ap\_hook\_open\_logs**

open any specified logs

**ap\_hook\_auth\_checker**

check if the resource requires authorization

**ap\_hook\_access\_checker**

check for module-specific restrictions

**ap\_hook\_check\_user\_id**

check the user-id and password

**ap\_hook\_default\_port**

retrieve the default port for the server

**ap\_hook\_pre\_connection**

do any setup required just before processing, but after accepting

**ap\_hook\_process\_connection**

run the correct protocol

**ap\_hook\_child\_init**

call as soon as the child is started

**ap\_hook\_create\_request**

??

**ap\_hook\_fixups**

last chance to modify things before generating content

**ap\_hook\_handler**

generate the content

**ap\_hook\_header\_parser**

lets modules look at the headers, not used by most modules, because they use `post_read_request` for this

**ap\_hook\_insert\_filter**

to insert filters into the filter chain

**ap\_hook\_log\_transaction**

log information about the request

**ap\_hook\_optional\_fn\_retrieve**

retrieve any functions registered as optional

**ap\_hook\_post\_read\_request**

called after reading the request, before any other phase

**ap\_hook\_quick\_handler**

called before any request processing, used by cache modules.

**ap\_hook\_translate\_name**

translate the URI into a filename

**ap\_hook\_type\_checker**

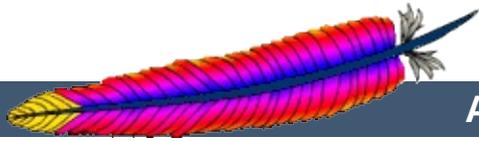
determine and/or set the doc type

---

Copyright 2013 The Apache Software Foundation.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

## Request Processing in Apache 2.0

### Warning

Warning - this is a first (fast) draft that needs further revision!

Several changes in Apache 2.0 affect the internal request processing mechanics. Module authors need to be aware of these changes so they may take advantage of the optimizations and security enhancements.

The first major change is to the subrequest and redirect mechanisms. There were a number of different code paths in Apache 1.3 to attempt to optimize subrequest or redirect behavior. As patches were introduced to 2.0, these optimizations (and the server behavior) were quickly broken due to this duplication of code. All duplicate code has been folded back into `ap_process_request_internal()` to prevent the code from falling out of sync again.

This means that much of the existing code was 'unoptimized'. It is the Apache HTTP Project's first goal to create a robust and correct implementation of the HTTP server RFC. Additional goals include security, scalability and optimization. New methods were sought to optimize the server (beyond the performance of Apache 1.3) without introducing fragile or insecure code.



## The Request Processing Cycle

All requests pass through `ap_process_request_internal()` in `request.c`, including subrequests and redirects. If a module doesn't pass generated requests through this code, the author is cautioned that the module may be broken by future changes to request processing.

To streamline requests, the module author can take advantage of the hooks offered to drop out of the request cycle early, or to bypass core Apache hooks which are irrelevant (and costly in terms of CPU.)



### Unescapes the URL

The request's `parsed_uri` path is unescaped, once and only once, at the beginning of internal request processing.

This step is bypassed if the `proxyreq` flag is set, or the `parsed_uri.path` element is unset. The module has no further control of this one-time unescape operation, either failing to unescape or multiply unescaping the URL leads to security repercussions.

### Strips Parent and This Elements from the URI

All `/../` and `/./` elements are removed by `ap_getparents()`. This helps to ensure the path is (nearly) absolute before the request processing continues.

This step cannot be bypassed.

### Initial URI Location Walk

Every request is subject to an `ap_location_walk()` call. This ensures that `<Location>` sections are consistently enforced for all requests. If the request is an internal redirect or a sub-request, it may borrow some or all of the processing from the previous or parent request's `ap_location_walk`, so this step is generally very efficient after processing the main request.

### `translate_name`

Modules can determine the file name, or alter the given URI in this step. For example, `mod_vhost_alias` will translate the URI's path into the configured virtual host, `mod_alias` will translate the path to an alias path, and if the request falls back on the core, the

[DocumentRoot](#) is prepended to the request resource.

If all modules DECLINE this phase, an error 500 is returned to the browser, and a "couldn't translate name" error is logged automatically.

### **Hook: map\_to\_storage**

After the file or correct URI was determined, the appropriate per-dir configurations are merged together. For example, [mod\\_proxy](#) compares and merges the appropriate [<Proxy>](#) sections. If the URI is nothing more than a local (non-proxy) TRACE request, the core handles the request and returns DONE. If no module answers this hook with OK or DONE, the core will run the request filename against the [<Directory>](#) and [<Files>](#) sections. If the request 'filename' isn't an absolute, legal filename, a note is set for later termination.

### **URI Location Walk**

Every request is hardened by a second `ap_location_walk()` call. This reassures that a translated request is still subjected to the configured [<Location>](#) sections. The request again borrows some or all of the processing from its previous `location_walk` above, so this step is almost always very efficient unless the translated URI mapped to a substantially different path or Virtual Host.

### **Hook: header\_parser**

The main request then parses the client's headers. This prepares the remaining request processing steps to better serve the client's request.



## Needs Documentation. Code is:

```
switch (ap_satisfies(r)) {
case SATISFY_ALL:
case SATISFY_NOSPEC:
    if ((access_status = ap_run_access_checker(r)) != 0) {
        return decl_die(access_status, "check access", r);
    }

    if (ap_some_auth_required(r)) {
        if (((access_status = ap_run_check_user_id(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                ? "check user.  No user file?"
                : "perform authentication. AuthType not
                    r");
        }

        if (((access_status = ap_run_auth_checker(r)) != 0)
            || !ap_auth_type(r)) {
            return decl_die(access_status, ap_auth_type(r)
                ? "check access.  No groups file?"
                : "perform authentication. AuthType not
                    r");
        }
    }
    break;

case SATISFY_ANY:
    if (((access_status = ap_run_access_checker(r)) != 0)) {
        if (!ap_some_auth_required(r)) {
            return decl_die(access_status, "check access", r);
        }
    }

    if (((access_status = ap_run_check_user_id(r)) != 0)
        || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r)
            ? "check user.  No user file?"
            : "perform authentication. AuthType not
                r");
    }

    if (((access_status = ap_run_auth_checker(r)) != 0)
        || !ap_auth_type(r)) {
        return decl_die(access_status, ap_auth_type(r)
            ? "check access.  No groups file?"
            : "perform authentication. AuthType not
                r");
    }
}
```

```
        r);  
    }  
    }  
    break;  
}
```



## Hook: `type_checker`

The modules have an opportunity to test the URI or filename against the target resource, and set mime information for the request. Both `mod_mime` and `mod_mime_magic` use this phase to compare the file name or contents against the administrator's configuration and set the content type, language, character set and request handler. Some modules may set up their filters or other request handling parameters at this time.

If all modules `DECLINE` this phase, an error 500 is returned to the browser, and a "couldn't find types" error is logged automatically.

## Hook: `fixups`

Many modules are 'trounced' by some phase above. The `fixups` phase is used by modules to 'reassert' their ownership or force the request's fields to their appropriate values. It isn't always the cleanest mechanism, but occasionally it's the only option.



## THE HANDLER PHASE

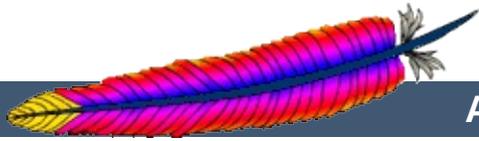
This phase is **not** part of the processing in `ap_process_request_internal()`. Many modules prepare one or more subrequests prior to creating any content at all. After the core, or a module calls `ap_process_request_internal()` it then calls `ap_invoke_handler()` to generate the request.

### **Hook: insert\_filter**

Modules that transform the content in some way can insert their values and override existing filters, such that if the user configured a more advanced filter out-of-order, then the module can move its order as need be. There is no result code, so actions in this hook better be trusted to always succeed.

### **Hook: handler**

The module finally has a chance to serve the request in its handler hook. Note that not every prepared request is sent to the handler hook. Many modules, such as `mod_autoindex`, will create subrequests for a given URI, and then never serve the subrequest, but simply lists it for the user. Remember not to put required teardown from the hooks above into this module, but register pool cleanups against the request pool to free resources as required.



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

## How filters work in Apache 2.0

### Warning

This is a cut 'n paste job from an email (<022501c1c529\$63a9550\$7f00000a@KOJ>) and only reformatted for better readability. It's not up to date but may be a good start for further research.



## Filter types

There are three basic filter types (each of these is actually broken down into two categories, but that comes later).

### **CONNECTION**

Filters of this type are valid for the lifetime of this connection. (AP\_FTYPE\_CONNECTION, AP\_FTYPE\_NETWORK)

### **PROTOCOL**

Filters of this type are valid for the lifetime of this request from the point of view of the client, this means that the request is valid from the time that the request is sent until the time that the response is received. (AP\_FTYPE\_PROTOCOL, AP\_FTYPE\_TRANSCODE)

### **RESOURCE**

Filters of this type are valid for the time that this content is used to satisfy a request. For simple requests, this is identical to PROTOCOL, but internal redirects and sub-requests can change the content without ending the request. (AP\_FTYPE\_RESOURCE, AP\_FTYPE\_CONTENT\_SET)

It is important to make the distinction between a protocol and a resource filter. A resource filter is tied to a specific resource, it may also be tied to header information, but the main binding is to a resource. If you are writing a filter and you want to know if it is resource or protocol, the correct question to ask is: "Can this filter be removed if the request is redirected to a different resource?" If the answer is yes, then it is a resource filter. If it is no, then it is most likely a protocol or connection filter. I won't go into connection filters, because they seem to be well understood. With this definition, a few examples might help:

### **Byterange**

We have coded it to be inserted for all requests, and it is removed if not used. Because this filter is active at the

beginning of all requests, it can not be removed if it is redirected, so this is a protocol filter.

### **http\_header**

This filter actually writes the headers to the network. This is obviously a required filter (except in the asis case which is special and will be dealt with below) and so it is a protocol filter.

### **Deflate**

The administrator configures this filter based on which file has been requested. If we do an internal redirect from an autoindex page to an index.html page, the deflate filter may be added or removed based on config, so this is a resource filter.

The further breakdown of each category into two more filter types is strictly for ordering. We could remove it, and only allow for one filter type, but the order would tend to be wrong, and we would need to hack things to make it work. Currently, the RESOURCE filters only have one filter type, but that should change.



This is actually rather simple in theory, but the code is complex. First of all, it is important that everybody realize that there are three filter lists for each request, but they are all concatenated together. So, the first list is `r->output_filters`, then `r->proto_output_filters`, and finally `r->connection->output_filters`. These correspond to the RESOURCE, PROTOCOL, and CONNECTION filters respectively. The problem previously, was that we used a singly linked list to create the filter stack, and we started from the "correct" location. This means that if I had a RESOURCE filter on the stack, and I added a CONNECTION filter, the CONNECTION filter would be ignored. This should make sense, because we would insert the connection filter at the top of the `c->output_filters` list, but the end of `r->output_filters` pointed to the filter that used to be at the front of `c->output_filters`. This is obviously wrong. The new insertion code uses a doubly linked list. This has the advantage that we never lose a filter that has been inserted. Unfortunately, it comes with a separate set of headaches.

The problem is that we have two different cases where we use subrequests. The first is to insert more data into a response. The second is to replace the existing response with an internal redirect. These are two different cases and need to be treated as such.

In the first case, we are creating the subrequest from within a handler or filter. This means that the next filter should be passed to `make_sub_request` function, and the last resource filter in the sub-request will point to the next filter in the main request. This makes sense, because the sub-request's data needs to flow through the same set of filters as the main request. A graphical representation might help:

```
Default_handler --> includes_filter --> byterange --> ...
```

If the includes filter creates a sub request, then we don't want the data from that sub-request to go through the includes filter, because it might not be SSI data. So, the subrequest adds the following:

```
Default_handler --> includes_filter -/-> byterange --> ...  
/   
Default_handler --> sub_request_core
```

What happens if the subrequest is SSI data? Well, that's easy, the `includes_filter` is a resource filter, so it will be added to the sub request in between the `Default_handler` and the `sub_request_core` filter.

The second case for sub-requests is when one sub-request is going to become the real request. This happens whenever a sub-request is created outside of a handler or filter, and NULL is passed as the next filter to the `make_sub_request` function.

In this case, the resource filters no longer make sense for the new request, because the resource has changed. So, instead of starting from scratch, we simply point the front of the resource filters for the sub-request to the front of the protocol filters for the old request. This means that we won't lose any of the protocol filters, neither will we try to send this data through a filter that shouldn't see it.

The problem is that we are using a doubly-linked list for our filter stacks now. But, you should notice that it is possible for two lists to intersect in this model. So, you do you handle the previous pointer? This is a very difficult question to answer, because there is no "right" answer, either method is equally valid. I looked at why we use the previous pointer. The only reason for it is to allow for easier addition of new servers. With that being said, the solution I

chose was to make the previous pointer always stay on the original request.

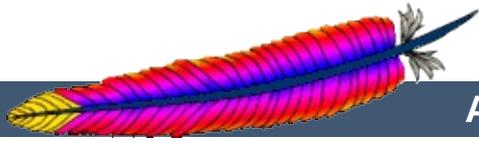
This causes some more complex logic, but it works for all cases. My concern in having it move to the sub-request, is that for the more common case (where a sub-request is used to add data to a response), the main filter chain would be wrong. That didn't seem like a good idea to me.



The final topic. :-) Mod\_Asis is a bit of a hack, but the handler needs to remove all filters except for connection filters, and send the data. If you are using mod\_asis, all other bets are off.



The absolutely last point is that the reason this code was so hard to get right, was because we had hacked so much to force it to work. I wrote most of the hacks originally, so I am very much to blame. However, now that the code is right, I have started to remove some hacks. Most people should have seen that the `reset_filters` and `add_required_filters` functions are gone. Those inserted protocol level filters for error conditions, in fact, both functions did the same thing, one after the other, it was really strange. Because we don't lose protocol filters for error cases any more, those hacks went away. The `HTTP_HEADER`, `Content-length`, and `Byterange` filters are all added in the `insert_filters` phase, because if they were added earlier, we had some interesting interactions. Now, those could all be moved to be inserted with the `HTTP_IN`, `CORE`, and `CORE_IN` filters. That would make the code easier to follow.



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

# Glossar

Diese Übersetzung ist möglicherweise nicht mehr aktuell. Bitte prüfen Sie die englische Version auf die neuesten Änderungen.

Dieses Glossar erläutert einige gebräuchliche Fachbegriffe im Zusammenhang mit dem Apache im Speziellen und Web-Diensten im Allgemeinen. Weitere Informationen zum jeweiligen Begriff erreichen Sie über die Links.



## Algorithmus

Eine eindeutige Formel oder ein Satz von Regeln zur Lösung eines Problems in einer endlichen Anzahl von Schritten. Algorithmen zur Verschlüsselung werden üblicherweise → [Chiffre](#) genannt.

## APache eXtension Tool (apxs)

Ein Perl-Skript zur Kompilierung von [Modul](#)-Quelltexten zu Dynamic-Shared-Objects (→ [DSOs](#)) und zur Installation dieser zum Apache-Webserver. Siehe: [apxs](#)-Dokumentation

## Authentifizierung

Die positive Identifizierung einer Netzwerkeinheit, wie z.B. eines Servers, eines Clients oder eines Benutzers. Siehe: [Authentisierung, Autorisierung und Zugriffskontrolle](#)

## Certification Authority [sə'tifi'keifən ɔ:θɔriti] (CA)

(Anm.d.Ü.: die Zertifizierungsstelle) Eine vertrauenswürdige dritte Partei, deren Zweck es ist, Zertifikate für Netzwerkeinheiten zu signieren. Andere Netzwerkeinheiten können die Signatur prüfen, um sicherzustellen, dass eine CA den Inhaber eines Zertifikats authentifiziert hat. Siehe: [SSL/TLS-Verschlüsselung](#)

## Certificate Signing Request [sə'tifikit sainiŋ ri'kwɛst] (CSR)

(Anm.d.Ü.: Zertifikats-Signierungsanfrage) Ein unsigniertes → [Zertifikat](#) zur Einreichung bei einer → [Zertifizierungsstelle](#), welche es mit dem → [privaten Schlüssel](#) ihres CA-Zertifikats signiert. Durch die Signatur wird ein CSR zum echten Zertifikat. Siehe: [SSL/TLS-Verschlüsselung](#)

## Chiffre

Die *Chiffre* ist ein Algorithmus oder System zur Datenverschlüsselung. Beispiele sind DES, IDEA, RC4 usw.

Im Englischen spricht man von *Cipher* ['saifə]

Siehe: [SSL/TLS-Verschlüsselung](#)

### Chiffretext

Das Ergebnis, nachdem ein → [Klartext](#) eine → [Chiffre](#) durchlaufen hat.

Siehe: [SSL/TLS-Verschlüsselung](#)

### Common Gateway Interface ['kɒmən geitwei 'intə:feɪs] (CGI)

Eine einheitliche Definition einer Schnittstelle zwischen einem Webserver und einem externen Programm, welches dem externen Programm die Behandlung von Anfragen ermöglicht. Die Schnittstelle ist ursprünglich von der [NCSA](#) definiert worden. Es existiert jedoch auch ein [RFC-Projekt](#).

Siehe: [Dynamische Inhalte mit CGI](#)

### CONNECT [kənekt]

Eine → [HTTP-Methode](#) zur Weiterleitung von Rohdaten über HTTP. Sie kann dazu verwendet werden, andere Protokolle wie zum Beispiel das SSL-Protokoll zu kapseln.

### Digitale Signatur

Ein chiffrierter Textblock, der die Gültigkeit eines Zertifikats oder einer anderen Datei bestätigt. Eine → [Zertifizierungsstelle](#) erstellt eine digitale Signatur durch Generierung eines → [Hashs](#) aus dem in einem *Zertifikat* enthaltenen *öffentlichen Schlüssel* und anschließender Codierung des Hashs mit dem *privaten Schlüssel* des Zertifikats. Nur der öffentliche Schlüssel der CA kann die Signatur decodieren. So wird sichergestellt, dass die CA die Netwerkeinheit, welche das *Zertifikat* besitzt, authentifiziert hat.

Siehe: [SSL/TLS-Verschlüsselung](#)

### Direktive

Eine Konfigurationsanweisung, die das Verhalten des Apache in einem oder mehreren Punkten steuert. Direktiven werden in

den → [Konfigurationsdateien](#) gesetzt.

Siehe: [Verzeichnis der Direktiven](#)

### **Dynamic Shared Object** [dai'næmik ʃæd 'ɔbdʒikt] **(DSO)**

Separat von der Apache-Binärdatei [httpd](#) kompilierte

→ [Module](#), die bei Bedarf geladen werden können.

Siehe: [Unterstützung für Dynamic-Shared-Objects](#)

### **exportbeschränkt**

Verminderte kryptografische Stärke (und Sicherheit), um den Exportbestimmungen der Vereinigten Staaten (*Anm.d.Ü.:*

konkret: United States' Export Administration Regulations (EAR)) zu entsprechen. Exportbeschränkte

Verschlüsselungssoftware ist auf eine kurze Schlüssellänge begrenzt, was zu *Chiffretexten* führt, die gewöhnlich mittels Brute-Force dekodiert werden können.

Siehe: [SSL/TLS-Verschlüsselung](#)

### **Filter**

Ein Verfahren, dass auf vom Server empfangene oder zu sendende Daten angewendet wird. Eingabefilter verarbeiten vom Client an den Server gesendete Daten, während Ausgabefilter vom Server an den Client zu sendende Daten verarbeiten. Der Ausgabefilter INCLUDES beispielsweise untersucht Dokumente nach → [Server-Side-Includes](#) und führt sie aus.

Siehe: [Filter](#)

### **Handler** ['hændlə]

Eine Apache-interne Darstellung der Aktion, die beim Aufruf einer Datei auszuführen ist. Im Allgemeinen besitzen Dateien implizite, auf dem Dateityp basierende Handler. Gewöhnlich werden alle Dateien vom Server bedient, einige Dateitypen werden jedoch separat "behandelt" (*Anm.d.Ü.:* besitzen einen separaten Handler). Der `cgi-script`-Handler

beispielsweise kennzeichnet Dateien, die als → [CGI-](#)

[Programme](#) ausgeführt werden sollen.

Siehe: [Verwendung von Apache-Handlern](#)

## Hash [hæʃ]

Ein mathematischer, unumkehrbarer Einweg-Algorithmus zur Generierung einer Zeichenfolge fester Länge aus einer anderen Zeichenfolge beliebiger Länge. Unterschiedliche Zeichenfolgen bei der Eingabe ergeben üblicherweise unterschiedliche Hashes (abhängig von der Hash-Funktion).

## Header [hedə]

Der Teil der → [HTTP](#)-Anfrage und -Antwort, der vor den eigentlichen Daten übermittelt wird und den Inhalt beschreibende Meta-Informationen enthält.

## .htaccess

Eine → [Konfigurationsdatei](#), die innerhalb des Web-Verzeichnisbaums abgelegt wird und zu dem Verzeichnis, in dem sie abgelegt ist, sowie allen Unterverzeichnissen → [Konfigurationsdirektiven](#) enthält. Trotz ihres Namens kann diese Datei nahezu alle Arten von Direktiven enthalten, nicht nur Direktiven zur Zugriffskontrolle.

Siehe: [Konfigurationsdateien](#)

## httpd.conf

Die → [Haupt-Konfigurationsdatei](#) ist `/usr/local/apache2/conf/httpd.conf`. Dies kann aber zur Laufzeit oder zur Kompilierungszeit anders konfiguriert werden.

Siehe: [Konfigurationsdateien](#)

## HTTPS

Das HyperText-Transfer-Protokoll (Secure), der Standard-Verschlüsselungsmechanismus im World Wide Web.

Tatsächlich handelt es sich hierbei um HTTP über → [SSL](#).

Siehe: [SSL/TLS-Verschlüsselung](#)

## HyperText-Transfer-Protokoll (HTTP)

Das Standard-Übertragungsprotokoll im World Wide Web. Der Apache implementiert die Protokollversion 1.1, bezeichnet als HTTP/1.1 und definiert in [RFC 2616](#).

### **Klartext**

Der unverschlüsselte Text.

### **Konfigurationsanweisung**

Siehe: → [Direktive](#)

### **Konfigurationsdatei**

Eine Textdatei mit → [Direktiven](#), welche die Konfiguration des Apache steuern.

Siehe: [Konfigurationsdateien](#)

### **Kontext**

Ein Bereich in den → [Konfigurationsdateien](#), in dem verschiedene Typen von → [Direktiven](#) erlaubt sind.

Siehe: [Erklärung der Fachbegriffe zu Apache-Direktiven](#)

### **Message-Digest** ['mesidʒ]

Ein Hash einer Nachricht, mit dem sich sicherstellen läßt, dass der Inhalt der Nachricht während der Übertragung nicht verändert wurde. (*Anm.d.Ü.:* ein so genannter Extrakt der Nachricht)

Siehe: [SSL/TLS-Verschlüsselung](#)

### **Methode**

Im → [HTTP](#)-Kontext eine in der Anfrage(zeile) des Clients angegebene Aktion, die auf eine Ressource angewendet wird. GET, POST und PUT sind einige der verfügbaren HTTP-Methoden.

### **MIME-Typ** [maim ty:p]

Eine Art und Weise, den Typ des übermittelten Dokuments zu beschreiben. Sein Name leitet sich davon ab, dass sein Format den Multipurpose Internet Mail Extensions entlehnt wurde. Er besteht aus einem Haupttyp und einem Untertyp,

getrennt durch einen Schrägstrich. Einige Beispiele sind `text/html`, `image/gif` und `application/octet-stream`. Bei HTTP wird der MIME-Typ mit dem → [Header Content-Type](#) übermittelt.

Siehe: [mod\\_mime](#)

## Modul

Ein selbstständiger Teil eines Programms. Ein Großteil der Funktionalität des Apache ist in Modulen enthalten, die Sie einbinden oder entfernen können. In die Apache-Binärdatei [httpd](#) einkompilierte Module werden *statische Module* genannt, während Module, die separat gespeichert sind und optional zur Laufzeit geladen werden können, *dynamische Module* oder → [DSOs](#) genannt werden. Standardmäßig eingebundene Module werden *Basismodule* genannt. Für den Apache sind viele Module verfügbar, die nicht als Bestandteil des → [Apache-HTTP-Server-Tarballs](#) ausgeliefert werden. Diese werden als *Drittmodule* bezeichnet.

Siehe: [Modulverzeichnis](#)

## Module-Magic-Number ['mɔju:l mædʒɪk 'nʌmbə] (MMN)

Die Module-Magic-Number ist eine Konstante, die im Apache-Quelltext definiert ist und im Zusammenhang mit der Binärkompatibilität von Modulen steht. Sie wird geändert, wenn sich interne Apache-Strukturen, -Funktionen oder andere signifikante Teile der API derart ändern, dass eine Binärkompatibilität nicht mehr gewährleistet werden kann. Bei einer MMN-Änderung müssen alle Module von Drittanbietern zumindest neu kompiliert und zuweilen auch geringfügig angepaßt werden, um mit der neuen Apache-Version zu funktionieren.

## Öffentlicher Schlüssel

Der öffentlich verfügbare Schlüssel in einem → [Public-Key-Kryptographie](#)-System, mit dem für seinen Eigentümer bestimmte Nachrichten verschlüsselt und Signaturen von

seinem Eigentümer entschlüsselt werden.

Siehe: [SSL/TLS-Verschlüsselung](#)

### **OpenSSL** ['əʊpənɛsɛs'ɛl]

Das Open-Source-Toolkit für SSL/TLS

Siehe: <http://www.openssl.org/>

### **Passphrase** [pa:freiz]

Das Wort oder die Phrase, welches private Schlüssel-Dateien schützt. Sie verhindert die Entschlüsselung durch nicht autorisierte Benutzer. Normalerweise ist dies einfach der geheimen (De-)Codierungsschlüssel, der für → [Chiffren](#) verwendet wird.

Siehe: [SSL/TLS-Verschlüsselung](#)

### **Privater Schlüssel**

Der geheime Schlüssel in einem → [Public-Key-Kryptographie](#)-System, mit dem hereinkommende Nachrichten decodiert und ausgehende signiert werden.

Siehe: [SSL/TLS-Verschlüsselung](#)

### **Proxy**

Ein zwischen dem Client und dem *ursprünglichen Server* (*Anm.d.Ü.:* der Server, den der Client tatsächlich erreichen möchte) liegender Server. Er nimmt Anfragen von Clients entgegen, übermittelt diese Anfragen dem ursprünglichen Server und liefert die Antwort des ursprünglichen Servers an den Client zurück. Wenn mehrere Clients den gleichen Inhalt abfragen, dann kann der Proxy diesen Inhalt aus seinem Zwischenspeicher ausliefern, anstatt ihn jedesmal vom ursprünglichen Server anzufordern, und dadurch die Antwortzeit verringern.

Siehe: [mod\\_proxy](#)

### **Public-Key-Kryptographie** ['pʌblik ki: 'kryptogra'fi:]

Theorie und Anwendung asymmetrischer Verschlüsselungssysteme, die einen Schlüssel zur

Verschlüsselung und einen anderen zur Entschlüsselung verwenden. Zwei derart zusammengehörende Schlüssel bilden Schüsselpaar. Man spricht auch von "Asymetrischer Kryptographie".

Siehe: [SSL/TLS-Verschlüsselung](#)

### **Regulärer Ausdruck (Regex)**

Eine Form, ein Muster im Text zu beschreiben - zum Beispiel: "alle Wörter, die mit dem Buchstaben A beginnen" oder "Jeder Satz mit zwei Kommata und ohne großes Q". Beim Apache sind reguläre Ausdrücke hilfreich, da sie auf sehr flexible Art und Weise die Anwendung bestimmter Eigenschaften auf eine Auswahl von Dateien oder Ressourcen ermöglichen. - Zum Beispiel können alle .gif- und .jpg-Dateien eines Verzeichnis "images" mit `"/images/.*(jpg|gif)$"` beschrieben werden. Der Apache verwendet Perl-kompatible reguläre Ausdrücke, wie sie die [PCRE](#)-Bibliothek bereitstellt.

### **Reverse Proxy** [ri:və:s 'prɔksi]

Ein → [Proxy](#)-Server, der dem Client gegenüber als *ursprünglicher Server* erscheint. Dies ist nützlich, um den tatsächlichen Server aus Sicherheitsgründen oder zur Lastverteilung vor dem Client zu verstecken.

### **Secure Sockets Layer** [si'kjuə 'sɔkits 'leia] (**SSL**)

Ein von der Firma Netscape Communications Corporation entwickeltes Protokoll zur allgemeinen Authentisierung und Verschlüsselung der Kommunikation über TCP/IP-Netzwerke. Die meistverbreitete Nutzung ist *HTTPS*, d.h. HyperText Transfer Protocol (HTTP) über SSL.

Siehe: [SSL/TLS-Verschlüsselung](#)

### **Server Side Includes** [sə:ə said in'klu:ds] (**SSI**)

Eine Technik zum Einbetten von weiterverarbeitenden Anweisungen in HTML-Dateien.

Siehe: [Einführung in Server Side Includes](#)

**Session** ['seʃən]

Allgemein der Kontext einer Kommunikation.

**SSL**

Die Bibliothek der Original-SSL/TLS-Implementation von Eric A. Young

**Symmetrische Kryptographie**

Die Theorie und Anwendung von *Chiffren*, die einen einzigen geheimen Schlüssel sowohl zur Verschlüsselung als auch zur Entschlüsselung benutzen.

Siehe: [SSL/TLS-Verschlüsselung](#)

**Tarball** [ta:bɔ:l]

Ein Paket von Dateien, die mit dem Hilfsprogramm tar zusammengefasst wurden. Apache-Distributionen werden in komprimierten tar-Archiven oder unter Verwendung von pkzip gespeichert.

**Transport Layer Security** [træns'pɔ:t 'eiə si'kjuəriiti] **(TLS)**

Das SSL-Nachfolgeprotokoll, das von der Internet Engineering Task Force (IETF) zur allgemeinen Authentisierung und Verschlüsselung einer Kommunikation über TCP/IP-Netzwerke entwickelt worden ist. TLS Version 1 ist nahezu identisch mit SSL Version 3.

Siehe: [SSL/TLS-Verschlüsselung](#)

**Umgebungsvariable (env-Variable)**

Benannte, von der Betriebssystem-Shell verwaltete Variablen zur Speicherung von Informationen und zur Kommunikation zwischen Programmen. Der Apache beinhaltet auch interne Variablen, die ebenfalls Umgebungsvariablen genannt werden, die aber statt in der Shell-Umgebung in internen Apache-Strukturen gespeichert sind.

Siehe: [Umgebungsvariablen im Apache](#)

**Uniform Resource Locator** ['ju:nifo:m ri'sɔ:s læu'keitə] **(URL)**

Der Name bzw. die Adresse einer Ressource im Internet. Dies

ist der allgemein gebräuchliche Ausdruck für die formale Bezeichnung → [Uniform Resource Identifier](#). URLs bestehen üblicherweise aus einem Schema wie `http` oder `https`, einem Hostnamen und einem Pfad. Die URL für diese Seite ist

`http://httpd.apache.org/docs/2.0/glossary.html`

### **Uniform Resource Identifier** [ˈjuːnɪfɔːm riːsɔːs aɪˈdentɪfaɪə] **(URI)**

Eine kompakte Zeichenfolge zur Identifizierung einer abstrakten oder physischen Ressource. Er wird in dem [RFC 2396](#) formell definiert. Im World Wide Web verwendete URIs werden üblicherweise als → [URLs](#) bezeichnet.

### **Virtual-Hosting** [vəˈtʃuəl həʊstɪŋ]

Die Bedienung mehrere Websites mit einer einzigen Apache-Instanz. *IP-basierte virtuelle Hosts* unterscheiden zwischen verschiedenen Websites aufgrund ihrer IP-Adressen, während *namensbasierte virtuelle Hosts* nur den Namen des Hosts verwenden und daher mehrere Angebote unter der gleichen IP-Adresse hosten können.

Siehe: [Apache-Dokumentation zu virtuellen Hosts](#)

### **Voll-qualifizierter Domainname (FQDN)**

Der eindeutige Name einer Netzwerkeinheit, bestehend aus einem Hostnamen und dem Domainnamen, welcher zu einer IP-Adresse aufgelöst werden kann. Zum Beispiel ist `www` ein Hostname, `example.com` ein Domainname und `www.example.com` ein voll-qualifizierter Domainname.

### **Website** [weɪbsaɪt]

Im Gegensatz zur Webseite, die einer konkreten URL entspricht, ist mit Website ein komplettes Angebot unter einem bestimmten Hostnamen (und Port) gemeint. Dieses kann aus vielen verschiedenen Webseiten bestehen.

### **X.509**

Ein von der International Telecommunication Union (ITU-T)

empfohlenes Schema für Authentifizierungszertifikate. Es wird für SSL/TLS-Authentifizierungen verwendet.

Siehe: [SSL/TLS-Verschlüsselung](#)

## Zertifikat

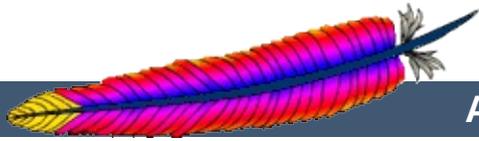
Ein Datensatz zur → [Authentisierung](#) einer Netzwerkeinheit wie Server oder Client. Ein Zertifikat enthält → [X.509](#)-Informationen über seinen Eigentümer (das sogenannte *Betreff* (*Anm.d.Ü.:* engl.: subject)) und die signierende → [Certification Authority](#) (der sogenannte Aussteller (*Anm.d.Ü.:* engl.: issuer)) sowie den → [öffentlichen Schlüssel](#) des Eigentümers und die Signatur der CA. Netzwerkeinheiten überprüfen diese Signatur mit Hilfe von CA-Zertifikaten.

Siehe: [SSL/TLS-Verschlüsselung](#)

## Zugriffskontrolle

Die Beschränkung des Zugriffs auf Netzwerkbereiche. Im Apache-Kontext in der Regel die Zugriffsbeschränkung auf bestimmte *URLs*.

Siehe: [Authentisierung, Autorisierung und Zugriffskontrolle](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

# Verzeichnis der Direktiven

Hier sind alle Apache-Direktiven aufgeführt, die in der Standard-Apache-Distribution verfügbar sind. Sie sind in einem einheitlichen Format beschrieben. Ein [Glossar](#) erläutert die in der Beschreibung verwendeten Begriffe.

Außerdem existiert eine [Kurzreferenz der Direktiven](#), welche zu jeder Direktive eine Zusammenfassung der Details enthält.

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) |  
[R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#) | [X](#)

- [AcceptMutex](#)
- [AcceptPathInfo](#)
- [AccessFileName](#)
- [Action](#)
- [AddAlt](#)
- [AddAltByEncoding](#)
- [AddAltByType](#)
- [AddCharset](#)
- [AddDefaultCharset](#)
- [AddDescription](#)
- [AddEncoding](#)
- [AddHandler](#)
- [AddIcon](#)
- [AddIconByEncoding](#)
- [AddIconByType](#)
- [AddInputFilter](#)
- [AddLanguage](#)
- [AddModuleInfo](#)
- [AddOutputFilter](#)
- [AddOutputFilterByType](#)
- [AddType](#)

- [Alias](#)
- [AliasMatch](#)
- [Allow](#)
- [AllowCONNECT](#)
- [AllowEncodedSlashes](#)
- [AllowOverride](#)
- [Anonymous](#)
- [Anonymous\\_Authoritative](#)
- [Anonymous\\_LogEmail](#)
- [Anonymous\\_MustGiveEmail](#)
- [Anonymous\\_NoUserID](#)
- [Anonymous\\_VerifyEmail](#)
- [AssignUserID](#)
- [AuthAuthoritative](#)
- [AuthDBMAuthoritative](#)
- [AuthDBMGroupFile](#)
- [AuthDBMType](#)
- [AuthDBMUserFile](#)
- [AuthDigestAlgorithm](#)
- [AuthDigestDomain](#)
- [AuthDigestFile](#)
- [AuthDigestGroupFile](#)
- [AuthDigestNcCheck](#)
- [AuthDigestNonceFormat](#)
- [AuthDigestNonceLifetime](#)
- [AuthDigestQop](#)
- [AuthDigestShmemSize](#)
- [AuthGroupFile](#)
- [AuthLDAPAuthoritative](#)
- [AuthLDAPBindDN](#)
- [AuthLDAPBindPassword](#)
- [AuthLDAPCharsetConfig](#)
- [AuthLDAPCompareDNOnServer](#)
- [AuthLDAPDereferenceAliases](#)

- [AuthLDAPEnabled](#)
- [AuthLDAPFrontPageHack](#)
- [AuthLDAPGroupAttribute](#)
- [AuthLDAPGroupAttributeIsDN](#)
- [AuthLDAPRemoteUserIsDN](#)
- [AuthLDAPUrl](#)
- [AuthName](#)
- [AuthType](#)
- [AuthUserFile](#)
- [BrowserMatch](#)
- [BrowserMatchNoCase](#)
- [BS2000Account](#)
- [BufferedLogs](#)
- [CacheDefaultExpire](#)
- [CacheDirLength](#)
- [CacheDirLevels](#)
- [CacheDisable](#)
- [CacheEnable](#)
- [CacheExpiryCheck](#)
- [CacheFile](#)
- [CacheForceCompletion](#)
- [CacheGcClean](#)
- [CacheGcDaily](#)
- [CacheGcInterval](#)
- [CacheGcMemUsage](#)
- [CacheGcUnused](#)
- [CacheIgnoreCacheControl](#)
- [CacheIgnoreHeaders](#)
- [CacheIgnoreNoLastMod](#)
- [CacheLastModifiedFactor](#)
- [CacheMaxExpire](#)
- [CacheMaxFileSize](#)
- [CacheMinFileSize](#)
- [CacheNegotiatedDocs](#)

- [CacheRoot](#)
- [CacheSize](#)
- [CacheTimeMargin](#)
- [CGIMapExtension](#)
- [CharsetDefault](#)
- [CharsetOptions](#)
- [CharsetSourceEnc](#)
- [CheckSpelling](#)
- [ChildPerUserID](#)
- [ContentDigest](#)
- [CookieDomain](#)
- [CookieExpires](#)
- [CookieLog](#)
- [CookieName](#)
- [CookieStyle](#)
- [CookieTracking](#)
- [CoreDumpDirectory](#)
- [CustomLog](#)
- [Dav](#)
- [DavDepthInfinity](#)
- [DavLockDB](#)
- [DavMinTimeout](#)
- [DefaultIcon](#)
- [DefaultLanguage](#)
- [DefaultType](#)
- [DeflateBufferSize](#)
- [DeflateCompressionLevel](#)
- [DeflateFilterNote](#)
- [DeflateMemLevel](#)
- [DeflateWindowSize](#)
- [Deny](#)
- [<Directory>](#)
- [DirectoryIndex](#)
- [<DirectoryMatch>](#)

- [DirectorySlash](#)
- [DocumentRoot](#)
- [DumpIOInput](#)
- [DumpIOOutput](#)
- [EnableExceptionHook](#)
- [EnableMMAP](#)
- [EnableSendfile](#)
- [ErrorDocument](#)
- [ErrorLog](#)
- [Example](#)
- [ExpiresActive](#)
- [ExpiresByType](#)
- [ExpiresDefault](#)
- [ExtendedStatus](#)
- [ExtFilterDefine](#)
- [ExtFilterOptions](#)
- [FileETag](#)
- [<Files>](#)
- [<FilesMatch>](#)
- [ForceLanguagePriority](#)
- [ForceType](#)
- [ForensicLog](#)
- [Group](#)
- [Header](#)
- [HeaderName](#)
- [HostnameLookups](#)
- [IdentityCheck](#)
- [<IfDefine>](#)
- [<IfModule>](#)
- [<IfVersion>](#)
- [ImapBase](#)
- [ImapDefault](#)
- [ImapMenu](#)
- [Include](#)

- [IndexIgnore](#)
- [IndexOptions](#)
- [IndexOrderDefault](#)
- [ISAPIAppendLogToErrors](#)
- [ISAPIAppendLogToQuery](#)
- [ISAPICacheFile](#)
- [ISAPIFakeAsync](#)
- [ISAPILogNotSupported](#)
- [ISAPIReadAheadBuffer](#)
- [KeepAlive](#)
- [KeepAliveTimeout](#)
- [LanguagePriority](#)
- [LDAPCacheEntries](#)
- [LDAPCacheTTL](#)
- [LDAPConnectionTimeout](#)
- [LDAPOpCacheEntries](#)
- [LDAPOpCacheTTL](#)
- [LDAPSharedCacheFile](#)
- [LDAPSharedCacheSize](#)
- [LDAPTrustedCA](#)
- [LDAPTrustedCAType](#)
- [<Limit>](#)
- [<LimitExcept>](#)
- [LimitInternalRecursion](#)
- [LimitRequestBody](#)
- [LimitRequestFields](#)
- [LimitRequestFieldSize](#)
- [LimitRequestLine](#)
- [LimitXMLRequestBody](#)
- [Listen](#)
- [ListenBackLog](#)
- [LoadFile](#)
- [LoadModule](#)
- [<Location>](#)

- [<LocationMatch>](#)
- [LockFile](#)
- [LogFormat](#)
- [LogLevel](#)
- [MaxClients](#)
- [MaxKeepAliveRequests](#)
- [MaxMemFree](#)
- [MaxRanges](#)
- [MaxRequestsPerChild](#)
- [MaxRequestsPerThread](#)
- [MaxSpareServers](#)
- [MaxSpareThreads](#)
- [MaxThreads](#)
- [MaxThreadsPerChild](#)
- [MCacheMaxObjectCount](#)
- [MCacheMaxObjectSize](#)
- [MCacheMaxStreamingBuffer](#)
- [MCacheMinObjectSize](#)
- [MCacheRemovalAlgorithm](#)
- [MCacheSize](#)
- [MetaDir](#)
- [MetaFiles](#)
- [MetaSuffix](#)
- [MimeMagicFile](#)
- [MinSpareServers](#)
- [MinSpareThreads](#)
- [MMapFile](#)
- [ModMimeUsePathInfo](#)
- [MultiviewsMatch](#)
- [NameVirtualHost](#)
- [NoProxy](#)
- [NumServers](#)
- [NWSSLTrustedCerts](#)
- [NWSSLUpgradeable](#)

- [Options](#)
- [Order](#)
- [PassEnv](#)
- [PidFile](#)
- [ProtocolEcho](#)
- [<Proxy>](#)
- [ProxyBadHeader](#)
- [ProxyBlock](#)
- [ProxyDomain](#)
- [ProxyErrorOverride](#)
- [ProxyFtpDirCharset](#)
- [ProxyIOBufferSize](#)
- [<ProxyMatch>](#)
- [ProxyMaxForwards](#)
- [ProxyPass](#)
- [ProxyPassReverse](#)
- [ProxyPreserveHost](#)
- [ProxyReceiveBufferSize](#)
- [ProxyRemote](#)
- [ProxyRemoteMatch](#)
- [ProxyRequests](#)
- [ProxyTimeout](#)
- [ProxyVia](#)
- [ReadmeName](#)
- [ReceiveBufferSize](#)
- [Redirect](#)
- [RedirectMatch](#)
- [RedirectPermanent](#)
- [RedirectTemp](#)
- [RemoveCharset](#)
- [RemoveEncoding](#)
- [RemoveHandler](#)
- [RemoveInputFilter](#)
- [RemoveLanguage](#)

- [RemoveOutputFilter](#)
- [RemoveType](#)
- [RequestHeader](#)
- [Require](#)
- [RewriteBase](#)
- [RewriteCond](#)
- [RewriteEngine](#)
- [RewriteLock](#)
- [RewriteLog](#)
- [RewriteLogLevel](#)
- [RewriteMap](#)
- [RewriteOptions](#)
- [RewriteRule](#)
- [RLimitCPU](#)
- [RLimitMEM](#)
- [RLimitNPROC](#)
- [Satisfy](#)
- [ScoreBoardFile](#)
- [Script](#)
- [ScriptAlias](#)
- [ScriptAliasMatch](#)
- [ScriptInterpreterSource](#)
- [ScriptLog](#)
- [ScriptLogBuffer](#)
- [ScriptLogLength](#)
- [ScriptSock](#)
- [SecureListen](#)
- [SendBufferSize](#)
- [ServerAdmin](#)
- [ServerAlias](#)
- [ServerLimit](#)
- [ServerName](#)
- [ServerPath](#)
- [ServerRoot](#)

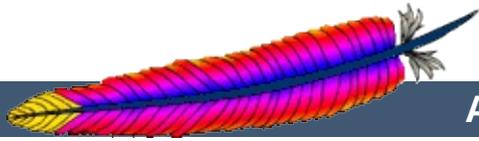
- [ServerSignature](#)
- [ServerTokens](#)
- [SetEnv](#)
- [SetEnvIf](#)
- [SetEnvIfNoCase](#)
- [SetHandler](#)
- [SetInputFilter](#)
- [SetOutputFilter](#)
- [SSIEndTag](#)
- [SSIErrorMsg](#)
- [SSIStartTag](#)
- [SSITimeFormat](#)
- [SSIUndefinedEcho](#)
- [SSLCACertificateFile](#)
- [SSLCACertificatePath](#)
- [SSLCARevocationFile](#)
- [SSLCARevocationPath](#)
- [SSLCertificateChainFile](#)
- [SSLCertificateFile](#)
- [SSLCertificateKeyFile](#)
- [SSLCipherSuite](#)
- [SSLEngine](#)
- [SSLHonorCipherOrder](#)
- [SSLInsecureRenegotiation](#)
- [SSLMutex](#)
- [SSLOptions](#)
- [SSLPassPhraseDialog](#)
- [SSLProtocol](#)
- [SSLProxyCACertificateFile](#)
- [SSLProxyCACertificatePath](#)
- [SSLProxyCARevocationFile](#)
- [SSLProxyCARevocationPath](#)
- [SSLProxyCipherSuite](#)
- [SSLProxyEngine](#)

- [SSLProxyMachineCertificateFile](#)
- [SSLProxyMachineCertificatePath](#)
- [SSLProxyProtocol](#)
- [SSLProxyVerify](#)
- [SSLProxyVerifyDepth](#)
- [SSLRandomSeed](#)
- [SSLRequire](#)
- [SSLRequireSSL](#)
- [SSLSessionCache](#)
- [SSLSessionCacheTimeout](#)
- [SSLUserName](#)
- [SSLVerifyClient](#)
- [SSLVerifyDepth](#)
- [StartServers](#)
- [StartThreads](#)
- [SuexecUserGroup](#)
- [ThreadLimit](#)
- [ThreadsPerChild](#)
- [ThreadStackSize](#)
- [TimeOut](#)
- [TraceEnable](#)
- [TransferLog](#)
- [TypesConfig](#)
- [UnsetEnv](#)
- [UseCanonicalName](#)
- [User](#)
- [UserDir](#)
- [VirtualDocumentRoot](#)
- [VirtualDocumentRootIP](#)
- [<VirtualHost>](#)
- [VirtualScriptAlias](#)
- [VirtualScriptAliasIP](#)
- [Win32DisableAcceptEx](#)
- [XBitHack](#)

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#) > [Module](#)

## Kurzreferenz der Direktiven

Die Kurzreferenz der Direktiven zeigt die Verwendung, Voreinstellung, den Status und den Kontext aller Apache-Konfigurationsanweisungen. Für weitergehende Informationen schauen Sie bitte im [Verzeichnis der Direktiven](#).

Die erste Spalte enthält den Namen und die Verwendung. Die zweite Spalte zeigt die Voreinstellung der Direktive, sofern eine Voreinstellung existiert. Wenn die Voreinstellung zu breit für die Anzeige ist, werden die ersten Buchstaben angegeben, gefolgt von einem "+".

Die dritte und vierte Spalte geben den Kontext an, in dem die Direktive erlaubt ist, sowie den Status der Direktive entsprechend der Legende.

<b>A</b>	<b>B</b>	<b>C</b>	<b>D</b>	<b>E</b>	<b>F</b>	<b>s</b> Serverkonfiguration	<b>C</b> Core
<b>G</b>	<b>H</b>	<b>I</b>	<b>K</b>	<b>L</b>	<b>M</b>	<b>v</b> Virtual Host	<b>M</b> MPM
<b>N</b>	<b>O</b>	<b>P</b>	<b>R</b>	<b>S</b>	<b>T</b>	<b>d</b> Verzeichnis	<b>B</b> Basis
<b>U</b>	<b>V</b>	<b>W</b>	<b>X</b>			<b>h</b> .htaccess	<b>E</b> Erweiterung
							<b>X</b> experimentell

<a href="#">AcceptMutex</a> <i>Default <u>Methode</u></i>	Default
Vom Apache verwendete Methode zur Serialisierung mehrerer Kindprozesse, die Anfragen Sockets entgegennehmen.	
<a href="#">AcceptPathInfo</a> <i>On Off Default</i>	Default
Ressourcen lassen angehängte Pfadangaben zu	
<a href="#">AccessFileName</a> <i><u>Dateiname</u> [<u>Dateiname</u>] ...</i>	.htaccess
Name der dezentralen Konfigurationsdateien	
<a href="#">Action</a> <i><u>action-type</u> <u>cgi-script</u></i>	
Activates a CGI script for a particular handler or content-type	
<a href="#">AddAlt</a> <i><u>string</u> <u>file</u> [<u>file</u>] ...</i>	
Alternate text to display for a file, instead of an icon selected by filename	

<a href="#">AddAltByEncoding <i>string MIME-encoding [MIME-encoding] ...</i></a>	
Alternate text to display for a file instead of an icon selected by MIME-encoding	
<a href="#">AddAltByType <i>string MIME-type [MIME-type] ...</i></a>	
Alternate text to display for a file, instead of an icon selected by MIME content-type	
<a href="#">AddCharset <i>charset extension [extension] ...</i></a>	
Maps the given filename extensions to the specified content charset	
<a href="#">AddDefaultCharset <i>On Off Zeichenkodierung</i></a>	Off
Standard-Charset-Parameter, der bei Antworten vom Content-Type text/plain oder tex hinzugefügt wird	
<a href="#">AddDescription <i>string file [file] ...</i></a>	
Description to display for a file	
<a href="#">AddEncoding <i>MIME-enc extension [extension] ...</i></a>	
Maps the given filename extensions to the specified encoding type	
<a href="#">AddHandler <i>handler-name extension [extension] ...</i></a>	
Maps the filename extensions to the specified handler	
<a href="#">AddIcon <i>icon name [name] ...</i></a>	
Icon to display for a file selected by name	
<a href="#">AddIconByEncoding <i>icon MIME-encoding [MIME-encoding] ...</i></a>	
Icon to display next to files selected by MIME content-encoding	
<a href="#">AddIconByType <i>icon MIME-type [MIME-type] ...</i></a>	
Icon to display next to files selected by MIME content-type	
<a href="#">AddInputFilter <i>filter[;filter...] extension [extension] ...</i></a>	
Maps filename extensions to the filters that will process client requests	
<a href="#">AddLanguage <i>MIME-lang extension [extension] ...</i></a>	
Maps the given filename extension to the specified content language	
<a href="#">AddModuleInfo <i>module-name string</i></a>	
Adds additional information to the module information displayed by the server-info handler	

<a href="#">AddOutputFilter <i>filter[:filter...] extension [extension] ...</i></a>	
Maps filename extensions to the filters that will process responses from the server	
<a href="#">AddOutputFilterByType <i>Filter[:Filter...] MIME-Type [MIME-Type] ...</i></a>	
einen Ausgabefilter einem bestimmten MIME-Type zuordnen	
<a href="#">AddType <i>MIME-type extension [extension] ...</i></a>	
Maps the given filename extensions onto the specified content type	
<a href="#">Alias <i>URL-path file-path directory-path</i></a>	
Maps URLs to filesystem locations	
<a href="#">AliasMatch <i>regex file-path directory-path</i></a>	
Maps URLs to filesystem locations using regular expressions	
<a href="#">Allow from <i>all host env=env-variable [host env=env-variable] ...</i></a>	
Controls which hosts can access an area of the server	
<a href="#">AllowCONNECT <i>port [port] ...</i></a>	443 563
Ports that are allowed to CONNECT through the proxy	
<a href="#">AllowEncodedSlashes <i>On Off</i></a>	Off
Legt fest, ob kodierte Pfadtrennzeichen in URLs durchgereicht werden dürfen	
<a href="#">AllowOverride <i>All None Direktiven-Typ [Direktiven-Typ] ...</i></a>	All
Direktiven-Typen, die in .htaccess-Dateien erlaubt sind.	
<a href="#">Anonymous <i>user [user] ...</i></a>	
Specifies userIDs that are allowed access without password verification	
<a href="#">Anonymous_ <u>Authoritative</u> <i>On Off</i></a>	Off
Configures if authorization will fall-through to other methods	
<a href="#">Anonymous_ <u>LogEmail</u> <i>On Off</i></a>	On
Sets whether the password entered will be logged in the error log	
<a href="#">Anonymous_ <u>MustGiveEmail</u> <i>On Off</i></a>	On
Specifies whether blank passwords are allowed	
<a href="#">Anonymous_ <u>NoUserID</u> <i>On Off</i></a>	Off
Sets whether the userID field may be empty	
<a href="#">Anonymous_ <u>VerifyEmail</u> <i>On Off</i></a>	Off
Sets whether to check the password field for a correctly formatted email address	
<a href="#">AssignUserID <i>user-id group-id</i></a>	

	Tie a virtual host to a user and group ID	
<a href="#">AuthAuthoritative</a>	On Off	On
	Sets whether authorization and authentication are passed to lower level modules	
<a href="#">AuthDBMAuthoritative</a>	On Off	On
	Sets whether authentication and authorization will be passed on to lower level modules	
<a href="#">AuthDBMGroupFile</a>	<i>file-path</i>	
	Sets the name of the database file containing the list of user groups for authentication	
<a href="#">AuthDBMType</a>	default	default
	<a href="#">default SDBM GDBM NDBM DB</a>	
	Sets the type of database file that is used to store passwords	
<a href="#">AuthDBMUserFile</a>	<i>file-path</i>	
	Sets the name of a database file containing the list of users and passwords for authentication	
<a href="#">AuthDigestAlgorithm</a>	MD5 MD5-sess	MD5
	Selects the algorithm used to calculate the challenge and response hashes in digest authentication	
<a href="#">AuthDigestDomain</a>	<i>URI [URI] ...</i>	
	URIs that are in the same protection space for digest authentication	
<a href="#">AuthDigestFile</a>	<i>file-path</i>	
	Location of the text file containing the list of users and encoded passwords for digest authentication	
<a href="#">AuthDigestGroupFile</a>	<i>file-path</i>	
	Name of the text file containing the list of groups for digest authentication	
<a href="#">AuthDigestNcCheck</a>	On Off	Off
	Enables or disables checking of the nonce-count sent by the server	
<a href="#">AuthDigestNonceFormat</a>	<i>format</i>	
	Determines how the nonce is generated	
<a href="#">AuthDigestNonceLifetime</a>	<i>seconds</i>	300
	How long the server nonce is valid	
<a href="#">AuthDigestQop</a>	none auth auth-int [ <a href="#">auth auth-int</a> ]	auth
	Determines the quality-of-protection to use in digest authentication	
<a href="#">AuthDigestShmemSize</a>	<i>size</i>	1000
	The amount of shared memory to allocate for keeping track of clients	
<a href="#">AuthGroupFile</a>	<i>file-path</i>	
	Sets the name of a text file containing the list of user groups for authentication	
<a href="#">AuthLDAPAuthoritative</a>	on off	on
	Prevent other authentication modules from authenticating the user if this one fails	

<a href="#">AuthLDAPBindDN</a> <i>distinguished-name</i>	
Optional DN to use in binding to the LDAP server	
<a href="#">AuthLDAPBindPassword</a> <i>password</i>	
Password used in conjunction with the bind DN	
<a href="#">AuthLDAPCharsetConfig</a> <i>file-path</i>	
Language to charset conversion configuration file	
<a href="#">AuthLDAPCompareDNOnServer</a> <i>on off</i>	on
Use the LDAP server to compare the DN's	
<a href="#">AuthLDAPDereferenceAliases</a> <i>never searching finding always</i>	Always
When will the module de-reference aliases	
<a href="#">AuthLDAPEnabled</a> <i>on off</i>	on
Turn on or off LDAP authentication	
<a href="#">AuthLDAPFrontPageHack</a> <i>on off</i>	off
Allow LDAP authentication to work with MS FrontPage	
<a href="#">AuthLDAPGroupAttribute</a> <i>attribute</i>	
LDAP attributes used to check for group membership	
<a href="#">AuthLDAPGroupAttributeIsDN</a> <i>on off</i>	on
Use the DN of the client username when checking for group membership	
<a href="#">AuthLDAPRemoteUserIsDN</a> <i>on off</i>	off
Use the DN of the client username to set the REMOTE_USER environment variable	
<a href="#">AuthLDAPUrl</a> <i>url</i>	
URL specifying the LDAP search parameters	
<a href="#">AuthName</a> <i>auth-Bereich</i>	
Autorisierungsbereich zur Verwendung in der HTTP-Authentisierung	
<a href="#">AuthType</a> <i>Basic Digest</i>	
Art der Authentisierung	
<a href="#">AuthUserFile</a> <i>file-path</i>	
Sets the name of a text file containing the list of users and passwords for authentication	
<a href="#">BrowserMatch</a> <i>regex [!]env-variable[=value]</i> <i>[!]env-variable[=value]] ...</i>	
Sets environment variables conditional on HTTP User-Agent	
<a href="#">BrowserMatchNoCase</a> <i>regex [!]env-</i> <i>variable[=value] [!]env-variable[=value]] ...</i>	
Sets environment variables conditional on User-Agent without respect to case	

<a href="#">BS2000Account</a> <i>Account</i>	
Bestimmt den nicht-privilegierten Account auf BS2000-Maschinen	
<a href="#">BufferedLogs</a> <i>On Off</i>	Off
Buffer log entries in memory before writing to disk	
<a href="#">CacheDefaultExpire</a> <i>seconds</i>	3600 (one hour)
The default duration to cache a document when no expiry date is specified.	
<a href="#">CacheDirLength</a> <i>length</i>	2
The number of characters in subdirectory names	
<a href="#">CacheDirLevels</a> <i>levels</i>	3
The number of levels of subdirectories in the cache.	
<a href="#">CacheDisable</a> <i>url-string</i>	
Disable caching of specified URLs	
<a href="#">CacheEnable</a> <i>cache_type url-string</i>	
Enable caching of specified URLs using a specified storage manager	
<a href="#">CacheExpiryCheck</a> <i>On Off</i>	On
Indicates if the cache observes Expires dates when seeking files	
<a href="#">CacheFile</a> <i>file-path [file-path] ...</i>	
Cache a list of file handles at startup time	
<a href="#">CacheForceCompletion</a> <i>Percentage</i>	60
Percentage of document served, after which the server will complete caching the file even if cancelled.	
<a href="#">CacheGcClean</a> <i>hours url-string</i>	?
The time to retain unchanged cached files that match a URL	
<a href="#">CacheGcDaily</a> <i>time</i>	?
The recurring time each day for garbage collection to be run. (24 hour clock)	
<a href="#">CacheGcInterval</a> <i>hours</i>	
The interval between garbage collection attempts.	
<a href="#">CacheGcMemUsage</a> <i>KBytes</i>	?
The maximum kilobytes of memory used for garbage collection	
<a href="#">CacheGcUnused</a> <i>hours url-string</i>	?
The time to retain unreferenced cached files that match a URL.	
<a href="#">CacheIgnoreCacheControl</a> <i>On Off</i>	Off
Ignore the fact that the client requested the content not be cached.	
<a href="#">CacheIgnoreHeaders</a> <i>header-string [header-string] ...</i>	None

Do not store the given HTTP header(s) in the cache.	
<a href="#"><u>CacheIgnoreNoLastMod On Off</u></a>	Off
Ignore the fact that a response has no Last Modified header.	
<a href="#"><u>CacheLastModifiedFactor float</u></a>	0.1
The factor used to compute an expiry date based on the LastModified date.	
<a href="#"><u>CacheMaxExpire seconds</u></a>	86400 (one day)
The maximum time in seconds to cache a document	
<a href="#"><u>CacheMaxFileSize bytes</u></a>	1000000
The maximum size (in bytes) of a document to be placed in the cache	
<a href="#"><u>CacheMinFileSize bytes</u></a>	1
The minimum size (in bytes) of a document to be placed in the cache	
<a href="#"><u>CacheNegotiatedDocs On Off</u></a>	Off
Allows content-negotiated documents to be cached by proxy servers	
<a href="#"><u>CacheRoot directory</u></a>	
The directory root under which cache files are stored	
<a href="#"><u>CacheSize KBytes</u></a>	1000000
The maximum amount of disk space that will be used by the cache in KBytes	
<a href="#"><u>CacheTimeMargin ?</u></a>	?
The minimum time margin to cache a document	
<a href="#"><u>CGIMapExtension CGI-Pfad .Endung</u></a>	
Technik zur Bestimmung des Interpreters für CGI-Skripte	
<a href="#"><u>CharsetDefault charset</u></a>	
Charset to translate into	
<a href="#"><u>CharsetOptions option [option] ...</u></a>	DebugLevel=0 NoImp
Configures charset translation behavior	
<a href="#"><u>CharsetSourceEnc charset</u></a>	
Source charset of files	
<a href="#"><u>CheckSpelling on off</u></a>	Off
Enables the spelling module	
<a href="#"><u>ChildPerUserID user-id group-id num-children</u></a>	
Specify user ID and group ID for a number of child processes	
<a href="#"><u>ContentDigest On Off</u></a>	Off
Aktiviert die Generierung von Content - MD5 HTTP-Response-Headern	
<a href="#"><u>CookieDomain domain</u></a>	
The domain to which the tracking cookie applies	

<a href="#"><u>CookieExpires</u></a> <i>expiry-period</i>	
Expiry time for the tracking cookie	
<a href="#"><u>CookieLog</u></a> <i>filename</i>	
Sets filename for the logging of cookies	
<a href="#"><u>CookieName</u></a> <i>token</i>	Apache
Name of the tracking cookie	
<a href="#"><u>CookieStyle</u></a>	Netscape
<a href="#"><u>Netscape Cookie Cookie2 RFC2109 RFC2965</u></a>	
Format of the cookie header field	
<a href="#"><u>CookieTracking</u></a> on off	off
Enables tracking cookie	
<a href="#"><u>CoreDumpDirectory</u></a> <i>Verzeichnis</i>	
Verzeichnis, in das der Apache zu wechseln versucht, bevor er einen Hauptspeicherauszug	
<a href="#"><u>CustomLog</u></a> <i>file pipe format nickname [env=</i>	
<i>[!]<u>environment-variable</u></i>	
Sets filename and format of log file	
<a href="#"><u>Dav On Off provider-name</u></a>	Off
Enable WebDAV HTTP methods	
<a href="#"><u>DavDepthInfinity</u></a> on off	off
Allow PROPFIND, Depth: Infinity requests	
<a href="#"><u>DavLockDB</u></a> <i>file-path</i>	
Location of the DAV lock database	
<a href="#"><u>DavMinTimeout</u></a> <i>seconds</i>	0
Minimum amount of time the server holds a lock on a DAV resource	
<a href="#"><u>DefaultIcon</u></a> <i>url-path</i>	
Icon to display for files when no specific icon is configured	
<a href="#"><u>DefaultLanguage</u></a> <i>MIME-lang</i>	
Sets all files in the given scope to the specified language	
<a href="#"><u>DefaultType</u></a> <i>MIME-Type</i>	text/plain
MIME-Content-Type, der gesendet wird, wenn der Server den Typ nicht auf andere Weise e	
<a href="#"><u>DeflateBufferSize</u></a> <i>value</i>	8096
Fragment size to be compressed at one time by zlib	
<a href="#"><u>DeflateCompressionLevel</u></a> <i>value</i>	
How much compression do we apply to the output	
<a href="#"><u>DeflateFilterNote</u></a> [ <i>type</i> ] <i>notename</i>	

Places the compression ratio in a note for logging	
<a href="#"><u>DeflateMemLevel <i>value</i></u></a>	9
How much memory should be used by zlib for compression	
<a href="#"><u>DeflateWindowSize <i>value</i></u></a>	15
Zlib compression window size	
<a href="#"><u>Deny from all <i>host</i> <i>env=env-variable</i></u></a> <a href="#"><u>[<i>host</i> <i>env=env-variable</i>] ...</u></a>	
Controls which hosts are denied access to the server	
<a href="#"><u>&lt;Directory <i>Verzeichnispfad</i>&gt; ... &lt;/Directory&gt;</u></a>	
Umschließt eine Gruppe von Direktiven, die nur auf das genannte Verzeichnis des Dateisystems Unterverzeichnisse angewendet werden	
<a href="#"><u>DirectoryIndex <i>local-url</i> [<i>local-url</i>] ...</u></a>	index.html
List of resources to look for when the client requests a directory	
<a href="#"><u>&lt;DirectoryMatch <i>regex</i>&gt; ... &lt;/DirectoryMatch&gt;</u></a>	
Umschließt eine Gruppe von Direktiven, die auf Verzeichnisse des Dateisystems und ihre Unterverzeichnisse abgebildet werden, welche auf einen regulären Ausdruck passen	
<a href="#"><u>DirectorySlash On Off</u></a>	On
Toggle trailing slash redirects on or off	
<a href="#"><u>DocumentRoot <i>Verzeichnis</i></u></a>	/usr/local/apache/h +
Verzeichnis, welches den Haupt-Dokumentenbaum bildet, der im Web sichtbar ist.	
<a href="#"><u>DumpIOInput On Off</u></a>	Off
Dump all input data to the error log	
<a href="#"><u>DumpIOOutput On Off</u></a>	Off
Dump all output data to the error log	
<a href="#"><u>EnableExceptionHook On Off</u></a>	Off
Aktiviert einen Hook, der nach einem Absturz noch Ausnahmefehler behandeln lassen kann	
<a href="#"><u>EnableMMAP On Off</u></a>	On
Verwende Memory-Mapping, um Dateien während der Auslieferung zu lesen	
<a href="#"><u>EnableSendfile On Off</u></a>	On
Verwende die sendfile-Unterstützung des Kernels, um Dateien an den Client auszuliefern	
<a href="#"><u>ErrorDocument <i>Fehlercode</i> <i>Dokument</i></u></a>	
Das, was der Server im Fehlerfall an den Client zurückgibt	
<a href="#"><u>ErrorLog <i>Dateiname</i> <i>syslog[:facility]</i></u></a>	logs/error_log (Uni +
Ablageort, an dem der Server Fehler protokolliert	
<a href="#"><u>Example</u></a>	
Demonstration directive to illustrate the Apache module API	

<a href="#">ExpiresActive On Off</a>	
Enables generation of Expires headers	
<a href="#">ExpiresByType <i>MIME-type</i> &lt;code&gt;seconds</a>	
Value of the Expires header configured by MIME type	
<a href="#">ExpiresDefault &lt;code&gt;seconds</a>	
Default algorithm for calculating expiration time	
<a href="#">ExtendedStatus On Off</a>	Off
Keep track of extended status information for each request	
<a href="#">ExtFilterDefine <i>filtername parameters</i></a>	
Define an external filter	
<a href="#">ExtFilterOptions <i>option [option] ...</i></a>	DebugLevel=0 NoLog +
Configure <code>mod_ext_filter</code> options	
<a href="#">FileETag <i>Komponente ...</i></a>	INode MTime Size
Dateiattribute, die zur Erstellung des HTTP-Response-Headers ETag verwendet werden	
<a href="#">&lt;Files <i>Dateiname</i>&gt; ... &lt;/Files&gt;</a>	
Enthält Direktiven, die sich nur auf passende Dateinamen beziehen	
<a href="#">&lt;FilesMatch <i>regex</i>&gt; ... &lt;/FilesMatch&gt;</a>	
Enthält Direktiven, die für Dateinamen gelten, die auf einen regulären Ausdruck passen	
<a href="#">ForceLanguagePriority None Prefer Fallback [Prefer Fallback]</a>	Prefer
Action to take if a single acceptable document is not found	
<a href="#">ForceType <i>MIME-Type</i> None</a>	
Erzwingt die Auslieferung aller passenden Dateien mit dem angegebenen MIME-Content	
<a href="#">ForensicLog <i>filename pipe</i></a>	
Sets filename of the forensic log	
<a href="#">Group <i>Unix-Gruppe</i></a>	#-1
Benutzergruppe, unter welcher der Server Anfragen beantwortet	
<a href="#">Header [<i>condition</i>] set append add unset echo <i>header [value] [env=[!]variable</i>]</a>	
Configure HTTP response headers	
<a href="#">HeaderName <i>filename</i></a>	
Name of the file that will be inserted at the top of the index listing	
<a href="#">HostnameLookups On Off Double</a>	Off
Aktiviert DNS-Lookups auf Client-IP-Adressen	

<a href="#">IdentityCheck On Off</a>	Off
Ermöglicht die Protokollierung der Identität des entfernten Anwenders nach RFC1413	
<a href="#">&lt;IfDefine [!]<i>Parametername</i>&gt; ... &lt;/IfDefine&gt;</a>	
Schließt Direktiven ein, die nur ausgeführt werden, wenn eine Testbedingung beim Start wa	
<a href="#">&lt;IfModule [!]<i>Modulname</i>&gt; ... &lt;/IfModule&gt;</a>	
Schließt Direktiven ein, die abhängig vom Vorhandensein oder Fehlen eines speziellen Mod	
<a href="#">&lt;IfVersion [[!]<i>operator</i>] <i>version</i>&gt; ...</a>	
<a href="#">&lt;/IfVersion&gt;</a>	
contains version dependent configuration	
<a href="#">ImapBase <i>map referer URL</i></a>	http://servername/
Default base for imagemap files	
<a href="#">ImapDefault <i>error nocontent map referer URL</i></a>	nocontent
Default action when an imagemap is called with coordinates that are not explicitly mapped	
<a href="#">ImapMenu</a>	
<a href="#">none formatted semiformatted unformatted</a>	
Action if no coordinates are given when calling an imagemap	
<a href="#">Include <i>Dateiname Verzeichnis</i></a>	
Fügt andere Konfigurationsdateien innerhalb der Server-Konfigurationsdatei ein	
<a href="#">IndexIgnore <i>file [file] ...</i></a>	". "
Adds to the list of files to hide when listing a directory	
<a href="#">IndexOptions [+ -]<i>option</i> [[+ -]<i>option</i>] ...</a>	
Various configuration settings for directory indexing	
<a href="#">IndexOrderDefault <i>Ascending Descending</i></a>	Ascending Name
<a href="#">Name Date Size Description</a>	
Sets the default ordering of the directory index	
<a href="#">ISAPIAppendLogToErrors on off</a>	off
Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the error log	
<a href="#">ISAPIAppendLogToQuery on off</a>	on
Record HSE_APPEND_LOG_PARAMETER requests from ISAPI extensions to the query field	
<a href="#">ISAPICacheFile <i>file-path [file-path] ...</i></a>	
ISAPI .dll files to be loaded at startup	
<a href="#">ISAPIFakeAsync on off</a>	off
Fake asynchronous support for ISAPI callbacks	
<a href="#">ISAPILogNotSupported on off</a>	off

Log unsupported feature requests from ISAPI extensions	
<a href="#">ISAPIReadAheadBuffer_size</a>	49152
Size of the Read Ahead Buffer sent to ISAPI extensions	
<a href="#">KeepAlive On Off</a>	On
Aktiviert persistente HTTP-Verbindungen	
<a href="#">KeepAliveTimeout_Sekunden</a>	15
Zeitspanne, die der Server während persistenter Verbindungen auf nachfolgende Anfragen	
<a href="#">LanguagePriority MIME-lang [MIME-lang] ...</a>	
The precedence of language variants for cases where the client does not express a prefer	
<a href="#">LDAPCacheEntries_number</a>	1024
Maximum number of entries in the primary LDAP cache	
<a href="#">LDAPCacheTTL_seconds</a>	600
Time that cached items remain valid	
<a href="#">LDAPConnectionTimeout_seconds</a>	
Specifies the socket connection timeout in seconds	
<a href="#">LDAPOpCacheEntries_number</a>	1024
Number of entries used to cache LDAP compare operations	
<a href="#">LDAPOpCacheTTL_seconds</a>	600
Time that entries in the operation cache remain valid	
<a href="#">LDAPSharedCacheFile_directory-path/filename</a>	
Sets the shared memory cache file	
<a href="#">LDAPSharedCacheSize_bytes</a>	102400
Size in bytes of the shared-memory cache	
<a href="#">LDAPTrustedCA_directory-path/filename</a>	
Sets the file containing the trusted Certificate Authority certificate or database	
<a href="#">LDAPTrustedCAType_type</a>	
Specifies the type of the Certificate Authority file	
<a href="#">&lt;Limit Methode [Methode] ... &gt; ... &lt;/Limit&gt;</a>	
Beschränkt die eingeschlossenen Zugriffskontrollen auf bestimmte HTTP-Methoden	
<a href="#">&lt;LimitExcept Methode [Methode] ... &gt; ... &lt;/LimitExcept&gt;</a>	
Beschränkt Zugriffskontrollen auf alle HTTP-Methoden außer den genannten	
<a href="#">LimitInternalRecursion_Zahl [Zahl]</a>	10
Bestimmt die maximale Anzahl interner Umleitungen und verschachtelter Unteranfragen	

<a href="#"><u>LimitRequestBody Bytes</u></a>	0
Begrenzt die Gesamtgröße des vom Client gesendeten HTTP-Request-Body	
<a href="#"><u>LimitRequestFields Anzahl</u></a>	100
Begrenzt die Anzahl der HTTP-Request-Header, die vom Client entgegengenommen werden	
<a href="#"><u>LimitRequestFieldSize Bytes</u></a>	
Begrenzt die Länge des vom Client gesendeten HTTP-Request-Headers	
<a href="#"><u>LimitRequestLine Bytes</u></a>	8190
Begrenzt die Länge der vom Client entgegengenommenen HTTP-Anfragezeile	
<a href="#"><u>LimitXMLRequestBody Bytes</u></a>	1000000
Begrenzt die Größe eines XML-basierten Request-Bodys	
<a href="#"><u>Listen [IP-Adresse:]Port</u></a>	
IP-Adressen und Ports, an denen der Server lauscht	
<a href="#"><u>ListenBacklog backlog</u></a>	
Maximale Länge der Warteschlange schwebender Verbindungen	
<a href="#"><u>LoadFile filename [filename] ...</u></a>	
Link in the named object file or library	
<a href="#"><u>LoadModule module filename</u></a>	
Links in the object file or library, and adds to the list of active modules	
<a href="#"><u>&lt;Location URL-Pfad URL&gt; ... &lt;/Location&gt;</u></a>	
Wendet die enthaltenen Direktiven nur auf die entsprechenden URLs an	
<a href="#"><u>&lt;LocationMatch regex&gt; ... &lt;/LocationMatch&gt;</u></a>	
Wendet die enthaltenen Direktiven nur auf URLs an, die auf reguläre Ausdrücke passen	
<a href="#"><u>LockFile Dateiname</u></a>	logs/accept.lock
Ablageort der Lock-Datei für die Serialisierung von entgegengenommenen Anfragen	
<a href="#"><u>LogFormat format[nickname [nickname]</u></a>	"%h %l %u %t \"%r\" r" +
Describes a format for use in a log file	
<a href="#"><u>LogLevel Level</u></a>	warn
Steuert die Ausführlichkeit des Fehlerprotokolls	
<a href="#"><u>MaxClients Anzahl</u></a>	
Maximale Anzahl der Kindprozesse, die zur Bedienung von Anfragen gestartet wird	
<a href="#"><u>MaxKeepAliveRequests Anzahl</u></a>	100
Anzahl der Anfragen, die bei einer persistenten Verbindung zulässig sind	
<a href="#"><u>MaxMemFree KBytes</u></a>	0
Maximale Menge des Arbeitsspeichers, den die Haupt-Zuteilungsroutine verwalten darf, ohne aufzurufen	

<a href="#"><u>MaxRanges default   unlimited   none   number-of-ranges</u></a>	200
Number of ranges allowed before returning the complete resource	
<a href="#"><u>MaxRequestsPerChild number</u></a>	10000
Obergrenze für die Anzahl von Anfragen, die ein einzelner Kindprozess während seines Lebens	
<a href="#"><u>MaxRequestsPerThread Anzahl</u></a>	0
Die maximale Anzahl von Anfragen, die ein einzelner Thread während seiner Lebensdauer	
<a href="#"><u>MaxSpareServers Anzahl</u></a>	10
Maximale Anzahl der unbeschäftigten Kindprozesse des Servers	
<a href="#"><u>MaxSpareThreads Anzahl</u></a>	
Maximale Anzahl unbeschäftigter Threads	
<a href="#"><u>MaxThreads number</u></a>	2048
Set the maximum number of worker threads	
<a href="#"><u>MaxThreadsPerChild number</u></a>	64
Maximum number of threads per child process	
<a href="#"><u>MCacheMaxObjectCount value</u></a>	1009
The maximum number of objects allowed to be placed in the cache	
<a href="#"><u>MCacheMaxObjectSize bytes</u></a>	10000
The maximum size (in bytes) of a document allowed in the cache	
<a href="#"><u>MCacheMaxStreamingBuffer size in bytes</u></a>	the smaller of 1000 +
Maximum amount of a streamed response to buffer in memory before declaring the response	
<a href="#"><u>MCacheMinObjectSize bytes</u></a>	0
The minimum size (in bytes) of a document to be allowed in the cache	
<a href="#"><u>MCacheRemovalAlgorithm LRU GDSE</u></a>	GDSF
The algorithm used to select documents for removal from the cache	
<a href="#"><u>MCacheSize KBytes</u></a>	100
The maximum amount of memory used by the cache in KBytes	
<a href="#"><u>MetaDir directory</u></a>	.web
Name of the directory to find CERN-style meta information files	
<a href="#"><u>MetaFiles on off</u></a>	off
Activates CERN meta-file processing	
<a href="#"><u>MetaSuffix suffix</u></a>	.meta
File name suffix for the file containing CERN-style meta information	
<a href="#"><u>MimeMagicFile file-path</u></a>	
Enable MIME-type determination based on file contents using the specified magic file	

<a href="#"><u>MinSpareServers <i>Anzahl</i></u></a>	5
Minimale Anzahl der unbeschäftigten Kindprozesse des Servers	
<a href="#"><u>MinSpareThreads <i>Anzahl</i></u></a>	
Minimale Anzahl unbeschäftigter Threads, die zur Bedienung von Anfragespitzen zur Verfügung stehen	
<a href="#"><u>MMapFile <i>file-path [file-path] ...</i></u></a>	
Map a list of files into memory at startup time	
<a href="#"><u>ModMimeUsePathInfo On Off</u></a>	Off
Tells <code>mod_mime</code> to treat <code>path_info</code> components as part of the filename	
<a href="#"><u>MultiviewsMatch Any NegotiatedOnly Filters Handlers [Handlers Filters]</u></a>	NegotiatedOnly
The types of files that will be included when searching for a matching file with MultiViews	
<a href="#"><u>NameVirtualHost <i>Adresse[:Port]</i></u></a>	
Bestimmt eine IP-Adresse für den Betrieb namensbasierter virtueller Hosts	
<a href="#"><u>NoProxy <i>host [host] ...</i></u></a>	
Hosts, domains, or networks that will be connected to directly	
<a href="#"><u>NumServers <i>number</i></u></a>	2
Total number of children alive at the same time	
<a href="#"><u>NWSSLTrustedCerts <i>filename [filename] ...</i></u></a>	
List of additional client certificates	
<a href="#"><u>NWSSLUpgradeable <i>[IP-address:]portnumber</i></u></a>	
Allows a connection to be upgraded to an SSL connection upon request	
<a href="#"><u>Options [+ -]Option [[+ -]Option] ...</u></a>	All
Definiert, welche Eigenschaften oder Funktionen in einem bestimmten Verzeichnis verfügbar sind	
<a href="#"><u>Order <i>ordering</i></u></a>	Deny,Allow
Controls the default access state and the order in which <code>Allow</code> and <code>Deny</code> are evaluated.	
<a href="#"><u>PassEnv <i>env-variable [env-variable] ...</i></u></a>	
Passes environment variables from the shell	
<a href="#"><u>PidFile <i>Dateiname</i></u></a>	logs/httpd.pid
Datei, in welcher der Server die Prozess-ID des Daemons ablegt	
<a href="#"><u>ProtocolEcho On Off</u></a>	Off
Turn the echo server on or off	
<a href="#"><u>&lt;Proxy <i>wildcard-url</i>&gt; ...&lt;/Proxy&gt;</u></a>	
Container for directives applied to proxied resources	
<a href="#"><u>ProxyBadHeader IsError Ignore StartBody</u></a>	IsError

Determines how to handle bad header lines in a response	
<a href="#">ProxyBlock</a> <i>* word host domain</i> <a href="#">[word host domain] ...</a>	
Words, hosts, or domains that are banned from being proxied	
<a href="#">ProxyDomain</a> <i>Domain</i>	
Default domain name for proxied requests	
<a href="#">ProxyErrorOverride</a> On Off	Off
Override error pages for proxied content	
<a href="#">ProxyFtpDirCharset</a> <i>character set</i>	ISO-8859-1
Define the character set for proxied FTP listings	
<a href="#">ProxyIOBufferSize</a> <i>bytes</i>	8192
Determine size of internal data throughput buffer	
<a href="#">&lt;ProxyMatch regex&gt; ...&lt;/ProxyMatch&gt;</a>	
Container for directives applied to regular-expression-matched proxied resources	
<a href="#">ProxyMaxForwards</a> <i>number</i>	10
Maximum number of proxies that a request can be forwarded through	
<a href="#">ProxyPass</a> <i>[path] ! url</i>	
Maps remote servers into the local server URL-space	
<a href="#">ProxyPassReverse</a> <i>[path] url</i>	
Adjusts the URL in HTTP response headers sent from a reverse proxied server	
<a href="#">ProxyPreserveHost</a> On Off	Off
Use incoming Host HTTP request header for proxy request	
<a href="#">ProxyReceiveBufferSize</a> <i>bytes</i>	0
Network buffer size for proxied HTTP and FTP connections	
<a href="#">ProxyRemote</a> <i>match remote-server</i>	
Remote proxy used to handle certain requests	
<a href="#">ProxyRemoteMatch</a> <i>regex remote-server</i>	
Remote proxy used to handle requests matched by regular expressions	
<a href="#">ProxyRequests</a> On Off	Off
Enables forward (standard) proxy requests	
<a href="#">ProxyTimeout</a> <i>seconds</i>	300
Network timeout for proxied requests	
<a href="#">ProxyVia</a> On Off Full Block	Off
Information provided in the Via HTTP response header for proxied requests	
<a href="#">ReadmeName</a> <i>filename</i>	

Name of the file that will be inserted at the end of the index listing	
<a href="#">ReceiveBufferSize</a> <i>Bytes</i>	0
Größe des TCP-Empfangspuffers	
<a href="#">Redirect</a> [ <i>status</i> ] <i>URL-path URL</i>	
Sends an external redirect asking the client to fetch a different URL	
<a href="#">RedirectMatch</a> [ <i>status</i> ] <i>regex URL</i>	
Sends an external redirect based on a regular expression match of the current URL	
<a href="#">RedirectPermanent</a> <i>URL-path URL</i>	
Sends an external permanent redirect asking the client to fetch a different URL	
<a href="#">RedirectTemp</a> <i>URL-path URL</i>	
Sends an external temporary redirect asking the client to fetch a different URL	
<a href="#">RemoveCharset</a> <i>extension [extension] ...</i>	
Removes any character set associations for a set of file extensions	
<a href="#">RemoveEncoding</a> <i>extension [extension] ...</i>	
Removes any content encoding associations for a set of file extensions	
<a href="#">RemoveHandler</a> <i>extension [extension] ...</i>	
Removes any handler associations for a set of file extensions	
<a href="#">RemoveInputFilter</a> <i>extension [extension] ...</i>	
Removes any input filter associations for a set of file extensions	
<a href="#">RemoveLanguage</a> <i>extension [extension] ...</i>	
Removes any language associations for a set of file extensions	
<a href="#">RemoveOutputFilter</a> <i>extension [extension] ...</i>	
Removes any output filter associations for a set of file extensions	
<a href="#">RemoveType</a> <i>extension [extension] ...</i>	
Removes any content type associations for a set of file extensions	
<a href="#">RequestHeader</a> <i>set append add unset header [value [env=[!]variable]]</i>	
Configure HTTP request headers	
<a href="#">Require</a> <i>Name [Name] ...</i>	
Wählt die authentisierten Benutzer aus, die auf eine Ressource zugreifen können	
<a href="#">RewriteBase</a> <i>URL-path</i>	
Sets the base URL for per-directory rewrites	
<a href="#">RewriteCond</a> <i>TestString CondPattern</i>	
Defines a condition under which rewriting will take place	
<a href="#">RewriteEngine</a> <i>on off</i>	off

<a href="#">RewriteEngine</a>	Enables or disables runtime rewriting engine	
<a href="#">RewriteLock</a> <i>file-path</i>	Sets the name of the lock file used for <a href="#">RewriteMap</a> synchronization	
<a href="#">RewriteLog</a> <i>file-path</i>	Sets the name of the file used for logging rewrite engine processing	
<a href="#">RewriteLogLevel</a> <i>Level</i>	Sets the verbosity of the log file used by the rewrite engine	0
<a href="#">RewriteMap</a> <i>MapName MapType:MapSource</i>	Defines a mapping function for key-lookup	
<a href="#">RewriteOptions</a> <i>Options</i>	Sets some special options for the rewrite engine	MaxRedirects=10
<a href="#">RewriteRule</a> <i>Pattern Substitution</i>	Defines rules for the rewriting engine	
<a href="#">RLimitCPU</a> <i>Sekunden max [Sekunden max]</i>	Begrenzt den CPU-Verbrauch von Prozessen, die von Apache-Kindprozessen gestartet wurden	
<a href="#">RLimitMEM</a> <i>Bytes max [Bytes max]</i>	Begrenzt den Speicherverbrauch von Prozessen, die von Apache-Kindprozessen gestartet wurden	
<a href="#">RLimitNPROC</a> <i>Zahl max [Zahl max]</i>	Begrenzt die Anzahl der Prozesse, die von Prozessen gestartet werden können, die ihrerseits Apache-Kindprozesse gestartet wurden	
<a href="#">Satisfy</a> <i>Any All</i>	Zusammenspiel von rechnerbasierter Zugriffskontrolle und Benutzerauthentifizierung	All
<a href="#">ScoreBoardFile</a> <i>Dateipfad</i>	Ablageort der Datei, die zur Speicherung von Daten zur Koordinierung der Kindprozesse verwendet wird	logs/apache_status
<a href="#">Script</a> <i>method cgi-script</i>	Activates a CGI script for a particular request method.	
<a href="#">ScriptAlias</a> <i>URL-path file-path directory-path</i>	Maps a URL to a filesystem location and designates the target as a CGI script	
<a href="#">ScriptAliasMatch</a> <i>regex file-path directory-path</i>	Maps a URL to a filesystem location using a regular expression and designates the target as a CGI script	
<a href="#">ScriptInterpreterSource</a> <i>Registry Registry-Strict Script</i>	Methode zur Ermittlung des Interpreters von CGI-Skripten	Script
<a href="#">ScriptLog</a> <i>file-path</i>	Location of the CGI script error logfile	

<a href="#">ScriptLogBuffer <i>bytes</i></a>	1024
Maximum amount of PUT or POST requests that will be recorded in the scriptlog	
<a href="#">ScriptLogLength <i>bytes</i></a>	10385760
Size limit of the CGI script logfile	
<a href="#">ScriptSock <i>file-path</i></a>	logs/cgisock
The name of the socket to use for communication with the cgi daemon	
<a href="#">SecureListen [<i>IP-address:</i>]<i>portnumber</i></a> <a href="#">Certificate-Name [MUTUAL]</a>	
Enables SSL encryption for the specified port	
<a href="#">SendBufferSize <i>Bytes</i></a>	0
Größe des TCP-Sendepuffers	
<a href="#">ServerAdmin <i>E-Mail-Adresse</i></a>	
E-Mail-Adresse, die der Server in Fehlermeldungen einfügt, welche an den Client gesendet	
<a href="#">ServerAlias <i>Hostname [Hostname] ...</i></a>	
Alternativer Name für einen Host, der verwendet wird, wenn Anfragen einem namensbasier Host zugeordnet werden	
<a href="#">ServerLimit <i>Anzahl</i></a>	
Obergrenze für die konfigurierbare Anzahl von Prozessen	
<a href="#">ServerName <i>voll-qualifizierter-Domainname[:port]</i></a>	
Rechnername und Port, die der Server dazu verwendet, sich selbst zu identifizieren	
<a href="#">ServerPath <i>URL-Pfad</i></a>	
Veralteter URL-Pfad für einen namensbasierten virtuellen Host, auf den von einem inkompeten Client zugegriffen wird	
<a href="#">ServerRoot <i>Verzeichnis</i></a>	/usr/local/apache
Basisverzeichnis der Serverinstallation	
<a href="#">ServerSignature <i>On Off EMail</i></a>	Off
Konfiguriert die Fußzeile von servergenerierten Dokumenten	
<a href="#">ServerTokens</a>	Full
<a href="#">Major Minor Min[imal]  Prod[uctOnly]  OS Full</a>	
Konfiguriert den HTTP-Response-Header Server	
<a href="#">SetEnv <i>env-variable value</i></a>	
Sets environment variables	
<a href="#">SetEnvIf <i>attribute regex [!]<i>env-variable</i>[=<i>value</i>]</i></a> <a href="#">[[!]<i>env-variable</i>[=<i>value</i>]] ...</a>	

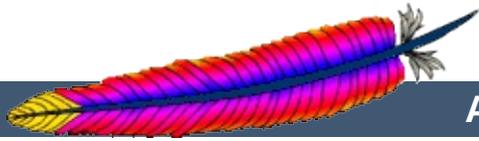
<a href="#">SetEnvIfNoCase</a> <i>attribute regex [!]env-variable[=value] [[!]env-variable[=value]] ...</i>	
Sets environment variables based on attributes of the request without respect to case	
<a href="#">SetHandler</a> <i>Handlername None</i>	
Erzwingt die Verarbeitung aller passenden Dateien durch einen Handler	
<a href="#">SetInputFilter</a> <i>Filter[:Filter...]</i>	
Bestimmt die Filter, die Client-Anfragen und POST-Eingaben verarbeiten	
<a href="#">SetOutputFilter</a> <i>Filter[:Filter...]</i>	
Bestimmt die Filter, die Antworten des Servers verarbeiten	
<a href="#">SSIEndTag</a> <i>tag</i>	"-->"
String that ends an include element	
<a href="#">SSIErrorMsg</a> <i>message</i>	"[an error occurred +
Error message displayed when there is an SSI error	
<a href="#">SSIStartTag</a> <i>tag</i>	"<!--#"
String that starts an include element	
<a href="#">SSITimeFormat</a> <i>formatstring</i>	"%A, %d-%b-%Y %H:%M +
Configures the format in which date strings are displayed	
<a href="#">SSIUndefinedEcho</a> <i>string</i>	"(none)"
String displayed when an unset variable is echoed	
<a href="#">SSLCACertificateFile</a> <i>file-path</i>	
File of concatenated PEM-encoded CA Certificates for Client Auth	
<a href="#">SSLCACertificatePath</a> <i>directory-path</i>	
Directory of PEM-encoded CA Certificates for Client Auth	
<a href="#">SSLCARevocationFile</a> <i>file-path</i>	
File of concatenated PEM-encoded CA CRLs for Client Auth	
<a href="#">SSLCARevocationPath</a> <i>directory-path</i>	
Directory of PEM-encoded CA CRLs for Client Auth	
<a href="#">SSLCertificateChainFile</a> <i>file-path</i>	
File of PEM-encoded Server CA Certificates	
<a href="#">SSLCertificateFile</a> <i>file-path</i>	
Server PEM-encoded X.509 Certificate file	
<a href="#">SSLCertificateKeyFile</a> <i>file-path</i>	
Server PEM-encoded Private Key file	

<a href="#">SSLCipherSuite <i>cipher-spec</i></a>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL handshake	
<a href="#">SSLEngine on off</a>	off
SSL Engine Operation Switch	
<a href="#">SSLHonorCipherOrder <i>flag</i></a>	
Option to prefer the server's cipher preference order	
<a href="#">SSLInsecureRenegotiation <i>flag</i></a>	off
Option to enable support for insecure renegotiation	
<a href="#">SSLMutex <i>type</i></a>	none
Semaphore for internal mutual exclusion of operations	
<a href="#">SSLOptions [+ -]<i>option ...</i></a>	
Configure various SSL engine run-time options	
<a href="#">SSLPassPhraseDialog <i>type</i></a>	builtin
Type of pass phrase dialog for encrypted private keys	
<a href="#">SSLProtocol [+ -]<i>protocol ...</i></a>	all
Configure usable SSL protocol flavors	
<a href="#">SSLProxyCACertificateFile <i>file-path</i></a>	
File of concatenated PEM-encoded CA Certificates for Remote Server Auth	
<a href="#">SSLProxyCACertificatePath <i>directory-path</i></a>	
Directory of PEM-encoded CA Certificates for Remote Server Auth	
<a href="#">SSLProxyCARevocationFile <i>file-path</i></a>	
File of concatenated PEM-encoded CA CRLs for Remote Server Auth	
<a href="#">SSLProxyCARevocationPath <i>directory-path</i></a>	
Directory of PEM-encoded CA CRLs for Remote Server Auth	
<a href="#">SSLProxyCipherSuite <i>cipher-spec</i></a>	ALL:!ADH:RC4+RSA: +
Cipher Suite available for negotiation in SSL proxy handshake	
<a href="#">SSLProxyEngine on off</a>	off
SSL Proxy Engine Operation Switch	
<a href="#">SSLProxyMachineCertificateFile <i>filename</i></a>	
File of concatenated PEM-encoded client certificates and keys to be used by the proxy	
<a href="#">SSLProxyMachineCertificatePath <i>directory</i></a>	
Directory of PEM-encoded client certificates and keys to be used by the proxy	
<a href="#">SSLProxyProtocol [+ -]<i>protocol ...</i></a>	all

Configure usable SSL protocol flavors for proxy usage	
<a href="#"><u>SSLProxyVerify level</u></a>	none
Type of remote server Certificate verification	
<a href="#"><u>SSLProxyVerifyDepth number</u></a>	1
Maximum depth of CA Certificates in Remote Server Certificate verification	
<a href="#"><u>SSLRandomSeed context source [bytes]</u></a>	
Pseudo Random Number Generator (PRNG) seeding source	
<a href="#"><u>SSLRequire expression</u></a>	
Allow access only when an arbitrarily complex boolean expression is true	
<a href="#"><u>SSLRequireSSL</u></a>	
Deny access when SSL is not used for the HTTP request	
<a href="#"><u>SSLSessionCache type</u></a>	none
Type of the global/inter-process SSL Session Cache	
<a href="#"><u>SSLSessionCacheTimeout seconds</u></a>	300
Number of seconds before an SSL session expires in the Session Cache	
<a href="#"><u>SSLUserName varname</u></a>	
Variable name to determine user name	
<a href="#"><u>SSLVerifyClient level</u></a>	none
Type of Client Certificate verification	
<a href="#"><u>SSLVerifyDepth number</u></a>	1
Maximum depth of CA Certificates in Client Certificate verification	
<a href="#"><u>StartServers Anzahl</u></a>	
Anzahl der Kindprozesse des Servers, die beim Start erstellt werden	
<a href="#"><u>StartThreads Anzahl</u></a>	
Anzahl der Threads, die beim Start erstellt werden	
<a href="#"><u>SuexecUserGroup User Group</u></a>	
User and group for CGI programs to run as	
<a href="#"><u>ThreadLimit Anzahl</u></a>	
Bestimmt die Obergrenze der konfigurierbaren Anzahl von Threads pro Kindprozess	
<a href="#"><u>ThreadsPerChild Anzahl</u></a>	
Anzahl der Threads, die mit jedem Kindprozess gestartet werden	
<a href="#"><u>ThreadStackSize number</u></a>	65536
Determine the stack size for each thread	
<a href="#"><u>TimeOut Sekunden</u></a>	300
Zeitspanne, die der Server auf verschiedene Ereignisse wartet, bevor er die Anfrage abbricht	

<a href="#">TraceEnable</a> <i>[on off extended]</i>	on
Legt das Serververhalten bei TRACE-Anfragen fest	
<a href="#">TransferLog</a> <i>file pipe</i>	
Specify location of a log file	
<a href="#">TypesConfig</a> <i>file-path</i>	conf/mime.types
The location of the mime.types file	
<a href="#">UnsetEnv</a> <i>env-variable [env-variable] ...</i>	
Removes variables from the environment	
<a href="#">UseCanonicalName</a> <i>On Off DNS</i>	On
Bestimmt, wie der Server seinen eigenen Namen und Port ermittelt	
<a href="#">User</a> <i>Unix-User-ID</i>	#-1
Die Benutzerkennung, unter welcher der Server Anfragen beantwortet	
<a href="#">UserDir</a> <i>directory-filename</i>	public_html
Location of the user-specific directories	
<a href="#">VirtualDocumentRoot</a> <i>interpolated-directory none</i>	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">VirtualDocumentRootIP</a> <i>interpolated-directory none</i>	none
Dynamically configure the location of the document root for a given virtual host	
<a href="#">&lt;VirtualHost Adresse[:Port] [Adresse[:Port]] ...&gt; ... &lt;/VirtualHost&gt;</a>	
Enthält Direktiven, die nur auf bestimmte Hostnamen oder IP-Adressen angewendet werden	
<a href="#">VirtualScriptAlias</a> <i>interpolated-directory none</i>	none
Dynamically configure the location of the CGI directory for a given virtual host	
<a href="#">VirtualScriptAliasIP</a> <i>interpolated-directory none</i>	none
Dynamically configure the location of the cgi directory for a given virtual host	
<a href="#">Win32DisableAcceptEx</a>	
Für die Annahme von Netzwerkverbindungen wird accept() anstelle von AcceptEx() verwenden	
<a href="#">XBitHack</a> <i>on off full</i>	off
Parse SSI directives in files with the execute bit set	





[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

## Modul-Index

Unten ist eine Liste aller Module angegeben, die als Bestandteil der Apache-Distribution mitgeliefert werden. Bitte beachten Sie auch die vollständige alphabetische Liste [aller Apache-Direktiven](#).

### Siehe auch

[Multi-Processing-Module \(MPMs\)](#)

[Kurzreferenz der Direktiven](#)



## core

Ständig verfügbare Kernfunktionen des Apache HTTP Servers

## mpm\_common

Eine Sammlung von Direktiven, die in mehr als einem Multi-Processing-Modul (MPM) implementiert sind.

## beos

Dieses Multi-Processing-Modul ist für BeOS optimiert.

## leader

Eine experimentelle Variante des Standard-MPMs worker

## mpm\_netware

Multi-Processing Module implementing an exclusively threaded web server optimized for Novell NetWare

## mpmt\_os2

Hybrid multi-process, multi-threaded MPM for OS/2

## perchild

Multi-Processing Module allowing for daemon processes serving requests to be assigned a variety of different userids

## prefork

Implementiert einen im Voraus forkenden Webserver ohne Thread-Unterstützung

## threadpool

Yet another experimental variant of the standard worker MPM

## mpm\_winnt

Das Multi-Processing-Modul ist optimiert für Windows NT.

## worker

Multi-Processing-Modul, das einen Hybrid-Webserver mit

Multi-Thread und Multi-Prozess-Unterstützung implementiert



A | C | D | E | F | H | I | L | M | N | P | R | S | U |  
V

### [mod\\_access](#)

Provides access control based on client hostname, IP address, or other characteristics of the client request.

### [mod\\_actions](#)

This module provides for executing CGI scripts based on media type or request method.

### [mod\\_alias](#)

Provides for mapping different parts of the host filesystem in the document tree and for URL redirection

### [mod\\_asis](#)

Sends files that contain their own HTTP headers

### [mod\\_auth](#)

User authentication using text files

### [mod\\_auth\\_anon](#)

Allows "anonymous" user access to authenticated areas

### [mod\\_auth\\_dbm](#)

Provides for user authentication using DBM files

### [mod\\_auth\\_digest](#)

User authentication using MD5 Digest Authentication.

### [mod\\_auth\\_ldap](#)

Allows an LDAP directory to be used to store the database for HTTP Basic authentication.

### [mod\\_autoindex](#)

Generates directory indexes, automatically, similar to the Unix ls command or the Win32 dir shell command

### [mod\\_cache](#)

Content cache keyed to URIs.

### [mod\\_cern\\_meta](#)

CERN httpd metafile semantics

### [mod\\_cgi](#)

Execution of CGI scripts

### [mod\\_cgid](#)

Execution of CGI scripts using an external CGI daemon

### [mod\\_charset\\_lite](#)

Specify character set translation or recoding

### [mod\\_dav](#)

Distributed Authoring and Versioning ([WebDAV](#)) functionality

### [mod\\_dav\\_fs](#)

filesystem provider for [mod\\_dav](#)

### [mod\\_deflate](#)

Compress content before it is delivered to the client

### [mod\\_dir](#)

Provides for "trailing slash" redirects and serving directory index files

### [mod\\_disk\\_cache](#)

Content cache storage manager keyed to URIs

### [mod\\_dumpio](#)

Dumps all I/O to error log as desired.

### [mod\\_echo](#)

A simple echo server to illustrate protocol modules

### [mod\\_env](#)

Modifies the environment which is passed to CGI scripts and SSI pages

### [mod\\_example](#)

Illustrates the Apache module API

### [mod expires](#)

Generation of Expires and Cache-Control HTTP headers according to user-specified criteria

### [mod\\_ext\\_filter](#)

Pass the response body through an external program before delivery to the client

### [mod\\_file\\_cache](#)

Caches a static list of files in memory

### [mod\\_headers](#)

Customization of HTTP request and response headers

### [mod\\_imap](#)

Server-side imagemap processing

### [mod\\_include](#)

Server-parsed html documents (Server Side Includes)

### [mod\\_info](#)

Provides a comprehensive overview of the server configuration

### [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](#)

Logging of the requests made to the server

### [mod\\_log\\_forensic](#)

Forensic Logging of the requests made to the server

### [mod\\_logio](#)

Logging of input and output bytes per request

### [mod\\_mem\\_cache](#)

Content cache keyed to URIs

### [mod\\_mime](#)

Associates the requested filename's extensions with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding)

### [mod\\_mime\\_magic](#)

Determines the MIME type of a file by looking at a few bytes of its contents

### [mod\\_negotiation](#)

Provides for [content negotiation](#)

### [mod\\_nw\\_ssl](#)

Enable SSL encryption for NetWare

### [mod\\_proxy](#)

HTTP/1.1 proxy/gateway server

### [mod\\_proxy\\_connect](#)

[mod\\_proxy](#) extension for CONNECT request handling

### [mod\\_proxy\\_ftp](#)

FTP support module for [mod\\_proxy](#)

### [mod\\_proxy\\_http](#)

HTTP support module for [mod\\_proxy](#)

### [mod\\_rewrite](#)

Provides a rule-based rewriting engine to rewrite requested URLs on the fly

### [mod\\_setenvif](#)

Allows the setting of environment variables based on characteristics of the request

### [mod\\_so](#)

Loading of executable code and modules into the server at start-up or restart time

### [mod\\_speling](#)

Attempts to correct mistaken URLs that users might have entered by ignoring capitalization and by allowing up to one misspelling

### [mod\\_ssl](#)

Strong cryptography using the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols

### [mod\\_status](#)

Provides information on server activity and performance

### [mod\\_suexec](#)

Allows CGI scripts to run as a specified user and Group

### [mod\\_unique\\_id](#)

Provides an environment variable with a unique identifier for each request

### [mod\\_userdir](#)

User-specific directories

### [mod\\_usertrack](#)

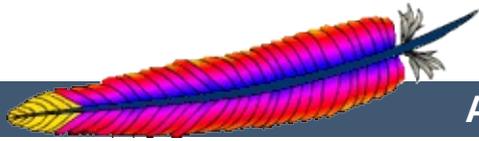
*Clickstream* logging of user activity on a site

### [mod\\_version](#)

Version dependent configuration

### [mod\\_vhost\\_alias](#)

Provides for dynamically configured mass virtual hosting



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Frequently Asked Questions

The latest version of this FAQ is always available from the main Apache web site, at <<http://httpd.apache.org/docs/2.0/faq/>>. In addition, you can view this FAQ [all in one page](#) for easy searching and printing.

Since Apache 2.0 is quite new, we don't yet know what the *Frequently Asked Questions* will be. While this section fills up, you should also consult the [Apache 1.3 FAQ](#) to see if your question is answered there.



## [Support](#)

What do I do when I have problems?

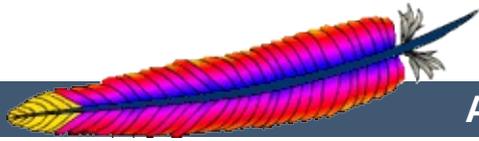
## [Error Messages](#)

What does this error message mean?

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

# Seitenindex

Diese Seite verzeichnet die zur Zeit verfügbaren Dokumente der [Dokumentation zum Apache HTTP Server Version 2.0.](#)



- 
- [Upgrade von 1.3 auf 2.0](#)
  - [Neue Funktionen in Version 2.0](#)
  - [Apache-Lizenz](#)



- [Kompilieren und Installieren](#)
- [Apache starten](#)
- [Beenden und Neustarten des Servers](#)
- [Konfigurationsdateien](#)
- [Konfigurationsabschnitte](#)
- [Serverweite Konfiguration](#)
- [Log-Dateien](#)
- [URLs auf das Dateisystem abbilden](#)
- [Tipps zur Sicherheit](#)
- [Dynamic Shared Object \(DSO\)](#)
- [Content Negotiation](#)
- [Individuelle Fehlermeldungen](#)
- [Bestimmen der vom Apache verwendeten Adressen und Ports](#)
- [Multi-Processing-Module \(MPMs\)](#)
- [Umgebungsvariablen](#)
- [Handler](#)
- [Filter](#)
- [suEXEC Unterstützung](#)
- [Performance-Hinweise](#)
- [Einführung in die URL-Manipulation](#)



- 
- [Übersicht](#)
  - [Namensbasierte virtuelle Hosts](#)
  - [IP-basierte virtuelle Hosts](#)
  - [Dynamisch konfiguriertes Massen-Virtual-Hosting](#)
  - [Beispiele für virtuelles Hosts in typischen Installationen](#)
  - [Tiefergehende Erörterung der Zuweisung virtueller Hosts](#)
  - [Datei-Deskriptor-Begrenzungen](#)
  - [Probleme bezüglich DNS und Apache](#)



## Training gesteuerte Fragen (TFQ)

- [Übersicht](#)
- [Support](#)
- [Fehlermeldungen](#)



## SSL/TLS-Verschlüsselung des Applikations

- [Übersicht](#)
- [SSL/TLS-Verschlüsselung: Einführung](#)
- [SSL/TLS-Verschlüsselung: Kompatibilität](#)
- [SSL/TLS-Verschlüsselung: Praxis](#)
- [SSL/TLS-Verschlüsselung: FAQ](#)



- [Übersicht](#)
- [Authentisierung, Autorisierung und Zugriffskontrolle](#)
- [Dynamische Inhalte mit CGI](#)
- [Einführung in Server Side Includes](#)
- [.htaccess-Dateien](#)
- [Web-Verzeichnisse für Benutzer](#)
- [Apache-Tutorials](#)



- [Übersicht](#)
- [Apache unter Microsoft Windows einsetzen](#)
- [Kompilieren des Apache für Microsoft Windows](#)
- [Apache unter Novell NetWare einsetzen](#)
- [Einen Hochleistungs-Web-Server auf HPUX betreiben](#)
- [Die Apache EBCDIC-Portierung](#)



## Apache HTTP Server and its programs

- [Übersicht](#)
- [httpd](#)
- [ab](#)
- [apachectl](#)
- [apxs](#)
- [configure](#)
- [dbmmanage](#)
- [htdbm](#)
- [htdigest](#)
- [htpasswd](#)
- [logresolve](#)
- [rotatelogs](#)
- [suexec](#)
- [Sonstige Programme](#)



- 
- [Übersicht](#)
  - [International angepaßte Server-Fehlermeldungen](#)
  - [Verbindungen im FIN\\_WAIT\\_2 und Apache](#)
  - [Bekannte Client-Probleme](#)
  - [Deskriptoren und Apache](#)
  - [Wichtige Standards](#)



- [Erklärung der Fachbegriffe zu Apache-Modulen](#)
- [Erklärung der Fachbegriffe zu Apache-Direktiven](#)
  
- [Apache-Kernfunktionen](#)
- [Allgemeine Direktiven der Apache-MPMs](#)
- [Apache-MPM beos](#)
- [Apache-MPM leader](#)
- [Apache-MPM network](#)
- [Apache-MPM os2](#)
- [Apache-MPM perchild](#)
- [Apache-MPM prefork](#)
- [Apache-MPM threadpool](#)
- [Apache-MPM winnt](#)
- [Apache-MPM worker](#)
  
- [Apache-Modul mod\\_access](#)
- [Apache-Modul mod\\_actions](#)
- [Apache-Modul mod\\_alias](#)
- [Apache-Modul mod\\_asis](#)
- [Apache-Modul mod\\_auth](#)
- [Apache-Modul mod\\_auth\\_anon](#)
- [Apache-Modul mod\\_auth\\_dbm](#)
- [Apache-Modul mod\\_auth\\_digest](#)
- [Apache-Modul mod\\_auth\\_ldap](#)
- [Apache-Modul mod\\_autoindex](#)
- [Apache-Modul mod\\_cache](#)
- [Apache-Modul mod\\_cern\\_meta](#)
- [Apache-Modul mod\\_cgi](#)
- [Apache-Modul mod\\_cgid](#)
- [Apache-Modul mod\\_charset\\_lite](#)
- [Apache-Modul mod\\_dav](#)
- [Apache-Modul mod\\_dav\\_fs](#)
- [Apache-Modul mod\\_deflate](#)

- [Apache-Modul mod\\_dir](#)
- [Apache-Modul mod\\_disk\\_cache](#)
- [Apache-Modul mod\\_dumpio](#)
- [Apache-Modul mod\\_echo](#)
- [Apache-Modul mod\\_env](#)
- [Apache-Modul mod\\_example](#)
- [Apache-Modul mod\\_expires](#)
- [Apache-Modul mod\\_ext\\_filter](#)
- [Apache-Modul mod\\_file\\_cache](#)
- [Apache-Modul mod\\_headers](#)
- [Apache-Modul mod\\_imap](#)
- [Apache-Modul mod\\_include](#)
- [Apache-Modul mod\\_info](#)
- [Apache-Modul mod\\_isapi](#)
- [Apache-Modul mod\\_ldap](#)
- [Apache-Modul mod\\_log\\_config](#)
- [Apache-Modul mod\\_log\\_forensic](#)
- [Apache-Modul mod\\_logio](#)
- [Apache-Modul mod\\_mem\\_cache](#)
- [Apache-Modul mod\\_mime](#)
- [Apache-Modul mod\\_mime\\_magic](#)
- [Apache-Modul mod\\_negotiation](#)
- [Apache-Modul mod\\_nw\\_ssl](#)
- [Apache-Modul mod\\_proxy](#)
- [Apache-Modul mod\\_proxy\\_connect](#)
- [Apache-Modul mod\\_proxy\\_ftp](#)
- [Apache-Modul mod\\_proxy\\_http](#)
- [Apache-Modul mod\\_rewrite](#)
- [Apache-Modul mod\\_setenvif](#)
- [Apache-Modul mod\\_so](#)
- [Apache-Modul mod\\_speling](#)
- [Apache-Modul mod\\_ssl](#)
- [Apache-Modul mod\\_status](#)
- [Apache-Modul mod\\_suexec](#)

- [Apache-Modul mod\\_unique\\_id](#)
- [Apache-Modul mod\\_userdir](#)
- [Apache-Modul mod\\_usertrack](#)
- [Apache-Modul mod\\_version](#)
- [Apache-Modul mod\\_vhost\\_alias](#)



- 
- [Übersicht](#)
  - [Anmerkungen zur Apache-API](#)
  - [Debuggen der Speicher-Belegung in der APR](#)
  - [Apache 2.0 dokumentieren](#)
  - [Hook-Funktionen des Apache 2.0](#)
  - [Module von Apache 1.3 nach Apache 2.0 konvertieren](#)
  - [Verarbeitung der Anfragen im Apache 2.0](#)
  - [Wie Filter im Apache 2.0 arbeiten](#)

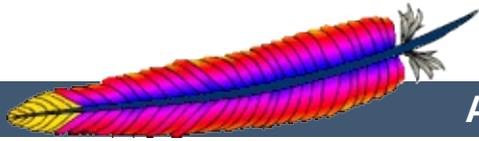


- [Glossar](#)
- [Modul-Index](#)
- [Direktiven-Index](#)
- [Kurzreferenz der Direktiven](#)

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Server and Supporting Programs

This page documents all the executable programs included with the Apache HTTP Server.



## [httpd](#)

Apache hypertext transfer protocol server

## [apachectl](#)

Apache HTTP server control interface

## [ab](#)

Apache HTTP server benchmarking tool

## [apxs](#)

APache eXtenSion tool

## [configure](#)

Configure the source tree

## [dbmmanage](#)

Create and update user authentication files in DBM format for basic authentication

## [htdigest](#)

Create and update user authentication files for digest authentication

## [htdbm](#)

Manipulate DBM password databases.

## [htpasswd](#)

Create and update user authentication files for basic authentication

## [logresolve](#)

Resolve hostnames for IP-addresses in Apache logfiles

## [rotatelogs](#)

Rotate Apache logs without having to kill the server

## [suexec](#)

Switch User For Exec

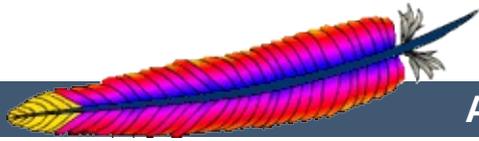
## [Other Programs](#)

Support tools with no own manual page.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

# Apache SSL/TLS Encryption

The Apache HTTP Server module [mod\\_ssl](#) provides an interface to the [OpenSSL](#) library, which provides Strong Encryption using the Secure Sockets Layer and Transport Layer Security protocols. The module and this documentation are based on Ralf S. Engelschall's [mod\\_ssl](#) project.



- 
- [Introduction](#)
  - [Compatibility](#)
  - [How-To](#)
  - [Frequently Asked Questions](#)
  - [Glossary](#)

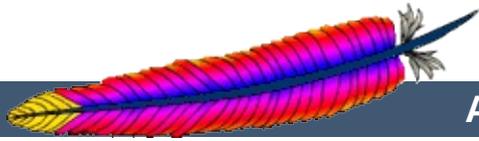


Extensive documentation on the directives and environment variables provided by this module is provided in the [mod\\_ssl reference documentation](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP-Server](#) > [Dokumentation](#) > [Version 2.0](#)

## Apache-Dokumentation zu virtuellen Hosts

Der Begriff *virtueller Host* (*Anm.d.Ü.*: engl. 'virtual host') bezieht sich auf die Praxis, mehr als ein Webangebot (z.B. `www.company1.com` und `www.company2.com`) auf einer einzigen Maschine zu betreiben. Virtuelle Hosts können "[IP-basiert](#)" sein, was bedeutet, dass jedes Webangebot eine andere IP besitzt, oder "[Namens-basiert](#)", was bedeutet, dass unter jeder IP-Adresse mehrere Namen laufen. Die Tatsache, dass sie auf dem gleichen physischen Server laufen, ist für den Endbenutzer nicht offensichtlich.

Der Apache war einer der ersten Server, der IP-basierte virtuelle Hosts von Haus aus direkt unterstützt hat. Seit Version 1.1 unterstützt der Apache sowohl IP-basierte als auch namensbasierte virtuelle Hosts (vhosts). Letzteres wird zuweilen auch *Host-basiert* oder *non-IP-Virtual-Host* genannt.

Nachfolgend finden Sie eine Liste von Dokumenten, die alle Details der Unterstützung von virtuellen Hosts ab Apache Version 1.3 beschreiben.

### Siehe auch

[mod\\_vhost\\_alias](#)

[Namensbasierte virtuelle Hosts](#)

[IP-basierte virtuelle Hosts](#)

[Beispiele für virtuelle Hosts](#)

[Datei-Deskriptor-Begrenzungen](#)

[Massen-Virtual-Hosting](#)

[Zuweisung virtueller Hosts](#)



- [Namensbasierte virtuelle Hosts](#) (Mehr als ein Webangebot pro IP-Adresse)
- [IP-basierte virtuelle Hosts](#) (Eine IP-Adresse für jedes Webangebot)
- [Beispiele für virtuelles Hosts in typischen Installationen](#)
- [Datei-Deskriptor-Begrenzungen](#) (oder *Zu viele Protokolldateien*)
- [Dynamisch konfiguriertes Massen-Virtual-Hosting](#)
- [Tiefere Erörterung der Zuweisung virtueller Hosts](#)



- [<VirtualHost>](#)
- [NameVirtualHost](#)
- [ServerName](#)
- [ServerAlias](#)
- [ServerPath](#)

Bei der Suche von Fehlern in Ihrer Virtual-Host-Konfiguration ist die Apache-Befehlszeilenoption `-S` möglicherweise hilfreich. Geben Sie dazu den folgenden Befehl ein:

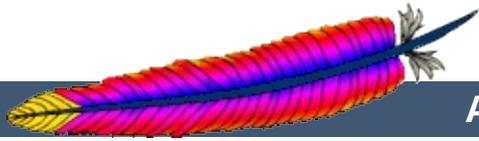
```
/usr/local/apache2/bin/httpd -S
```

Diese Anweisung gibt eine Beschreibung aus, wie der Apache die Konfigurationsdatei analysiert hat. Eine sorgfältige Überprüfung der IP-Adressen und Servernamen kann helfen, Konfigurationsfehler aufzudecken. (Lesen Sie die Dokumentation zum [httpd](#)-Programm für weitere Befehlszeilenoptionen.)

---

Copyright 2013 The Apache Software Foundation.  
Lizenziert unter der [Apache License, Version 2.0](#).

[Module](#) | [Direktiven](#) | [FAQ](#) | [Glossar](#) | [Seitenindex](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Developer Documentation for Apache 2.0

Many of the documents on these Developer pages are lifted from Apache 1.3's documentation. While they are all being updated to Apache 2.0, they are in different stages of progress. Please be patient, and point out any discrepancies or errors on the developer/ pages directly to the [dev@httpd.apache.org](mailto:dev@httpd.apache.org) mailing list.



- [Apache 1.3 API Notes](#)
- [Apache 2.0 Hook Functions](#)
- [Request Processing in Apache 2.0](#)
- [How filters work in Apache 2.0](#)
- [Converting Modules from Apache 1.3 to Apache 2.0](#)
- [Debugging Memory Allocation in APR](#)
- [Documenting Apache 2.0](#)
- [Apache 2.0 Thread Safety Issues](#)

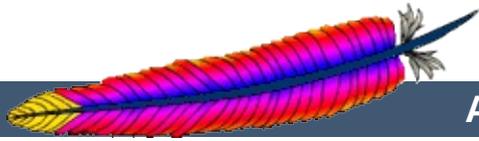


- 
- Tools provided by Ian Holsman:
    - [Apache 2 cross reference](#)
    - [Autogenerated Apache 2 code documentation](#)
  - Module Development Tutorials by Kevin O'Donnell
    - [Integrating a module into the Apache build system](#)
    - [Handling configuration directives](#)
  - [Some notes on Apache module development by Ryan Bloom](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

# Apache Miscellaneous Documentation

Below is a list of additional documentation pages that apply to the Apache web server development project.

## Warning

Some of the documents below have not been fully updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

## [How to use XSSI and Negotiation for custom ErrorDocuments](#)

Describes a solution which uses XSSI and negotiation to custom-tailor the Apache ErrorDocuments to taste, adding the advantage of returning internationalized versions of the error messages depending on the client's language preferences.

## [File Descriptor use in Apache](#)

Describes how Apache uses file descriptors and talks about various limits imposed on the number of descriptors available by various operating systems.

## [FIN\\_WAIT\\_2](#)

A description of the causes of Apache processes going into the FIN\_WAIT\_2 state, and what you can do about it.

## [Known Client Problems](#)

A list of problems in HTTP clients which can be mitigated by Apache.

## [Performance Notes - Apache Tuning](#)

Notes about how to (run-time and compile-time) configure Apache for highest performance. Notes explaining why Apache does some things, and why it doesn't do other things (which make it slower/faster).

## [Security Tips](#)

Some "do"s - and "don't"s - for keeping your Apache web site secure.

## [URL Rewriting Guide](#)

This document supplements the [mod\\_rewrite reference documentation](#). It describes how one can use Apache's [mod\\_rewrite](#) to solve typical URL-based problems webmasters are usually confronted with in practice.

## [Apache Tutorials](#)

A list of external resources which help to accomplish common tasks with the Apache HTTP server.

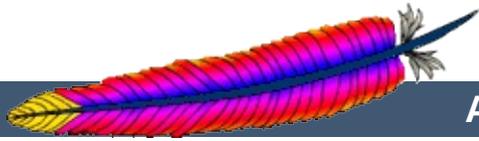
## [Relevant Standards](#)

This document acts as a reference page for most of the relevant standards that Apache follows.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Platform Specific Notes



---

## Using Apache

This document explains how to install, configure and run Apache 2.0 under Microsoft Windows.

See: [Using Apache with Microsoft Windows](#)

## Compiling Apache

There are many important points before you begin compiling Apache. This document explain them.

See: [Compiling Apache for Microsoft Windows](#)



## Novell NetWare

This document explains how to install, configure and run Apache 2.0 under Novell NetWare 5.1 and above.

See: [Using Apache With Novell NetWare](#)

## EBCDIC

Version 1.3 of the Apache HTTP Server is the first version which includes a port to a (non-ASCII) mainframe machine which uses the EBCDIC character set as its native codeset.

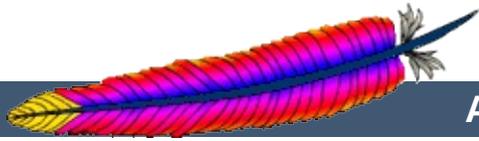
**Warning:** This document has not been updated to take into account changes made in the 2.0 version of the Apache HTTP Server. Some of the information may still be relevant, but please use it with care.

See: [The Apache EBCDIC Port](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Programs](#)

## **suexec - Switch user before executing external programs**

suexec is used by the Apache HTTP Server to switch to another user before executing CGI programs. In order to achieve this, it must run as root. Since the HTTP daemon normally doesn't run as root, the suexec executable needs the setuid bit set and must be owned by root. It should never be writable for any other person than root.

For further information about the concepts and the security model of suexec please refer to the suexec documentation (<http://httpd.apache.org/docs/2.0/suexec.html>).



## suexec -V



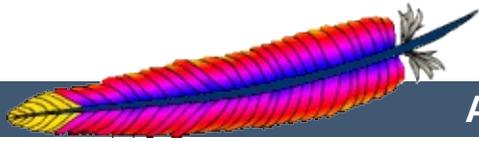
## **-V**

If you are root, this option displays the compile options of suexec. For security reasons all configuration options are changeable only at compile time.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## How-To / Tutorials



## Authentication

Authentication is any process by which you verify that someone is who they claim they are. Authorization is any process by which someone is allowed to be where they want to go, or to have information that they want to have.

See: [Authentication, Authorization, and Access Control](#)

## Dynamic Content with CGI

The CGI (Common Gateway Interface) defines a way for a web server to interact with external content-generating programs, which are often referred to as CGI programs or CGI scripts. It is the simplest, and most common, way to put dynamic content on your web site. This document will be an introduction to setting up CGI on your Apache web server, and getting started writing CGI programs.

See: [CGI: Dynamic Content](#)

## .htaccess files

.htaccess files provide a way to make configuration changes on a per-directory basis. A file, containing one or more configuration directives, is placed in a particular document directory, and the directives apply to that directory, and all subdirectories thereof.

See: [.htaccess files](#)

## Introduction to Server Side Includes

SSI (Server Side Includes) are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. They let you add dynamically generated content to an existing HTML page, without having to serve the entire page via a CGI program, or other dynamic technology.

See: [Server Side Includes \(SSI\)](#)

### **Per-user web directories**

On systems with multiple users, each user can be permitted to have a web site in their home directory using the [UserDir](#) directive. Visitors to a URL

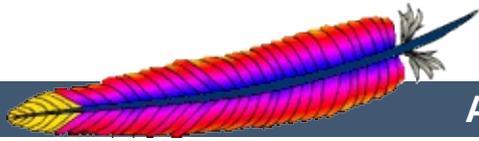
`http://example.com/~username/` will get content out of the home directory of the user "username", out of the subdirectory specified by the [UserDir](#) directive.

See: [User web directories \(public html\)](#)

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

## Apache mod\_rewrite

*``The great thing about mod\_rewrite is it gives you all the configurability and flexibility of Sendmail. The downside to mod\_rewrite is that it gives you all the configurability and flexibility of Sendmail."*

-- Brian Behlendorf  
Apache Group

*`` Despite the tons of examples and docs, mod\_rewrite is voodoo. Damned cool voodoo, but still voodoo. "*

-- Brian Moore  
bem@news.cmc.net

Welcome to mod\_rewrite, the Swiss Army Knife of URL manipulation!

This module uses a rule-based rewriting engine (based on a regular-expression parser) to rewrite requested URLs on the fly. It supports an unlimited number of rules and an unlimited number of attached rule conditions for each rule to provide a really flexible and powerful URL manipulation mechanism. The URL manipulations can depend on various tests, for instance server variables, environment variables, HTTP headers, time stamps and even external database lookups in various formats can be used to achieve granular URL matching.

This module operates on the full URLs (including the path-info part) both in per-server context (`httpd.conf`) and per-directory context (`.htaccess`) and can even generate query-string parts on result. The rewritten result can lead to internal sub-processing, external request redirection or even to an internal proxy throughput.

But all this functionality and flexibility has its drawback: complexity. So don't expect to understand this entire module in just one day.



- [Introduction](#)
- [Technical details](#)
- [Practical solutions to common problems](#)
- [Glossary](#)

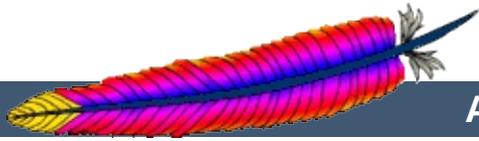


Extensive documentation on the directives provided by this module is provided in the [mod\\_rewrite reference documentation](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

# URL Rewriting Guide

This document supplements the [mod\\_rewrite reference documentation](#). It describes how one can use Apache's [mod\\_rewrite](#) to solve typical URL-based problems with which webmasters are commonly confronted. We give detailed descriptions on how to solve each problem by configuring URL rewriting rulesets.

ATTENTION: Depending on your server configuration it may be necessary to slightly change the examples for your situation, e.g. adding the [PT] flag when additionally using [mod\\_alias](#) and [mod\\_userdir](#), etc. Or rewriting a ruleset to fit in `.htaccess` context instead of per-server context. Always try to understand what a particular ruleset really does before you use it. This avoids many problems.

## See also

[Module documentation](#)  
[mod\\_rewrite introduction](#)  
[Technical details](#)



## Description:

On some web servers there are more than one URL for a resource. Usually there are canonical URLs (which should be actually used and distributed) and those which are just shortcuts, internal ones, etc. Independent of which URL the user supplied with the request he should finally see the canonical one only.

## Solution:

We do an external HTTP redirect for all non-canonical URLs to fix them in the location view of the Browser and for all subsequent requests. In the example ruleset below we replace `/~user` by the canonical `/u/user` and fix a missing trailing slash for `/u/user`.

```
RewriteRule ^/~([^/]+)/?(.*) /u/$1/$2 [R]
RewriteRule ^/([uqe])/([^/]+)$ /$1/$2/ [R]
```



## Description:

The goal of this rule is to force the use of a particular hostname, in preference to other hostnames which may be used to reach the same site. For example, if you wish to force the use of **www.example.com** instead of **example.com**, you might use a variant of the following recipe.

## Solution:

For sites running on a port other than 80:

```
RewriteCond %{HTTP_HOST}    !^fully\.qualified\.domain\.name
RewriteCond %{HTTP_HOST}    !^$
RewriteCond %{SERVER_PORT}  !^80$
RewriteRule ^/(.*)          http://fully.qualified.domain.name
```

And for a site running on port 80

```
RewriteCond %{HTTP_HOST}    !^fully\.qualified\.domain\.name
RewriteCond %{HTTP_HOST}    !^$
RewriteRule ^/(.*)          http://fully.qualified.domain.name
```



## Description:

Usually the [DocumentRoot](#) of the webserver directly relates to the URL "/". But often this data is not really of top-level priority. For example, you may wish for visitors, on first entering a site, to go to a particular subdirectory /about/. This may be accomplished using the following ruleset:

## Solution:

We redirect the URL / to /about/:

```
RewriteEngine on
RewriteRule ^/$ /about/ [R]
```

Note that this can also be handled using the [RedirectMatch](#) directive:

```
RedirectMatch ^/$ http://example.com/e/www/
```



## Description:

The vast majority of "trailing slash" problems can be dealt with using the techniques discussed in the [FAQ entry](#). However, occasionally, there is a need to use `mod_rewrite` to handle a case where a missing trailing slash causes a URL to fail. This can happen, for example, after a series of complex rewrite rules.

## Solution:

The solution to this subtle problem is to let the server add the trailing slash automatically. To do this correctly we have to use an external redirect, so the browser correctly requests subsequent images etc. If we only did a internal rewrite, this would only work for the directory page, but would go wrong when any images are included into this page with relative URLs, because the browser would request an in-lined object. For instance, a request for `image.gif` in `/~quux/foo/index.html` would become `/~quux/image.gif` without the external redirect!

So, to do this trick we write:

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo$ foo/ [R]
```

Alternately, you can put the following in a top-level `.htaccess` file in the content directory. But note that this creates some processing overhead.

```
RewriteEngine on
RewriteBase /~quux/
RewriteCond %{REQUEST_FILENAME} -d
```

```
RewriteRule ^(.+[^/])$ $1/ [R]
```



## Description:

Many webmasters have asked for a solution to the following situation: They wanted to redirect just all homedirs on a webserver to another webserver. They usually need such things when establishing a newer webserver which will replace the old one over time.

## Solution:

The solution is trivial with `mod_rewrite`. On the old webserver we just redirect all `/~user/anypath` URLs to `http://newserver/~user/anypath`.

```
RewriteEngine on
RewriteRule ^/~(.+) http://newserver/~$1 [R,L]
```



### Description:

Sometimes it is necessary to let the webserver search for pages in more than one directory. Here MultiViews or other techniques cannot help.

### Solution:

We program a explicit ruleset which searches for the files in the directories.

```
RewriteEngine on

# first try to find it in custom/...
# ...and if found stop and be happy:
RewriteCond          /your/docroot/dir1/{REQUEST_FILENAME}
RewriteRule  ^(.+) /your/docroot/dir1/$1  [L]

# second try to find it in pub/...
# ...and if found stop and be happy:
RewriteCond          /your/docroot/dir2/{REQUEST_FILENAME}
RewriteRule  ^(.+) /your/docroot/dir2/$1  [L]

# else go on for other Alias or ScriptAlias directives,
# etc.
RewriteRule  ^(.+) - [PT]
```



## Description:

Perhaps you want to keep status information between requests and use the URL to encode it. But you don't want to use a CGI wrapper for all pages just to strip out this information.

## Solution:

We use a rewrite rule to strip out the status information and remember it via an environment variable which can be later dereferenced from within XSSI or CGI. This way a URL `/foo/S=java/bar/` gets translated to `/foo/bar/` and the environment variable named `STATUS` is set to the value `"java"`.

```
RewriteEngine on
RewriteRule ^(.*)/S=([^/]+)/(.*) $1/$3 [E=STATUS:$2]
```



## Description:

Assume that you want to provide `www.username.host.domain.com` for the homepage of `username` via just DNS A records to the same machine and without any virtualhosts on this machine.

## Solution:

For HTTP/1.0 requests there is no solution, but for HTTP/1.1 requests which contain a Host: HTTP header we can use the following ruleset to rewrite `http://www.username.host.com/anypath` internally to `/home/username/anypath`:

```
RewriteEngine on
RewriteCond  %{HTTP_HOST}          ^www\.[^.]+\\.host
RewriteRule  ^(.+)                %{HTTP_HOST}$1
RewriteRule  ^www\.(.[^.]+)\.host\.com(.*) /home/$1$2
```



## Description:

We want to redirect homedir URLs to another webserver `www.somewhere.com` when the requesting user does not stay in the local domain `ourdomain.com`. This is sometimes used in virtual host contexts.

## Solution:

Just a rewrite condition:

```
RewriteEngine on
RewriteCond  %{REMOTE_HOST}  !^.+\.ourdomain\.com$
RewriteRule  ^(/~.+ )        http://www.somewhere.com/$1 [R
```



### **Description:**

By default, redirecting to an HTML anchor doesn't work, because `mod_rewrite` escapes the `#` character, turning it into `%23`. This, in turn, breaks the redirection.

### **Solution:**

Use the `[NE]` flag on the `RewriteRule`. `NE` stands for No Escape.



## Time-Dependent Redirecting

### Description:

When tricks like time-dependent content should happen a lot of webmasters still use CGI scripts which do for instance redirects to specialized pages. How can it be done via [mod\\_rewrite](#)?

### Solution:

There are a lot of variables named TIME\_XXX for rewrite conditions. In conjunction with the special lexicographic comparison patterns <STRING, >STRING and =STRING we can do time-dependent redirects:

```
RewriteEngine on
RewriteCond    %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond    %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule    ^foo\.html$              foo.day.html
RewriteRule    ^foo\.html$              foo.night.html
```

This provides the content of `foo.day.html` under the URL `foo.html` from `07:00-19:00` and at the remaining time the contents of `foo.night.html`. Just a nice feature for a homepage...



## Description:

How can we make URLs backward compatible (still existing virtually) after migrating document .YYYY to document .XXXX, e.g. after translating a bunch of .html files to .phtml?

## Solution:

We just rewrite the name to its basename and test for existence of the new extension. If it exists, we take that name, else we rewrite the URL to its original state.

```
# backward compatibility ruleset for
# rewriting document.html to document.phtml
# when and only when document.phtml exists
# but no longer document.html
RewriteEngine on
RewriteBase /~quux/
# parse out basename, but remember the fact
RewriteRule ^(.*)\.html$ $1 [C,E=wasHTML]
# rewrite to document.phtml if exists
RewriteCond %{REQUEST_FILENAME}.phtml -f
RewriteRule ^(.*)$ $1.phtml [S=1]
# else reverse the previous basename cutout
RewriteCond %{ENV:wasHTML} ^yes$
RewriteRule ^(.*)$ $1.html
```



## From Old to New (intern)

### Description:

Assume we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. Actually we want that users of the old URL even not recognize that the pages was renamed.

### Solution:

We rewrite the old URL to the new one internally via the following rule:

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ bar.html
```

## From Old to New (extern)

### Description:

Assume again that we have recently renamed the page `foo.html` to `bar.html` and now want to provide the old URL for backward compatibility. But this time we want that the users of the old URL get hinted to the new one, i.e. their browsers Location field should change, too.

### Solution:

We force a HTTP redirect to the new URL which leads to a change of the browsers and thus the users view:

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ bar.html [R]
```

## From Static to Dynamic

### Description:

How can we transform a static page `foo.html` into a dynamic variant `foo.cgi` in a seamless way, i.e. without notice by the browser/user.

### Solution:

We just rewrite the URL to the CGI-script and force the correct MIME-type so it gets really run as a CGI-script. This way a request to `/~quux/foo.html` internally leads to the invocation of `/~quux/foo.cgi`.

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^foo\.html$ foo.cgi [T=application/x-httpd-
```



## Blocking of Robots

### Description:

How can we block a really annoying robot from retrieving pages of a specific webarea? A `/robots.txt` file containing entries of the "Robot Exclusion Protocol" is typically not enough to get rid of such a robot.

### Solution:

We use a ruleset which forbids the URLs of the webarea `/~quux/foo/arc/` (perhaps a very deep directory indexed area where the robot traversal would create big server load). We have to make sure that we forbid access only to the particular robot, i.e. just forbidding the host where the robot runs is not enough. This would block users from this host, too. We accomplish this by also matching the User-Agent HTTP header information.

```
RewriteCond %{HTTP_USER_AGENT} ^NameOfBadRobot.*
RewriteCond %{REMOTE_ADDR} ^123\.45\.67\.[8-9]$
RewriteRule ^/~quux/foo/arc/.+ - [F]
```

## Blocked Inline-Images

### Description:

Assume we have under `http://www.quux-corp.de/~quux/` some pages with inlined GIF graphics. These graphics are nice, so others directly incorporate them via hyperlinks to their pages. We don't like this practice because it adds useless traffic to our server.

### Solution:

While we cannot 100% protect the images from inclusion, we

can at least restrict the cases where the browser sends a HTTP Referer header.

```
RewriteCond %{HTTP_REFERER} !^$  
RewriteCond %{HTTP_REFERER} !^http://www.quux-corp.de/~quux/  
RewriteRule .*\.gif$ -
```

```
RewriteCond %{HTTP_REFERER} !^$  
RewriteCond %{HTTP_REFERER} !.* /foo-with-gif\.html$  
RewriteRule ^inlined-in-foo\.gif$ -
```

## Proxy Deny

### Description:

How can we forbid a certain host or even a user of a special host from using the Apache proxy?

### Solution:

We first have to make sure `mod_rewrite` is below(!) `mod_proxy` in the Configuration file when compiling the Apache webserver. This way it gets called *before* `mod_proxy`. Then we configure the following for a host-dependent deny...

```
RewriteCond %{REMOTE_HOST} ^badhost\.mydomain\.com$  
RewriteRule !^http://[^\./]\.mydomain.com.* - [F]
```

...and this one for a user@host-dependent deny:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} ^badguy@badhost\  
RewriteRule !^http://[^\./]\.mydomain.com.* - [F]
```



## External Rewriting Engine

### Description:

A FAQ: How can we solve the FOO/BAR/QUUX/etc. problem?  
There seems no solution by the use of `mod_rewrite`...

### Solution:

Use an external `RewriteMap`, i.e. a program which acts like a `RewriteMap`. It is run once on startup of Apache receives the requested URLs on STDIN and has to put the resulting (usually rewritten) URL on STDOUT (same order!).

```
RewriteEngine on
RewriteMap    quux-map      prog:/path/to/map.quux.pl
RewriteRule   ^/~quux/(.*)$ /~quux/${quux-map:$1}
```

```
#!/path/to/perl

#  disable buffered I/O which would lead
#  to deadloops for the Apache server
$| = 1;

#  read URLs one per line from stdin and
#  generate substitution URL on stdout
while (<>) {
    s|^foo/|bar/|;
    print $_;
}
```

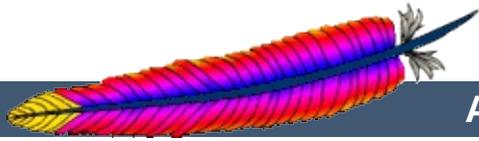
This is a demonstration-only example and just rewrites all URLs `/~quux/foo/...` to `/~quux/bar/...`. Actually you can program whatever you like. But notice that while such

maps can be **used** also by an average user, only the system administrator can **define** it.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

# URL Rewriting Guide - Advanced topics

This document supplements the [mod\\_rewrite reference documentation](#). It describes how one can use Apache's [mod\\_rewrite](#) to solve typical URL-based problems with which webmasters are commonly confronted. We give detailed descriptions on how to solve each problem by configuring URL rewriting rulesets.

ATTENTION: Depending on your server configuration it may be necessary to adjust the examples for your situation, e.g., adding the [PT] flag if using [mod\\_alias](#) and [mod\\_userdir](#), etc. Or rewriting a ruleset to work in .htaccess context instead of per-server context. Always try to understand what a particular ruleset really does before you use it; this avoids many problems.

## See also

[Module documentation](#)  
[mod\\_rewrite introduction](#)  
[Technical details](#)



## Description:

We want to create a homogeneous and consistent URL layout across all WWW servers on an Intranet web cluster, i.e., all URLs (by definition server-local and thus server-dependent!) become server *independent*! What we want is to give the WWW namespace a single consistent layout: no URL should refer to any particular target server. The cluster itself should connect users automatically to a physical target host as needed, invisibly.

## Solution:

First, the knowledge of the target servers comes from (distributed) external maps which contain information on where our users, groups, and entities reside. They have the form:

```
user1  server_of_user1
user2  server_of_user2
:      :
```

We put them into files `map.xxx-to-host`. Second we need to instruct all servers to redirect URLs of the forms:

```
/u/user/anypath
/g/group/anypath
/e/entity/anypath
```

to

```
http://physical-host/u/user/anypath
http://physical-host/g/group/anypath
http://physical-host/e/entity/anypath
```

when any URL path need not be valid on every server. The following ruleset does this for us with the help of the map files (assuming that server0 is a default server which will be used if a user has no entry in the map):

```
RewriteEngine on

RewriteMap      user-to-host      txt:/path/to/map.user-to-host
RewriteMap      group-to-host     txt:/path/to/map.group-to-hos
RewriteMap      entity-to-host    txt:/path/to/map.entity-to-ho

RewriteRule     ^/u/([^/]+)/?(.*) http://${user-to-host:$1|s
RewriteRule     ^/g/([^/]+)/?(.*) http://${group-to-host:$1|s
RewriteRule     ^/e/([^/]+)/?(.*) http://${entity-to-host:$1|s

RewriteRule     ^/([uge])/([^/]+)/?$          /$1/$2/.www/
RewriteRule     ^/([uge])/([^/]+)/([^.]+.+)$ /$1/$2/.www/$3\
```



## Description:

Some sites with thousands of users use a structured homedir layout, *i.e.* each homedir is in a subdirectory which begins (for instance) with the first character of the username. So, `/~foo/anypath` is `/home/f/foo/.www/anypath` while `/~bar/anypath` is `/home/b/bar/.www/anypath`.

## Solution:

We use the following ruleset to expand the tilde URLs into the above layout.

```
RewriteEngine on
RewriteRule ^/~(([a-z])[a-z0-9]+)(.*) /home/$2/$1/.www$3
```



## Description:

This really is a hardcore example: a killer application which heavily uses per-directory RewriteRules to get a smooth look and feel on the Web while its data structure is never touched or adjusted. Background: *net.sw* is my archive of freely available Unix software packages, which I started to collect in 1992. It is both my hobby and job to do this, because while I'm studying computer science I have also worked for many years as a system and network administrator in my spare time. Every week I need some sort of software so I created a deep hierarchy of directories where I stored the packages:

```
drwxrwxr-x  2 netsw  users    512 Aug  3 18:39 Audio/
drwxrwxr-x  2 netsw  users    512 Jul  9 14:37 Benchmark/
drwxrwxr-x 12 netsw  users    512 Jul  9 00:34 Crypto/
drwxrwxr-x  5 netsw  users    512 Jul  9 00:41 Database/
drwxrwxr-x  4 netsw  users    512 Jul 30 19:25 Dicts/
drwxrwxr-x 10 netsw  users    512 Jul  9 01:54 Graphic/
drwxrwxr-x  5 netsw  users    512 Jul  9 01:58 Hackers/
drwxrwxr-x  8 netsw  users    512 Jul  9 03:19 InfoSys/
drwxrwxr-x  3 netsw  users    512 Jul  9 03:21 Math/
drwxrwxr-x  3 netsw  users    512 Jul  9 03:24 Misc/
drwxrwxr-x  9 netsw  users    512 Aug  1 16:33 Network/
drwxrwxr-x  2 netsw  users    512 Jul  9 05:53 Office/
drwxrwxr-x  7 netsw  users    512 Jul  9 09:24 SoftEng/
drwxrwxr-x  7 netsw  users    512 Jul  9 12:17 System/
drwxrwxr-x 12 netsw  users    512 Aug  3 20:15 Typesetting/
drwxrwxr-x 10 netsw  users    512 Jul  9 14:08 X11/
```

In July 1996 I decided to make this archive public to the world via a nice Web interface. "Nice" means that I wanted to offer an interface where you can browse directly through the

archive hierarchy. And "nice" means that I didn't want to change anything inside this hierarchy - not even by putting some CGI scripts at the top of it. Why? Because the above structure should later be accessible via FTP as well, and I didn't want any Web or CGI stuff mixed in there.

### Solution:

The solution has two parts: The first is a set of CGI scripts which create all the pages at all directory levels on-the-fly. I put them under `/e/netsw/.www/` as follows:

```
-rw-r--r--  1 netsw  users    1318 Aug  1 18:10 .wwwacl
drwxr-xr-x 18 netsw  users     512 Aug  5 15:51 DATA/
-rw-rw-rw-  1 netsw  users  372982 Aug  5 16:35 LOGFILE
-rw-r--r--  1 netsw  users     659 Aug  4 09:27 TODO
-rw-r--r--  1 netsw  users    5697 Aug  1 18:01 netsw-about
-rwxr-xr-x  1 netsw  users     579 Aug  2 10:33 netsw-acces
-rwxr-xr-x  1 netsw  users    1532 Aug  1 17:35 netsw-chang
-rwxr-xr-x  1 netsw  users    2866 Aug  5 14:49 netsw-home.
drwxr-xr-x  2 netsw  users     512 Jul  8 23:47 netsw-img/
-rwxr-xr-x  1 netsw  users   24050 Aug  5 15:49 netsw-lsdir
-rwxr-xr-x  1 netsw  users    1589 Aug  3 18:43 netsw-searc
-rwxr-xr-x  1 netsw  users    1885 Aug  1 17:41 netsw-tree.
-rw-r--r--  1 netsw  users     234 Jul 30 16:35 netsw-unlim
```

The `DATA/` subdirectory holds the above directory structure, *i.e.* the real **net.sw** stuff, and gets automatically updated via `rdist` from time to time. The second part of the problem remains: how to link these two structures together into one smooth-looking URL tree? We want to hide the `DATA/` directory from the user while running the appropriate CGI scripts for the various URLs. Here is the solution: first I put the following into the per-directory configuration file in the [DocumentRoot](#) of the server to rewrite the public URL path

[/net.sw/](#) to the internal path `/e/netsw`:

```
RewriteRule ^net.sw$ net.sw/ [R]
RewriteRule ^net.sw/(.*)$ e/netsw/$1
```

The first rule is for requests which miss the trailing slash! The second rule does the real thing. And then comes the killer configuration which stays in the per-directory config file `/e/netsw/.www/.wwwacl`:

```
Options ExecCGI FollowSymLinks Includes MultiViews

RewriteEngine on

# we are reached via /net.sw/ prefix
RewriteBase /net.sw/

# first we rewrite the root dir to
# the handling cgi script
RewriteRule ^$ netsw-home.cgi [L]
RewriteRule ^index\.html$ netsw-home.cgi [L]

# strip out the subdirs when
# the browser requests us from perdir pages
RewriteRule ^.+/(netsw-[^/]+/+.+)$ $1 [L]

# and now break the rewriting for local files
RewriteRule ^netsw-home\.cgi.* - [L]
RewriteRule ^netsw-changes\.cgi.* - [L]
RewriteRule ^netsw-search\.cgi.* - [L]
RewriteRule ^netsw-tree\.cgi$ - [L]
RewriteRule ^netsw-about\.html$ - [L]
RewriteRule ^netsw-img/.*$ - [L]
```

```
# anything else is a subdir which gets handled
# by another cgi script
RewriteRule    !^netsw-lsdir\.cgi.*      -                [C
RewriteRule    (.*)                      netsw-lsdir.cgi/$1
```

Some hints for interpretation:

1. Notice the L (last) flag and no substitution field ('-') in the fourth part
2. Notice the ! (not) character and the C (chain) flag at the first rule in the last part
3. Notice the catch-all pattern in the last rule



## Description:

A typical FAQ about URL rewriting is how to redirect failing requests on webserver A to webserver B. Usually this is done via [ErrorDocument](#) CGI scripts in Perl, but there is also a [mod\\_rewrite](#) solution. But note that this performs more poorly than using an [ErrorDocument](#) CGI script!

## Solution:

The first solution has the best performance but less flexibility, and is less safe:

```
RewriteEngine on
RewriteCond    /your/docroot/%{REQUEST_FILENAME} !-f
RewriteRule    ^(.+)                                http://webse
```

The problem here is that this will only work for pages inside the [DocumentRoot](#). While you can add more Conditions (for instance to also handle homedirs, etc.) there is a better variant:

```
RewriteEngine on
RewriteCond    %{REQUEST_URI} !-U
RewriteRule    ^(.+)                http://webserverB.dom/$1
```

This uses the URL look-ahead feature of [mod\\_rewrite](#). The result is that this will work for all types of URLs and is safe. But it does have a performance impact on the web server, because for every request there is one more internal subrequest. So, if your web server runs on a powerful CPU, use this one. If it is a slow machine, use the first approach or better an [ErrorDocument](#) CGI script.



## Description:

Do you know the great CPAN (Comprehensive Perl Archive Network) under <http://www.perl.com/CPAN>? CPAN automatically redirects browsers to one of many FTP servers around the world (generally one near the requesting client); each server carries a full CPAN mirror. This is effectively an FTP access multiplexing service. CPAN runs via CGI scripts, but how could a similar approach be implemented via [mod\\_rewrite](#)?

## Solution:

First we notice that as of version 3.0.0, [mod\\_rewrite](#) can also use the "ftp:" scheme on redirects. And second, the location approximation can be done by a [RewriteMap](#) over the top-level domain of the client. With a tricky chained ruleset we can use this top-level domain as a key to our multiplexing map.

```
RewriteEngine on
RewriteMap    multiplex          txt:/path/to/map.cxan
RewriteRule   ^/CxAN/(.*)       %{REMOTE_HOST}::$1
RewriteRule   ^.+\.([a-zA-Z]+)::(.*)$  ${multiplex:$1|ftp.de
```

```
##
##  map.cxan -- Multiplexing Map for CxAN
##

de      ftp://ftp.cxan.de/CxAN/
uk      ftp://ftp.cxan.uk/CxAN/
com     ftp://ftp.cxan.com/CxAN/
:
##EOF##
```



## Browser Dependent Content

### Description:

At least for important top-level pages it is sometimes necessary to provide the optimum of browser dependent content, i.e., one has to provide one version for current browsers, a different version for the Lynx and text-mode browsers, and another for other browsers.

### Solution:

We cannot use content negotiation because the browsers do not provide their type in that form. Instead we have to act on the HTTP header "User-Agent". The following config does the following: If the HTTP header "User-Agent" begins with "Mozilla/3", the page `foo.html` is rewritten to `foo.NS.html` and the rewriting stops. If the browser is "Lynx" or "Mozilla" of version 1 or 2, the URL becomes `foo.20.html`. All other browsers receive page `foo.32.html`. This is done with the following ruleset:

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/3
RewriteRule ^foo\.html$      foo.NS.html      [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx/          [OR]
RewriteCond %{HTTP_USER_AGENT} Mozilla/[12]
RewriteRule ^foo\.html$      foo.20.html      [L]

RewriteRule ^foo\.html$      foo.32.html      [L]
```

## Dynamic Mirror

### Description:

Assume there are nice web pages on remote hosts we want

to bring into our namespace. For FTP servers we would use the `mirror` program which actually maintains an explicit up-to-date copy of the remote data on the local machine. For a web server we could use the program `webcopy` which runs via HTTP. But both techniques have a major drawback: The local copy is always only as up-to-date as the last time we ran the program. It would be much better if the mirror was not a static one we have to establish explicitly. Instead we want a dynamic mirror with data which gets updated automatically as needed on the remote host(s).

### Solution:

To provide this feature we map the remote web page or even the complete remote web area to our namespace by the use of the *Proxy Throughput* feature (flag [P]):

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^hotsheet/(.*)$ http://www.tstimpreso.com/ho
```

```
RewriteEngine on
RewriteBase /~quux/
RewriteRule ^usa-news\.html$ http://www.quux-corp.com/n
```

## Reverse Dynamic Mirror

### Description:

...

### Solution:

```
RewriteEngine on
RewriteCond /mirror/of/remotesite/$1 -U
RewriteRule ^http://www\.remotesite\.com/(.*)$ /mirror/of/
```

## Retrieve Missing Data from Intranet

### Description:

This is a tricky way of virtually running a corporate (external) Internet web server (`www.quux-corp.dom`), while actually keeping and maintaining its data on an (internal) Intranet web server (`www2.quux-corp.dom`) which is protected by a firewall. The trick is that the external web server retrieves the requested data on-the-fly from the internal one.

### Solution:

First, we must make sure that our firewall still protects the internal web server and only the external web server is allowed to retrieve data from it. On a packet-filtering firewall, for instance, we could configure a firewall ruleset like the following:

```
ALLOW Host www.quux-corp.dom Port >1024 --> Host www2.quux-c
DENY  Host *                      Port *          --> Host www2.quux-c
```

Just adjust it to your actual configuration syntax. Now we can establish the [mod\\_rewrite](#) rules which request the missing data in the background through the proxy throughput feature:

```
RewriteRule ^/~([^/]+)/?(.*) /home/$1/.www/$2
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^/home/([^/]+)/.www/?(.*) http://www2.quux-corp.
```

## Load Balancing

### Description:

Suppose we want to load balance the traffic to `www.foo.com` over `www[0-5].foo.com` (a total of 6 servers). How can this

be done?

## Solution:

There are many possible solutions for this problem. We will first discuss a common DNS-based method, and then one based on [mod\\_rewrite](#):

### 1. DNS Round-Robin

The simplest method for load-balancing is to use DNS round-robin. Here you just configure `www[0-9].foo.com` as usual in your DNS with A (address) records, e.g.,

```
www0    IN  A      1.2.3.1
www1    IN  A      1.2.3.2
www2    IN  A      1.2.3.3
www3    IN  A      1.2.3.4
www4    IN  A      1.2.3.5
www5    IN  A      1.2.3.6
```

Then you additionally add the following entries:

```
www     IN  A      1.2.3.1
www     IN  A      1.2.3.2
www     IN  A      1.2.3.3
www     IN  A      1.2.3.4
www     IN  A      1.2.3.5
```

Now when `www.foo.com` gets resolved, BIND gives out `www0-www5` - but in a permuted (rotated) order every time. This way the clients are spread over the various servers. But notice that this is not a perfect load balancing scheme, because DNS resolutions are cached by clients and other nameservers, so once a client has

resolved `www.foo.com` to a particular `wwwN.foo.com`, all its subsequent requests will continue to go to the same IP (and thus a single server), rather than being distributed across the other available servers. But the overall result is okay because the requests are collectively spread over the various web servers.

## 2. DNS Load-Balancing

A sophisticated DNS-based method for load-balancing is to use the program `lbname` which can be found at <http://www.stanford.edu/~schemers/docs/lbname/lbname>. It is a Perl 5 program which, in conjunction with auxiliary tools, provides real load-balancing via DNS.

## 3. Proxy Throughput Round-Robin

In this variant we use `mod_rewrite` and its proxy throughput feature. First we dedicate `www0.foo.com` to be actually `www.foo.com` by using a single

```
www    IN    CNAME    www0.foo.com.
```

entry in the DNS. Then we convert `www0.foo.com` to a proxy-only server, i.e., we configure this machine so all arriving URLs are simply passed through its internal proxy to one of the 5 other servers (`www1-www5`). To accomplish this we first establish a ruleset which contacts a load balancing script `lb.pl` for all URLs.

```
RewriteEngine on
RewriteMap    lb      prg:/path/to/lb.pl
RewriteRule   ^/(.+)$ ${lb:$1}          [P,L]
```

Then we write `lb.pl`:

```

#!/path/to/perl
##
## lb.pl -- load balancing script
##

$| = 1;

$name     = "www";      # the hostname base
$first    = 1;          # the first server (not 0 here, bec
$last     = 5;          # the last server in the round-robi
$domain   = "foo.dom"; # the domainname

$cnt = 0;
while (<STDIN>) {
    $cnt = (($cnt+1) % ($last+1-$first));
    $server = sprintf("%s%d.%s", $name, $cnt+$first, $dc
    print "http://$server/$_";
}

##EOF##

```

A last notice: Why is this useful? Seems like `www0.foo.com` still is overloaded? The answer is yes, it is overloaded, but with plain proxy throughput requests, only! All SSI, CGI, ePerl, etc. processing is handled done on the other machines. For a complicated site, this may work well. The biggest risk here is that `www0` is now a single point of failure -- if it crashes, the other servers are inaccessible.

#### 4. Dedicated Load Balancers

There are more sophisticated solutions, as well. Cisco, F5, and several other companies sell hardware load

balancers (typically used in pairs for redundancy), which offer sophisticated load balancing and auto-failover features. There are software packages which offer similar features on commodity hardware, as well. If you have enough money or need, check these out. The [lb-l mailing list](#) is a good place to research.

## New MIME-type, New Service

### Description:

On the net there are many nifty CGI programs. But their usage is usually boring, so a lot of webmasters don't use them. Even Apache's Action handler feature for MIME-types is only appropriate when the CGI programs don't need special URLs (actually PATH\_INFO and QUERY\_STRING) as their input. First, let us configure a new file type with extension `.scgi` (for secure CGI) which will be processed by the popular `cgwrap` program. The problem here is that for instance if we use a Homogeneous URL Layout (see above) a file inside the user homedirs might have a URL like `/u/user/foo/bar.scgi`, but `cgwrap` needs URLs in the form `/~user/foo/bar.scgi/`. The following rule solves the problem:

```
RewriteRule ^/[uqe]/([^\s/]+)/\.\www/(.+)\.scgi(.*). . .  
. . . /internal/cgi/user/cgwrap/~$1/$2.scgi$3 [NS,T=applicat
```

Or assume we have some more nifty programs: `wwwlog` (which displays the `access.log` for a URL subtree) and `wwidx` (which runs Glimpse on a URL subtree). We have to provide the URL area to these programs so they know which area they are really working with. But usually this is complicated, because they may still be requested by the alternate URL form, i.e., typically we would run the `swidx`

program from within `/u/user/foo/` via hyperlink to

```
/internal/cgi/user/swwidx?i=/u/user/foo/
```

which is ugly, because we have to hard-code **both** the location of the area **and** the location of the CGI inside the hyperlink. When we have to reorganize, we spend a lot of time changing the various hyperlinks.

### **Solution:**

The solution here is to provide a special new URL format which automatically leads to the proper CGI invocation. We configure the following:

```
RewriteRule ^/([uge])/([^/]+)(/?.*)\^* /internal/cgi/user
RewriteRule ^/([uge])/([^/]+)(/?.*):log /internal/cgi/user
```

Now the hyperlink to search at `/u/user/foo/` reads only

```
HREF="*"
```

which internally gets automatically transformed to

```
/internal/cgi/user/wwwidx?i=/u/user/foo/
```

The same approach leads to an invocation for the access log CGI program when the hyperlink `:log` gets used.

## **On-the-fly Content-Regeneration**

### **Description:**

Here comes a really esoteric feature: Dynamically generated but statically served pages, i.e., pages should be delivered as

pure static pages (read from the filesystem and just passed through), but they have to be generated dynamically by the web server if missing. This way you can have CGI-generated pages which are statically served unless an admin (or a cron job) removes the static contents. Then the contents gets refreshed.

### **Solution:**

This is done via the following ruleset:

```
RewriteCond %{REQUEST_FILENAME}    !-s
RewriteRule ^page\.html$           page.cgi    [T=application/
```

Here a request for `page.html` leads to an internal run of a corresponding `page.cgi` if `page.html` is missing or has filesize null. The trick here is that `page.cgi` is a CGI script which (additionally to its `STDOUT`) writes its output to the file `page.html`. Once it has completed, the server sends out `page.html`. When the webmaster wants to force a refresh of the contents, he just removes `page.html` (typically from cron).

## **Document With Autorefresh**

### **Description:**

Wouldn't it be nice, while creating a complex web page, if the web browser would automatically refresh the page every time we save a new version from within our editor? Impossible?

### **Solution:**

No! We just combine the MIME multipart feature, the web server NPH feature, and the URL manipulation power of [mod\\_rewrite](#). First, we establish a new URL feature: Adding just `:refresh` to any URL causes the 'page' to be refreshed

every time it is updated on the filesystem.

```
RewriteRule ^(/[uge]/[^/]+/?.*):refresh /internal/cgi/apa
```

Now when we reference the URL

```
/u/foo/bar/page.html:refresh
```

this leads to the internal invocation of the URL

```
/internal/cgi/apache/nph-refresh?f=/u/foo/bar/page.html
```

The only missing part is the NPH-CGI script. Although one would usually say "left as an exercise to the reader" ;-)) I will provide this, too.

```
#!/sw/bin/perl
##
## nph-refresh -- NPH/CGI script for auto refreshing pages
## Copyright (c) 1997 Ralf S. Engelschall, All Rights Reser
##
$| = 1;

# split the QUERY_STRING variable
@pairs = split(/&/, $ENV{'QUERY_STRING'});
foreach $pair (@pairs) {
    ($name, $value) = split(/=/, $pair);
    $name =~ tr/A-Z/a-z/;
    $name = 'QS_' . $name;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack("C", hex($1))/;
    eval "\$$name = \"$value\"";
}
$QS_s = 1 if ($QS_s eq '');
```

```
$QS_n = 3600 if ($QS_n eq '');
if ($QS_f eq '') {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: No file given\n";
    exit(0);
}
if (! -f $QS_f) {
    print "HTTP/1.0 200 OK\n";
    print "Content-type: text/html\n\n";
    print "&lt;b&gt;ERROR&lt;/b&gt;: File $QS_f not found\n";
    exit(0);
}

sub print_http_headers_multipart_begin {
    print "HTTP/1.0 200 OK\n";
    $bound = "ThisRandomString12345";
    print "Content-type: multipart/x-mixed-replace;boundary="
    &print_http_headers_multipart_next;
}

sub print_http_headers_multipart_next {
    print "\n--$bound\n";
}

sub print_http_headers_multipart_end {
    print "\n--$bound--\n";
}

sub displayhtml {
    local($buffer) = @_;
    $len = length($buffer);
    print "Content-type: text/html\n";
    print "Content-length: $len\n\n";
}
```

```

    print $buffer;
}

sub readfile {
    local($file) = @_;
    local(*FP, $size, $buffer, $bytes);
    ($x, $x, $x, $x, $x, $x, $x, $size) = stat($file);
    $size = sprintf("%d", $size);
    open(FP, "&lt;$file");
    $bytes = sysread(FP, $buffer, $size);
    close(FP);
    return $buffer;
}

$buffer = &readfile($QS_f);
&print_http_headers_multipart_begin;
&displayhtml($buffer);

sub mystat {
    local($file) = $_[0];
    local($time);

    ($x, $x, $x, $x, $x, $x, $x, $x, $x, $mtime) = stat($file);
    return $mtime;
}

$mtimeL = &mystat($QS_f);
$mtime = $mtime;
for ($n = 0; $n &lt; $QS_n; $n++) {
    while (1) {
        $mtime = &mystat($QS_f);
        if ($mtime ne $mtimeL) {
            $mtimeL = $mtime;
            sleep(2);
        }
    }
}

```

```

        $buffer = &readfile($QS_f);
        &print_http_headers_multipart_next;
        &displayhtml($buffer);
        sleep(5);
        $mtimeL = &mystat($QS_f);
        last;
    }
    sleep($QS_s);
}
}

&print_http_headers_multipart_end;

exit(0);

##EOF##

```

## Mass Virtual Hosting

### Description:

The [VirtualHost](#) feature of Apache is nice and works great when you just have a few dozen virtual hosts. But when you are an ISP and have hundreds of virtual hosts, this feature is suboptimal.

### Solution:

To provide this feature we map the remote web page or even the complete remote web area to our namespace using the *Proxy Throughput* feature (flag [P]):

```

##
## vhost.map
##
www.vhost1.dom:80 /path/to/docroot/vhost1

```

```
www.vhost2.dom:80 /path/to/docroot/vhost2
:
www.vhostN.dom:80 /path/to/docroot/vhostN
```

```
##
## httpd.conf
##
:
# use the canonical hostname on redirects, etc.
UseCanonicalName on

:
# add the virtual host in front of the CLF-format
CustomLog /path/to/access_log "%{VHOST}e %h %l %u %t \"%r\"
:

# enable the rewriting engine in the main server
RewriteEngine on

# define two maps: one for fixing the URL and one which de
# the available virtual hosts with their corresponding
# DocumentRoot.
RewriteMap lowercase int:tolower
RewriteMap vhost txt:/path/to/vhost.map

# Now do the actual virtual host mapping
# via a huge and complicated single rule:
#
# 1. make sure we don't map for common locations
RewriteCond %{REQUEST_URI} !^/commonurl1/.*
RewriteCond %{REQUEST_URI} !^/commonurl2/.*
:
RewriteCond %{REQUEST_URI} !^/commonurlN/.*
#
```

```
# 2. make sure we have a Host header, because
#     currently our approach only supports
#     virtual hosting through this header
RewriteCond  %{HTTP_HOST}  !^$
#
# 3. lowercase the hostname
RewriteCond  ${lowercase:%{HTTP_HOST}|NONE}  ^(.+)$
#
# 4. lookup this hostname in vhost.map and
#     remember it only when it is a path
#     (and not "NONE" from above)
RewriteCond  ${vhost:%1}  ^(/.*)$
#
# 5. finally we can map the URL to its docroot location
#     and remember the virtual host for logging purposes
RewriteRule  ^/(.*)$  %1/$1  [E=VHOST:${lowercase:%{HTTP_H
:

```



## Host Deny

### Description:

How can we forbid a list of externally configured hosts from using our server?

### Solution:

For Apache >= 1.3b6:

```
RewriteEngine on
RewriteMap    hosts-deny    txt:/path/to/hosts.deny
RewriteCond   ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND} !=NOT-F
RewriteCond   ${hosts-deny:%{REMOTE_ADDR}|NOT-FOUND} !=NOT-F
RewriteRule   ^/.* - [F]
```

For Apache <= 1.3b6:

```
RewriteEngine on
RewriteMap    hosts-deny    txt:/path/to/hosts.deny
RewriteRule   ^/(.*)$ ${hosts-deny:%{REMOTE_HOST}|NOT-FOUND}
RewriteRule   !^NOT-FOUND/.* - [F]
RewriteRule   ^NOT-FOUND/(.*)$ ${hosts-deny:%{REMOTE_ADDR}|N
RewriteRule   !^NOT-FOUND/.* - [F]
RewriteRule   ^NOT-FOUND/(.*)$ /$1
```

```
##
##  hosts.deny
##
##  ATTENTION! This is a map, not a list, even when we treat
##              mod_rewrite parses it for key/value pairs, so
##              dummy value "-" must be present for each entr
##
```

```
193.102.180.41 -  
bsdti1.sdm.de -  
192.76.162.40 -
```

## Proxy Deny

### Description:

How can we forbid a certain host or even a user of a special host from using the Apache proxy?

### Solution:

We first have to make sure `mod_rewrite` is below(!) `mod_proxy` in the Configuration file when compiling the Apache web server. This way it gets called *before* `mod_proxy`. Then we configure the following for a host-dependent deny...

```
RewriteCond %{REMOTE_HOST} ^badhost\.mydomain\.com$  
RewriteRule !^http://[^\./]\.mydomain.com.* - [F]
```

...and this one for a user@host-dependent deny:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} ^badguy@badhost\  
RewriteRule !^http://[^\./]\.mydomain.com.* - [F]
```

## Special Authentication Variant

### Description:

Sometimes very special authentication is needed, for instance authentication which checks for a set of explicitly configured users. Only these should receive access and without explicit prompting (which would occur when using Basic Auth via `mod_auth`).

## Solution:

We use a list of rewrite conditions to exclude all except our friends:

```
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend1@client1
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend2@client2
RewriteCond %{REMOTE_IDENT}@%{REMOTE_HOST} !^friend3@client3
RewriteRule ^/~quux/only-for-friends/ -
```

## Referer-based Deflector

### Description:

How can we program a flexible URL Deflector which acts on the "Referer" HTTP header and can be configured with as many referring pages as we like?

### Solution:

Use the following really tricky ruleset...

```
RewriteMap deflector txt:/path/to/deflector.map

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}} ^-$
RewriteRule ^.* %{HTTP_REFERER} [R,L]

RewriteCond %{HTTP_REFERER} !=""
RewriteCond ${deflector:%{HTTP_REFERER}|NOT-FOUND} !=NOT-FOU
RewriteRule ^.* ${deflector:%{HTTP_REFERER}} [R,L]
```

... in conjunction with a corresponding rewrite map:

```
##
## deflector.map
##
```

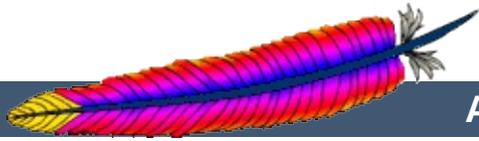
```
http://www.badguys.com/bad/index.html -  
http://www.badguys.com/bad/index2.html -  
http://www.badguys.com/bad/index3.html http://somewhere.co
```

This automatically redirects the request back to the referring page (when "-" is used as the value in the map) or to a specific URL (when an URL is specified in the map as the second argument).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

# Apache mod\_rewrite Technical Details

This document discusses some of the technical details of mod\_rewrite and URL matching.

## See also

[Module documentation](#)

[mod\\_rewrite introduction](#)

[Practical solutions to common problems](#)



The internal processing of this module is very complex but needs to be explained once even to the average user to avoid common mistakes and to let you exploit its full functionality.



First you have to understand that when Apache processes a HTTP request it does this in phases. A hook for each of these phases is provided by the Apache API. Mod\_rewrite uses two of these hooks: the URL-to-filename translation hook which is used after the HTTP request has been read but before any authorization starts and the Fixup hook which is triggered after the authorization phases and after the per-directory config files (`.htaccess`) have been read, but before the content handler is activated.

So, after a request comes in and Apache has determined the corresponding server (or virtual server) the rewriting engine starts processing of all mod\_rewrite directives from the per-server configuration in the URL-to-filename phase. A few steps later when the final data directories are found, the per-directory configuration directives of mod\_rewrite are triggered in the Fixup phase. In both situations mod\_rewrite rewrites URLs either to new URLs or to filenames, although there is no obvious distinction between them. This is a usage of the API which was not intended to be this way when the API was designed, but as of Apache 1.x this is the only way mod\_rewrite can operate. To make this point more clear remember the following two points:

1. Although mod\_rewrite rewrites URLs to URLs, URLs to filenames and even filenames to filenames, the API currently provides only a URL-to-filename hook. In Apache 2.0 the two missing hooks will be added to make the processing more clear. But this point has no drawbacks for the user, it is just a fact which should be remembered: Apache does more in the URL-to-filename hook than the API intends for it.
2. Unbelievably mod\_rewrite provides URL manipulations in per-directory context, *i.e.*, within `.htaccess` files, although these are reached a very long time after the URLs have been translated to filenames. It has to be this way because

.htaccess files live in the filesystem, so processing has already reached this stage. In other words: According to the API phases at this time it is too late for any URL manipulations. To overcome this chicken and egg problem mod\_rewrite uses a trick: When you manipulate a URL/filename in per-directory context mod\_rewrite first rewrites the filename back to its corresponding URL (which is usually impossible, but see the RewriteBase directive below for the trick to achieve this) and then initiates a new internal sub-request with the new URL. This restarts processing of the API phases.

Again mod\_rewrite tries hard to make this complicated step totally transparent to the user, but you should remember here: While URL manipulations in per-server context are really fast and efficient, per-directory rewrites are slow and inefficient due to this chicken and egg problem. But on the other hand this is the only way mod\_rewrite can provide (locally restricted) URL manipulations to the average user.

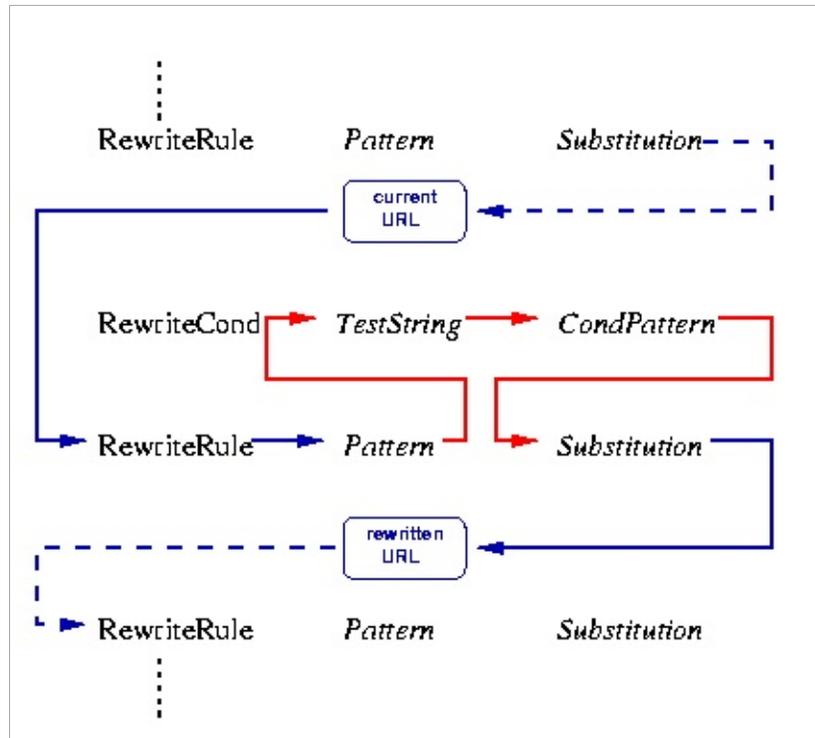
Don't forget these two points!



## URL Rewriting

Now when `mod_rewrite` is triggered in these two API phases, it reads the configured rulesets from its configuration structure (which itself was either created on startup for per-server context or during the directory walk of the Apache kernel for per-directory context). Then the URL rewriting engine is started with the contained ruleset (one or more rules together with their conditions). The operation of the URL rewriting engine itself is exactly the same for both configuration contexts. Only the final result processing is different.

The order of rules in the ruleset is important because the rewriting engine processes them in a special (and not very obvious) order. The rule is this: The rewriting engine loops through the ruleset rule by rule (`RewriteRule` directives) and when a particular rule matches it optionally loops through existing corresponding conditions (`RewriteCond` directives). For historical reasons the conditions are given first, and so the control flow is a little bit long-winded. See Figure 1 for more details.



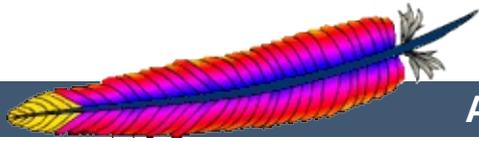
**Figure**

**1:** The control flow through the rewriting ruleset

As you can see, first the URL is matched against the *Pattern* of each rule. When it fails `mod_rewrite` immediately stops processing this rule and continues with the next rule. If the *Pattern* matches, `mod_rewrite` looks for corresponding rule conditions. If none are present, it just substitutes the URL with a new value which is constructed from the string *Substitution* and goes on with its rule-looping. But if conditions exist, it starts an inner loop for processing them in the order that they are listed. For conditions the logic is different: we don't match a pattern against the current URL. Instead we first create a string *TestString* by expanding variables, back-references, map lookups, *etc.* and then we try to match *CondPattern* against it. If the pattern doesn't match, the complete set of conditions and the corresponding rule fails. If the pattern matches, then the next condition is processed until no more conditions are available. If all conditions match, processing is continued with the substitution of the URL with *Substitution*.

Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [FAQ](#)

## Frequently Asked Questions

The latest version of this FAQ is always available from the main Apache web site, at <<http://httpd.apache.org/docs/2.0/faq/>>.

Since Apache 2.0 is quite new, we don't yet know what the *Frequently Asked Questions* will be. While this section fills up, you should also consult the [Apache 1.3 FAQ](#) to see if your question is answered there.



---

## Support

What do I do when I have problems?

## Error Messages

What does this error message mean?



- ["Why can't I ...? Why won't ... work?" What to do in case of problems](#)
- [Whom do I contact for support?](#)

## "Why can't I ...? Why won't ... work?" What to do in case of problems

If you are having trouble with your Apache server software, you should take the following steps:

### Check the errorlog!

Apache tries to be helpful when it encounters a problem. In many cases, it will provide some details by writing one or messages to the server error log. Sometimes this is enough for you to diagnose & fix the problem yourself (such as file permissions or the like). The default location of the error log is `/usr/local/apache2/logs/error_log`, but see the [ErrorLog](#) directive in your config files for the location on your server.

### Check the [FAQ!](#)

The latest version of the Apache Frequently-Asked Questions list can always be found at the main Apache web site.

### Check the Apache bug database

Most problems that get reported to The Apache Group are recorded in the [bug database](#). **Please** check the existing reports, open **and** closed, before adding one. If you find that your issue has already been reported, please *don't* add a "me, too" report. If the original report isn't closed yet, we suggest that you check it periodically. You might also consider contacting the original submitter, because there may be an email exchange going on about the issue that isn't getting recorded in the database.

## Ask in a user support forum

Apache has an active community of users who are willing to share their knowledge. Participating in this community is usually the best and fastest way to get answers to your questions and problems.

[Users mailing list](#)

[#httpd](#) on [Freenode IRC](#) is available for user support issues.

USENET newsgroups:

- [comp.infosystems.www.servers.unix](#) [[news](#)] [[google](#)]
- [comp.infosystems.www.servers.ms-windows](#) [[news](#)] [[google](#)]
- [comp.infosystems.www.authoring.cgi](#) [[news](#)] [[google](#)]

## If all else fails, report the problem in the bug database

If you've gone through those steps above that are appropriate and have obtained no relief, then please *do* let the httpd developers know about the problem by [logging a bug report](#).

If your problem involves the server crashing and generating a core dump, please include a backtrace (if possible). As an example,

```
# cd ServerRoot
# dbx httpd core
(dbx) where
```

(Substitute the appropriate locations for your *ServerRoot* and your *httpd* and *core* files. You may have to use *gdb* instead of *dbx*.)

## Whom do I contact for support?

With several million users and fewer than forty volunteer developers, we cannot provide personal support for Apache. For free support, we suggest participating in a [user forum](#).



- [Invalid argument: core\\_output\\_filter: writing data to the network](#)
- [AcceptEx failed](#)
- [Premature end of script headers](#)

## **Invalid argument: core\_output\_filter: writing data to the network**

Apache uses the `sendfile` syscall on platforms where it is available in order to speed sending of responses. Unfortunately, on some systems, Apache will detect the presence of `sendfile` at compile-time, even when it does not work properly. This happens most frequently when using network or other non-standard file-system.

Symptoms of this problem include the above message in the error log and zero-length responses to non-zero-sized files. The problem generally occurs only for static files, since dynamic content usually does not make use of `sendfile`.

To fix this problem, simply use the [EnableSendfile](#) directive to disable `sendfile` for all or part of your server. Also see the [EnableMMAP](#), which can help with similar problems.

## **AcceptEx Failed**

If you get error messages related to the `AcceptEx` syscall on win32, see the [Win32DisableAcceptEx](#) directive.

## **Premature end of script headers**

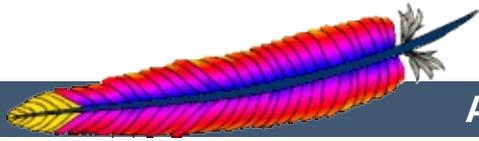
Most problems with CGI scripts result in this message written in the error log together with an `Internal Server Error` delivered to the browser. A guide to helping debug this type of

problem is available in the [CGI tutorial](#).

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#) > [Developer Documentation](#)

## Apache 2.0 Thread Safety Issues

When using any of the threaded mpms in Apache 2.0 it is important that every function called from Apache be thread safe. When linking in 3rd party extensions it can be difficult to determine whether the resulting server will be thread safe. Casual testing generally won't tell you this either as thread safety problems can lead to subtle race conditons that may only show up in certain conditions under heavy load.



## Global and Static Variables

When writing your module or when trying to determine if a module or 3rd party library is thread safe there are some common things to keep in mind.

First, you need to recognize that in a threaded model each individual thread has its own program counter, stack and registers. Local variables live on the stack, so those are fine. You need to watch out for any static or global variables. This doesn't mean that you are absolutely not allowed to use static or global variables. There are times when you actually want something to affect all threads, but generally you need to avoid using them if you want your code to be thread safe.

In the case where you have a global variable that needs to be global and accessed by all threads, be very careful when you update it. If, for example, it is an incrementing counter, you need to atomically increment it to avoid race conditions with other threads. You do this using a mutex (mutual exclusion). Lock the mutex, read the current value, increment it and write it back and then unlock the mutex. Any other thread that wants to modify the value has to first check the mutex and block until it is cleared.

If you are using [APR](#), have a look at the `apr_atomic_*` functions and the `apr_thread_mutex_*` functions.



This is a common global variable that holds the error number of the last error that occurred. If one thread calls a low-level function that sets `errno` and then another thread checks it, we are bleeding error numbers from one thread into another. To solve this, make sure your module or library defines `_REENTRANT` or is compiled with `-D_REENTRANT`. This will make `errno` a per-thread variable and should hopefully be transparent to the code. It does this by doing something like this:

```
#define errno (*(__errno_location()))
```

which means that accessing `errno` will call `__errno_location()` which is provided by the `libc`. Setting `_REENTRANT` also forces redefinition of some other functions to their `*_r` equivalents and sometimes changes the common `getc/putc` macros into safer function calls. Check your `libc` documentation for specifics. Instead of, or in addition to `_REENTRANT` the symbols that may affect this are `_POSIX_C_SOURCE`, `_THREAD_SAFE`, `_SVID_SOURCE`, and `_BSD_SOURCE`.



---

Not only do things have to be thread safe, but they also have to be reentrant. `strtok()` is an obvious one. You call it the first time with your delimiter which it then remembers and on each subsequent call it returns the next token. Obviously if multiple threads are calling it you will have a problem. Most systems have a reentrant version of the function called `strtok_r()` where you pass in an extra argument which contains an allocated `char *` which the function will use instead of its own static storage for maintaining the tokenizing state. If you are using [APR](#) you can use `apr_strtok()`.

`crypt()` is another function that tends to not be reentrant, so if you run across calls to that function in a library, watch out. On some systems it is reentrant though, so it is not always a problem. If your system has `crypt_r()` chances are you should be using that, or if possible simply avoid the whole mess by using `md5` instead.



The following is a list of common libraries that are used by 3rd party Apache modules. You can check to see if your module is using a potentially unsafe library by using tools such as `ldd(1)` and `nm(1)`. For [PHP](#), for example, try this:

```
% ldd libphp4.so
libsablot.so.0 => /usr/local/lib/libsablot.so.0 (0x401f6000)
libexpat.so.0 => /usr/lib/libexpat.so.0 (0x402da000)
libsnmp.so.0 => /usr/lib/libsnmp.so.0 (0x402f9000)
libpdf.so.1 => /usr/local/lib/libpdf.so.1 (0x40353000)
libz.so.1 => /usr/lib/libz.so.1 (0x403e2000)
libpng.so.2 => /usr/lib/libpng.so.2 (0x403f0000)
libmysqlclient.so.11 => /usr/lib/libmysqlclient.so.11
(0x40411000)
libming.so => /usr/lib/libming.so (0x40449000)
libm.so.6 => /lib/libm.so.6 (0x40487000)
libfreetype.so.6 => /usr/lib/libfreetype.so.6 (0x404a8000)
libjpeg.so.62 => /usr/lib/libjpeg.so.62 (0x404e7000)
libcrypt.so.1 => /lib/libcrypt.so.1 (0x40505000)
libssl.so.2 => /lib/libssl.so.2 (0x40532000)
libcrypto.so.2 => /lib/libcrypto.so.2 (0x40560000)
libresolv.so.2 => /lib/libresolv.so.2 (0x40624000)
libdl.so.2 => /lib/libdl.so.2 (0x40634000)
libnsl.so.1 => /lib/libnsl.so.1 (0x40637000)
libc.so.6 => /lib/libc.so.6 (0x4064b000)
/lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x80000000)
```

In addition to these libraries you will need to have a look at any libraries linked statically into the module. You can use `nm(1)` to look for individual symbols in the module.



Please drop a note to [dev@httpd.apache.org](mailto:dev@httpd.apache.org) if you have additions or corrections to this list.

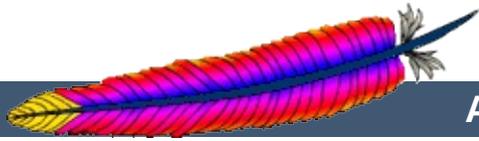
Library	Version	Thread Safe?	Notes
<a href="#">ASpell/PSpell</a>		?	
<a href="#">Berkeley DB</a>	3.x, 4.x	Yes	Be careful about sharing a connection across threads.
<a href="#">bzip2</a>		Yes	Both low-level and high-level APIs are thread-safe. However, high-level API requires <code>errno</code> to be thread-safe.
<a href="#">cdb</a>		?	
<a href="#">C-Client</a>		Perhaps	c-client uses <code>strtok()</code> and <code>gethostbyname()</code> which are not thread-safe on most implementations. c-client's static data is shared across threads. If <code>strtok()</code> and <code>gethostbyname()</code> are thread-safe, c-client <i>may</i> be thread-safe.
<a href="#">cpdflib</a>		?	
<a href="#">libcrypt</a>		?	
<a href="#">Expat</a>		Yes	Need a separate parser instance per thread.
<a href="#">FreeTDS</a>		?	
<a href="#">FreeType</a>		?	
<a href="#">GD 1.8.x</a>		?	
<a href="#">GD 2.0.x</a>		?	
<a href="#">gdbm</a>		No	Errors returned via a static <code>gdbm_errno</code> .
<a href="#">ImageMagick</a>	5.2.2	Yes	ImageMagick docs claim it is thread-safe starting in version 5.2.2 (see <a href="#">Change log</a> ).
<a href="#">Imlib2</a>		?	
<a href="#">libjpeg</a>	v6b	?	

<a href="#">libmysqlclient</a>		Yes	Use mysqlclient_r library variant for thread safety. For more information, please see <a href="http://www.mysql.com/doc/en/Thread-Safe-Client-Library.html">http://www.mysql.com/doc/en/Thread-Safe-Client-Library.html</a>
<a href="#">Ming</a>	0.2a	?	
<a href="#">Net-SNMP</a>	5.0.x	?	
<a href="#">OpenLDAP</a>	2.1.x	Yes	Use ldap_r library variant to ensure thread safety
<a href="#">OpenSSL</a>	0.9.6g	Yes	Requires proper usage of CRYPTO_THREADID. See <a href="http://www.openssl.org/docs/crypto/crypto_threadid.html">http://www.openssl.org/docs/crypto/crypto_threadid.html</a> for details. Also see CRYPTO_set_locking_callback and CRYPTO_set_id_callback
<a href="#">liboci8 (Oracle 8+)</a>	8.x,9.x	?	
<a href="#">pdflib</a>	5.0.x	Yes	PDFLib docs claim it is thread safe. However, the docs indicate it has been partially threaded since version V1.91: <a href="http://www.pdflib.com/products/pdflib.html">http://www.pdflib.com/products/pdflib.html</a>
<a href="#">libpng</a>	1.0.x	?	
<a href="#">libpng</a>	1.2.x	?	
<a href="#">libpq (PostgreSQL)</a>	7.x	Yes	Don't share connections across threads. Avoid out for crypt ( ) calls
<a href="#">Sablotron</a>	0.95	?	
<a href="#">zlib</a>	1.1.4	Yes	Relies upon thread-safe zalloc and zfree. Default is to use libc's calloc/free and free. See <a href="http://www.zlib.net/">http://www.zlib.net/</a> for details.

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)



## Apache HTTP Server Version 2.0

[Apache](#) > [HTTP Server](#) > [Documentation](#) > [Version 2.0](#)

# Apache mod\_rewrite Introduction

This document supplements the [mod\\_rewrite reference documentation](#). It describes the basic concepts necessary for use of [mod\\_rewrite](#). Other documents go into greater detail, but this doc should help the beginner get their feet wet.

## See also

[Module documentation](#)

[Technical details](#)

[Practical solutions to common problems](#)



---

The Apache module `mod_rewrite` is a very powerful and sophisticated module which provides a way to do URL manipulations. With it, you can do nearly all types of URL rewriting that you may need. It is, however, somewhat complex, and may be intimidating to the beginner. There is also a tendency to treat rewrite rules as magic incantation, using them without actually understanding what they do.

This document attempts to give sufficient background so that what follows is understood, rather than just copied blindly.



## Regular Expressions

---

### Basic regex building blocks



## REWRITERULE BASICS

Basic anatomy of a RewriteRule, with exhaustively annotated simple examples.



Discussion of the flags to RewriteRule, and when and why one might use them.



## REWRITE CONDITIONS

Discussion of RewriteCond, looping, and other related concepts.



## RewriteMap

Discussion of RewriteMap, including simple, but heavily annotated, examples.



---

Discussion of the differences between rewrite rules in httpd.conf and in .htaccess files.



## Environment Variables

This module keeps track of two additional (non-standard) CGI/SSI environment variables named `SCRIPT_URL` and `SCRIPT_URI`. These contain the *logical* Web-view to the current resource, while the standard CGI/SSI variables `SCRIPT_NAME` and `SCRIPT_FILENAME` contain the *physical* System-view.

Notice: These variables hold the URI/URL *as they were initially requested, i.e., before* any rewriting. This is important because the rewriting process is primarily used to rewrite logical URLs to physical pathnames.

### Example

```
SCRIPT_NAME=/sw/lib/w3s/tree/global/u/rse/.www/index.html
SCRIPT_FILENAME=/u/rse/.www/index.html
SCRIPT_URL=/u/rse/
SCRIPT_URI=http://en1.engelschall.com/u/rse/
```

---

Copyright 2013 The Apache Software Foundation.  
Licensed under the [Apache License, Version 2.0](#).

[Modules](#) | [Directives](#) | [FAQ](#) | [Glossary](#) | [Sitemap](#)