Configuration

These rules apply to all of HTTPd's configuration files.

- Case insensitive

  Except where pathnames are involved, these files are not case sensitive.

- Comment lines begin with #

  Lines which should be ignored begin with #, the hash sign. This must be the first character on the line. Comments must be on a line by themselves.

- One directive per line

  Each line of these files consists of:

  ```
  Directive data [data2 ... datan]
  ```

  `Directive` is a keyword HTTPd recognizes, followed by whitespace. `data` is specific to the directive. Any additional data entries should be separated by whitespace.

- Extra whitespace is ignored

  You can put extra spaces or tabs between `Directive` and `data`. To embed a space in `data` without separating it from any subsequent arguments use a \ character before the space.

Back to Configuration Overview[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / 9-11-95

[1] Configure.html
[2] ../FeedBack.html

Configuration

Configuration means customizing HTTPd's configuration files to reflect your system and how HTTPd should act on your system.

Please read this general information[1] which applies to all of HTTPd's configuration files.

Also, after changing the main configuration files of your server, you need to restart[2] the server for the changes to take effect.

There are three main configuration files you must edit.

Server Configuration[3]
: Technical aspects of server operation.

Resource Configuration[4]
: Document and script locations and aliases.

Access Control[5]
: Control which clients access certain directories, and which features are available in different directories.

Back to Server Administration[6]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[7] / Last Modified 8−29−95

1 ../ConfigureInfo.html
2 KillIt.html
3 ../httpd/Overview.html
4 ../srm/Overview.html
5 ../access/Overview.html
6 Overview.html
7 ../../FeedBack.html

Managing the Standalone Daemon

If you are using a ServerType[1] of `inetd`, you do not need to read these instructions.

If you make any changes to HTTPd's configuration files, you must restart the server. To do so, you will need to know where the server has logged its process id, given by the server configuration directive PidFile[2].

If `pidfile` is the setting of PidFile for your server, to restart the daemon execute the following on the command line:

```
kill -1 `cat pidfile`
```

or

```
kill -HUP `cat pidfile`
```

This will restart the daemon.

You should now check to see if the daemon restarted successfully. Check the last line of the server's error log, given by the server configuration directive ErrorLogFile[3], and make sure it says `HTTPd: successful restart`.

Alternatively, you may use the `ps` command to `grep` for the process id of the daemon.

If you ever need to terminate the server, use the following command:

```
kill `cat pidfile`
```

Node: **Don't** use `kill -9 `cat pidfile`` to kill the server because `kill -9` will kill the parents but will not kill all the children on some systems.

Return to the Administration Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 8−29−95

1 ../httpd/ServerType.html
2 ../httpd/PidFile.html
3 ../httpd/ErrorLog.html
4 Overview.html
5 ../../FeedBack.html

Server Configuration File

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

When you unpacked HTTPd, there should have been a directory called `conf`. In there is a file called `httpd.conf-dist`. This is a template for your server configuration file. We suggest you use this file only for reference, and edit the file `conf/httpd.conf` as your server configuration file.

Here is a list of the directives this file recognizes. Following the link will give you usage information, default values, and examples.

- ServerType[9]
- Port[10]
- User[11]
- Group[12]
- ServerAdmin[13]
- ServerRoot[14]
- ServerName[15]
- StartServers[16]
- MaxServers[17]
- TimeOut[18]
- AccessConfig[19]
- ResourceConfig[20]
- TypesConfig[21]
- IdentityCheck[22]
- BindAddress[23]
- Host[24]
- VirtualHost[25]
- SRMOptions[26]
- Logging
  - ErrorLog[27]
  - TransferLog[28]

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ServerType.html
10 Port.html
11 User.html
12 Group.html
13 ServerAdmin.html
14 ServerRoot.html
15 ServerName.html
16 StartServer.html
17 MaxServer.html
18 TimeOut.html
19 AccessConfig.html
20 ResourceConfig.html
21 TypesConfig.html
22 IdentityCheck.html
23 BindAddress.html
24 Host.html
25 VirtualHost.html
26 SRMOptions.html
27 ErrorLog.html
28 TransferLog.html

  - AgentLog[29]
  - RefererLog[30]
  - RefererIgnore[31]
  - PidFile[32]
  - LogDirGroupWriteOk[33]
  - LogDirOtherWriteOk[34]
  - LogOptions[35]
- KeepAlive
  - KeepAlive[36]
  - KeepAliveTimeout[37]
  - MaxKeepAliveRequests[38]
- Security Related
  - AssumeDigestSupport[39]

Return to Configuration Instructions[40]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[41] / 10−02−95

29 AgentLog.html
30 RefererLog.html
31 RefererIgnore.html
32 PidFile.html
33 LogDirGroupWriteOk.html
34 LogDirOtherWriteOk.html
35 LogOptions.html
36 KeepAlive.html
37 KeepAliveTimeout.html
38 MaxKeepAliveRequests.html
39 AssumeDigestSupport.html
40 ../Configure.html
41 ../../FeedBack.html

AccessConfig Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The AccessConfig directive gives HTTPd the location of the global access configuration file[9].

Syntax

`AccessConfig file`

`file` is the name of the global access configuration file. It is either a full pathname, or a partial pathname relative to ServerRoot[10].

Only one AccessConfig directive is allowed in the configuration file.

Virtual Host

Access Information can be different across different hosts, but the configuration file is the same. For non−overlapping directory trees, you can merely specify different constraints for different directories. For overlapping trees, you must have different AccessFileNames[11] for each host and cannot use the global configuration.

Default

If you do not specify an AccessConfig, HTTPd assumes:

`AccessConfig conf/access.conf`

Compile Option: `ACCESS_CONFIG_FILE`

Examples

`AccessConfig conf/access-global`

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../access/Overview.html
10 ServerRoot.html
11 ../srm/AccessFileName.html

HTTPd looks for access configuration in the file conf/access−global in the ServerRoot[12] directory.

`AccessConfig /httpd/admin/access`

HTTPd looks for access configuration in the file `/httpd/admin/access`.

Return to Server Configuration File Overview[13]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[14] / Last Modified 7−12−95

12 ServerRoot.html
13 Overview.html
14 mailto:httpd@ncsa.uiuc.edu

AgentLog Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

The AgentLog directive tells HTTPd where to record client agent software. The UserAgent header, if present, is recorded in the format sent by the client. If LogOptions[9] is set to Combined, this directive does nothing.

This is for statistical purposes and the tracing of protocol violations. It should be included by all clients. The first white space delimited word must be the software product name, with an optional slash and version designator. Other products which form part of the user agent may be put as separate words.

A User Agent Log Example

The timestamp is the same from the access_log.

```
[02/Oct/1995:12:00:28] SPRY_Mosaic/v7.36 (Windows 16-bit) SPRY_package/v4.00
[02/Oct/1995:12:00:28] Mozilla/1.1N (X11; I; SunOS 5.3 sun4c)
[02/Oct/1995:12:00:29] Mozilla/1.0N (Windows)
[02/Oct/1995:12:00:29] Mozilla/1.22 (Windows; I; 16bit)
[02/Oct/1995:12:00:29] NCSA Mosaic for the X Window System/2.4  libwww/2.12 mod
[02/Oct/1995:12:00:30] Mozilla/1.0N (Windows)
[02/Oct/1995:12:00:30] IBM WebExplorer DLL /v1.02  via proxy gateway  CERN-HTTP
[02/Oct/1995:12:00:31] NCSA Mosaic for the X Window System/2.4  libwww/2.12 mod
[02/Oct/1995:12:00:32] MacWeb/1.00ALPHA3  libwww/2.17
```

Syntax

`AgentLog file`

`file` is the name of the file to which user agents will be logged. It is either a full pathname, or a partial pathname relative to ServerRoot[10].

Virtual Host

This directive is valid in the virtual host section.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 LogOptions.html
10 ServerRoot.html

Default

If you do not specify an AgentLog, HTTPd assumes:

`AgentLog logs/agent_log`

Compile Option: `DEFAULT_AGENTLOG`

Examples

`AgentLog logs/agents`

This logs user agents to the file logs/agents in the ServerRoot[11] directory.

`AgentLog /tmp/httpd-agents`

This logs user agents to the file `/tmp/httpd-agents`.

`AgentLog /dev/null`

This effectively turns off agent logging.

Return to Server Configuration File Overview[12]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[13] / 10–02–95

11 ServerRoot.html
12 Overview.html
13 ../../FeedBack.html

Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

Syntax

Virtual Host

Default

Compile Option:

Examples

Return to Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] / 10–02–95

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html
10 ../../FeedBack.html

BindAddress Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

This is the second method for providing virtual host support. This allows for greater flexibility in configuration specific to each host, but is also more wasteful of resources. This method works by telling the server to only listen to a single address. This means that you need a separate copy of the server running for each virtual host address.

Syntax

`BindAddress` *`servername`*

Where *servername* is a DNS name that maps to a single IP address, or an IP address.

Virtual Host

This directive is incompatible with VirtualHost configuration. You cannot run both a BindAddress server and a VirtualHost server on the same machine.

Default

The Default is `INADDR_ANY`, which is listen to all addresses.

Examples

`BindAddress 127.0.0.1`

This would cause the server to only respond to connects to localhost.

`BindAddress *`

This would cause the server to respond to connects to any address. This is the default.

---

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/

Return to Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] / 10−02−95

9 Overview.html
10 ../../FeedBack.html

ErrorLog Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The ErrorLog directive sets the file to which HTTPd will log errors and status messages it encounters. It currently logs the following:

- Startup Message, Restart Message, and Halt Message
- Debug Dumps (from USR1 signal)
- Clients which time out and abort
- Scripts which produce no output (malformed header from script)
- .htaccess[9] files which attempt to override things they do not have permission to
- Server bugs which produce a segmentation violation or bus error
- User Authentication[10] configuration problems
- File doesn't exist errors
- Resource Limit errors

Syntax

`ErrorLog file`

`file` is the name of the file to which errors will be logged. It is either a full pathname, or a partial pathname relative to ServerRoot[11].

Virtual Host

This directive is valid in the virtual host section.

Default

If you do not specify an ErrorLog, HTTPd assumes:

`ErrorLog logs/error_log`

Compile Option: `DEFAULT_ERRORLOG`

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../access/Overview.html
10 ../access/Overview.html#user_auth
11 ServerRoot.html

Examples

`ErrorLog logs/errors`

This logs errors to the file logs/errors in the ServerRoot[12] directory.

`ErrorLog /tmp/httpd-errors`

This logs errors to the file `/tmp/httpd-errors`.

`ErrorLog /dev/null`

This effectively turns off error logging.

Return to Server Configuration File Overview[13]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[14] / 10-02-95

12 ServerRoot.html
13 Overview.html
14 ../../FeedBack.html

Group Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The Group directive sets which group id the server will answer requests as.

In order to use this directive, the standalone server must be initially run as `root`.

This directive does not mean that the original daemon process will run as the given group, it means that the children which answer requests run as the given group.

This directive is only applicable if you are using a ServerType[9] of `standalone`.

Syntax

`Group id`

`id` is one of:

- A name

  Refer to the given group by name.

- # followed by a group number

  Refer to a group by its number.

Some systems require that this `id` must be a valid group found in `/etc/group`, or else a runtime server error will occur.

HPUX Systems cannot use nobody as a valid group.

Virtual Host

Because of security limitations of the Unix `(TM)` security model, this directive can only be set once for the entire server.

Default

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ServerType.html

If you do not specify a Group, HTTPd assumes:

`Group #-1`

Compile Option: `DEFAULT_GROUP`

Examples

`Group nogroup`

This will cause the server to run as group `nogroup`.

`Group #65536`

This will cause the server to run as group number 65536.

Return to Server Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10−02−95

10 Overview.html
11 ../../FeedBack.html

IdentityCheck Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

This directive enables RFC931 compliant logging of the remote user name for sites which run identd or something similar. This information is logged in the access_log[9]. It should not be trusted in any way except for rudimentary usage tracking.

If you do not plan to use this information, keep this directive set to `off`. Your network administrator will thank you later.

Syntax

`IdentityCheck setting`

`setting` is either `on` or `off`.

Only one IdentityCheck directive should appear in the configuration file.

Virtual Host

This directive is not valid in the virtual host section because this information is determined before the server knows which host is being accessed.

Default

If no IdentityCheck directive is present, httpd assumes:

`IdentityCheck off`

Compile Option: `DEFAULT_RFC931`

Example

`IdentityCheck on`

Causes the server to check the identity of each person accessing it.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 TransferLog.html

Return to Server Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10−02−95

10 Overview.html
11 ../../FeedBack.html

KeepAlive Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

In normal HTTP transactions, the browser opens up a separate connection to the server for everything the browser wants. So, if you have a page with 5 inline images, the browser will use separate connections to get the page and each inline image. With a browser which supports KeepAlive, the browsers requests the documents one at a time through the same connection. This eliminates the start up time and cost of each connection.

Syntax

KeepAlive is either `On` or `Off`

Virtual Host

This directive is not valid in VirtualHost configuration.

Default

The default is Off.

Compile Option: `DEFAULT_ALLOW_KEEPALIVE`

Examples

`KeepAlive On`

Turns on KeepAlive support.

Return to Configuration File Overview[9]

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] */ 10–02–95*

10 ../../FeedBack.html

KeepAliveTimeout Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

This directive specifies the maximum amount of time the server will wait between requests on an open KeepAlive connection.

Syntax

```
KeepAliveTimeout #num
```

Where #num is number of seconds before the server times out and closes the connection.

Virtual Host

This directive is not valid in VirtualHost configuration.

Default

The default is 10 seconds.

Compile Option: `DEFAULT_KEEPALIVE_TIMEOUT`

Examples

```
KeepAliveTimeout 60
```

This will allow the server to keep the connection open for a minute between requests.

Return to Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] / 10−02−95

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html
10 ../../FeedBack.html

NCSA HTTPd Directive

Top[1] *– Docs*[2] *– General*[3] *– Server*[4] *– Resource*[5] *– Access*[6] *– FAQ*[7] *– Tutorials*[8]

Purpose

Syntax

Virtual Host

Default

Compile Option:

Examples

Return to Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] */ 10−02−95*

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html
10 ../../FeedBack.html

NCSA HTTPd Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

Syntax

Virtual Host

Default

Compile Option:

Examples

Return to Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] / 10−02−95

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html
10 ../../FeedBack.html

LogOptions Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The LogOptions directive allows you to switch between the Common Log File (CLF) format with extra information going to separate files, or a somewhat extended CLF format in which the extra information is logged at the end of the normal CLF information. It also allows you to change various other aspects of server logging.

Some Log Analysers won't accept the combined format, so you might want to check first.

Example Combined Log Format

```
lothlorien.ncsa.uiuc.edu - - [19/Sep/1995:15:08:07 -0500] "GET / HTTP/1.0" 200
lothlorien.ncsa.uiuc.edu - - [19/Sep/1995:15:08:07 -0500] "GET /images/httpd.gi
lothlorien.ncsa.uiuc.edu - - [19/Sep/1995:15:08:07 -0500] "GET /images/mosaic-h
lothlorien.ncsa.uiuc.edu - - [19/Sep/1995:15:19:07 -0500] "GET /docs/Overview.h
lothlorien.ncsa.uiuc.edu - - [19/Sep/1995:15:19:07 -0500] "GET /images/icon.gif
```

Note the extra Referer field and the User agent field on the end

Example of CLF with Servername

```
lothlorien.ncsa.uiuc.edu - - [06/Oct/1995:13:51:23 -0500] "GET /beta-1.5/howto/
lothlorien.ncsa.uiuc.edu - - [06/Oct/1995:13:51:34 -0500] "GET /beta-1.5/patche
lothlorien.ncsa.uiuc.edu - - [06/Oct/1995:13:51:35 -0500] "GET /icons/blank.xbm
lothlorien.ncsa.uiuc.edu - - [06/Oct/1995:13:51:35 -0500] "GET /icons/menu.gif
lothlorien.ncsa.uiuc.edu - - [06/Oct/1995:13:51:35 -0500] "GET /icons/text.gif
```

Syntax

`LogOptions opt1 opt2`

Where `optn` is:

- `Combined`

  With this LogOption, the server doesn't open a RefererLog[9] or AgentLog[10]. The information normally logged to those files is now logged in the TransferLog[11].

[1] ../../../
[2] ../../
[3] ../ConfigureInfo.html
[4] ../httpd/Overview.html
[5] ../srm/Overview.html
[6] ../access/Overview.html
[7] ../../FAQ.html
[8] ../../tutorials/
[9] RefererLog.html
[10] AgentLog.html
[11] TransferLog.html

- `Separate`

  With this LogOption, the logging behavior is reminiscent of NCSA HTTPd 1.4 with separate log files for RefererLog[12] and AgentLog[13].

- `Servername`

  With this LogOption, the name which the server used is logged in the TransferLog[14]. If you are using VirtualHost[15] support, this will be the name of VirtualHost.

- `Date`

  With this LogOption, the Date code section of the TransferLog[16] will be prepended to each line of the RefererLog[17] and AgentLog[18] to help with correlating the various files. This is only valid with Separate.

Virtual Host

This directive is valid in virtual host configuration, and suggested if you intend to have a lot of virtual hosts as a way of reducing the number of file descriptors required.

Default

The default is Separate.

Examples

`LogOptions Separate`

This will have separate log files.

`LogOptions Combined ServerName`

This will make a single log file with as much information as possible.

Return to Configuration File Overview[19]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[20] / 10–02–95

[12] RefererLog.html
[13] AgentLog.html
[14] TransferLog.html
[15] VirtualHost.html
[16] TransferLog.html
[17] RefererLog.html
[18] AgentLog.html
[19] Overview.html
[20] ../../FeedBack.html

MaxKeepAliveRequests Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

This directive specifies how many requests the client can make through the connection before the server closes it.

Syntax

```
MaxKeepAliveRequests #num
```

Where #num is the number of requests the server will fulfill.
A #num of 0 implies no maximum, and the server will continue the KeepAlive transaction until the KeepAliveTimeout[9] or until the browser closes the connection.

Virtual Host

This directive is not valid in VirtualHost configuration.

Default

The default is 5.

Compile Option: `DEFAULT_KEEPALIVE_MAXREQUESTS`

Examples

```
MaxKeepAliveRequests 0
```

The server will only close the connection if the timeout value is reached, or a non−KeepAlive capable request is fulfilled (any request for which the server cannot determine before sending the size of the request, including most CGI scripts and Server Side Includes).

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 KeepAliveTimeout.html

Return to Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10−02−95

10 Overview.html
11 ../../FeedBack.html

MaxServers Directive

Top[1] *– Docs*[2] *– General*[3] *– Server*[4] *– Resource*[5] *– Access*[6] *– FAQ*[7] *– Tutorials*[8]

Purpose

The MaxServers directive sets the maximum number of children for the hunt group that NCSA HTTPd will launch to handle increased load from the number made at run time by StartServers[9].

For more information on the server hunt group, see the 1.4 Changes[10].

Syntax

```
MaxServers number
```

Default

If nothing is specified, the server will use 10.

Compile Option: `DEFAULT_MAX_DAEMON`

Return to Server Configuration File Overview[11]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[12] */ Last Modified 7−12−95*

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 StartServer.html
10 ../../Upgrade.html#1.4
11 Overview.html
12 mailto:httpd@ncsa.uiuc.edu

PidFile Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The PidFile directive sets the file to which HTTPd records the daemon process id.

This directive is used only if ServerType[9] is `standalone`.

Syntax

`PidFile file`

`file` is the name of the file to which the process id is logged. It is either a full pathname, or a partial pathname relative to ServerRoot[10].

Only one PidFile directive is allowed in the configuration file.

Virtual Host

This directive is not valid in virtual host configuration.

Default

If you do not specify a PidFile, HTTPd assumes:

`PidFile logs/httpd.pid`

Compile Option: `DEFAULT_PIDLOG`

Examples

`PidFile logs/pid`

This logs the process id to the file logs/pid in the ServerRoot[11] directory.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ServerType.html
10 ServerRoot.html
11 ServerRoot.html

`PidFile /tmp/httpd-pid`

This logs the process id to the file `/tmp/httpd-pid`.

`PidFile /dev/null`

The server does not log its process id.

Return to Server Configuration File Overview[12]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[13] / 10-02-95

12 Overview.html
13 mailto:httpd@ncsa.uiuc.edu

Port Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

The Port directive sets what port HTTPd listens to for clients.

Syntax

```
Port num
```

`num` is a number from 0 – 65536. Most ports below 1024, except port 80, are reserved by the system.

If you want to use a port below 1024 (such as 80, the standard HTTP port), and your ServerType[9] is `standalone`, you will need to run HTTPd as root when starting it up.

Only one Port directive is allowed in the configuration file.

Virtual Host

The server can only listen to one port, so the Port directive is invalid in virtual host configuration.

Default

If you do not specify a Port, HTTPd assumes:

```
Port 80
```

Compile Option: `DEFAULT_PORT`

Examples

```
Port 8080
```

The server would listen to port 8080.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ServerType.html


```
Port 84
```

The server would try to listen to port 84.

Return to Server Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10−02−95

10 Overview.html
11 ../../FeedBack.html

RefererIgnore Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The RefererIgnore directive tells HTTPd to ignore certain RefererLog[9] lines. If the RefererIgnore string is contained in the Referer header, then the line is not logged.

Syntax

`RefererIgnore string1 string2 ... stringn`

`stringn` is the set of characters to look for in the Referer header. If this string is found, then the Referer is not logged. If there is no RefererIgnore directive, then all referer information is logged.

Virtual Host

This directive is valid in virtual host configuration.

Default

If you do not specify a RefererIgnore, HTTPd doesn't ignore logging any RefererLog[10] lines.

Examples

`RefererIgnore www.ncsa.uiuc.edu`

This avoids logging referers from www.ncsa.uiuc.edu

`RefererIgnore news`

This avoids logging referers from news articles.

`RefererIgnore www.cen www.ncsa`

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 RefererLog.html
10 RefererLog.html

This avoids logging referers from either www.ncsa or www.cen

Return to Server Configuration File Overview[11]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[12] / 10−02−95

11 Overview.html
12 ../../FeedBack.html

RefererLog Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The RefererLog directive tells HTTPd where to record referers. The Referer header, if present, is recorded in the following format:

```
[Time Stamp] URI -> Document
```

Where `URI` is the Universal Resource Identifier for the document that references the server, and `Document` is the name of the file being refered to. This is a path relative to DocumentRoot[9]. It is also possible to ignore logging certain referers by using the RefererIgnore[10] directive.

This directive is non–functional under the Combined LogOption[11].

A Referer Example Log

```
[02/Oct/1995:13:03:29] http://www.webcrawler.com/cgi-bin/WebQuery -> /images/ic
[02/Oct/1995:13:03:29] http://www.webcrawler.com/cgi-bin/WebQuery -> /images/ba
[02/Oct/1995:13:03:34] http://www.lysator.liu.se:7500/mud/faq/faq1.html -> /ftp
[02/Oct/1995:13:04:25] http://rhine.ece.utexas.edu/~shishir/ -> /docs/tutorials
[02/Oct/1995:13:04:30] http://humnet.humberc.on.ca/ -> /archie.html/
```

Syntax

```
RefererLog file
```

`file` is the name of the file to which referers will be logged. It is either a full pathname, or a partial pathname relative to ServerRoot[12].

Virtual Host

This directive is valid in virtual host configuration.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../srm/DocumentRoot.html
10 RefererIgnore.html
11 LogOptions.html
12 ServerRoot.html

Default

If you do not specify a RefererLog, HTTPd assumes:

```
RefererLog logs/referer_log
```

Compile Option: `DEFAULT_REFERERLOG`

Examples

```
RefererLog logs/referers
```

This logs referers to the file logs/referers in the ServerRoot[13] directory.

```
RefererLog /tmp/httpd-referers
```

This logs referers to the file `/tmp/httpd-referers`.

```
RefererLog /dev/null
```

This effectively turns off referer logging.

Return to Server Configuration File Overview[14]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[15] / Last Modified 7–12–95

13 ServerRoot.html
14 Overview.html
15 mailto:httpd@ncsa.uiuc.edu

ResourceConfig Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The ResourceConfig directive gives HTTPd the location of the resource configuration file[9].

Syntax

`ResourceConfig file`

`file` is the name of the resource configuration file. It is either a full pathname, or a partial pathname relative to ServerRoot[10].

Virtual Host

This directive is valid in virtual host configuration.

Default

If you do not specify a ResourceConfig, HTTPd assumes:

`ResourceConfig conf/srm.conf`

Compile Option: `RESOURCE_CONFIG_FILE`

Examples

`ResourceConfig conf/resources`

HTTPd looks for resource configuration in conf/resources in the ServerRoot[11] directory.

`ResourceConfig /httpd/admin/resources`

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../srm/Overview.html
10 ServerRoot.html
11 ServerRoot.html

HTTPd looks for resource configuration in the file `/httpd/admin/resources`.

Return to Server Configuration File Overview[12]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[13] / 10-02-95

12 Overview.html
13 ../../FeedBack.html

SRMOptions

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

This command is mostly for VirtualHost configuration. It allows you to specify Server Resource Management (SRM) configuration in the Server Configuration file. It works by calling the SRM parser on the file after the SRMOption line.

Syntax

```
<SRMOptions>
Directive1 Config1
...
</SRMOptions>
```

Where *Directive1* is a SRM[9] directive. Any number of directives can be used in a SRMOptions section.

Virtual Host

This directive is valid in both the VirtualHost section and the rest of the server configuration file. More than one section is also possible.

Default

It works by default. There is no way to disable it.

Compile Option: *none*

Examples

```
<SRMOptions>
DocumentRoot /www
Alias /icons/ /usr/local/etc/httpd/icons/
</SRMOptions>

<VirtualHost 127.0.0.1 required>
        ServerName localhost
        ServerAdmin yow@wow.net
```

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../srm/Overview.html


```
        <SRMOptions>
                DocumentRoot /www/local
                Alias /icons/ /usr/local/etc/httpd/icons/
                ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-local/
                # Local can use internal imagemap
                AddType text/x-imagemap .map
                AddType application/x-httpd-cgi .cgi
        </SRMOptions>
</VirtualHost>
```

Return to Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 11−06−95

10 Overview.html
11 ../../FeedBack.html

ServerAdmin Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The ServerAdmin directive gives the server your e−mail address. This is so that the server can give people your address to report error conditions.

Syntax

`ServerAdmin address`

Where `address` is an e−mail address.

Virtual Host

This directive is valid in virtual host configuration.

Default

If you do not specify a ServerAdmin, HTTPd assumes nothing and prints no address for reporting errors.

Compile Option: `DEFAULT_ADMIN`

Examples

`ServerAdmin www@widget.com`

Errors will have the address www@widget.com as the person to blame.

Return to Server Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] / Last Modified

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html
10 mailto:httpd@ncsa.uiuc.edu

7−12−95

ServerName Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

The ServerName directive[9] sets the hostname HTTPd should return when creating redirection URLs.

This directive should be used on systems where `gethostbyname` may not work on the local host, or where the hostname returned should be a DNS alias such as `www.widget.com`.

Syntax

```
ServerName FQDN
```

Where `FQDN` is the full hostname (including domain name) to be returned as the server address.

Virtual Host

This directive is valid in virtual host configuration.

Default

If you do not specify a ServerName, HTTPd attempts to retrieve it through system calls.

Example

```
ServerName www.widget.com
```

This would set the server's hostname as www.widget.com, as opposed to monster.widget.com as would be returned from the gethostname system call.

---

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../ConfigureInfo.html#Directive

Return to Server Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10−02−95

10 Overview.html
11 ../../FeedBack.html

ServerRoot Directive

Purpose

The ServerRoot directive sets the directory in which HTTPd lives.

Upon startup, HTTPd expects to find the Server Configuration File as conf/httpd.conf in the ServerRoot directory.

Other Server Configuration directives may use this directory to give relative paths for locations of files.

Syntax

`ServerRoot dir`

Where `dir` is an absolute path of a directory on your server machine.

Only one ServerRoot directive is allowed in the server configuration file.

Virtual Host

This directive is not available for virtual host configuration.

Default

If you do not specify a ServerRoot, HTTPd assumes:

`ServerRoot /usr/local/etc/httpd`

Compile Option: `HTTPD_ROOT`

Examples

`ServerRoot /usr/local/httpd`

This would set ServerRoot to the directory /usr/local/httpd.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/

`ServerRoot /web1/http`

This would set ServerRoot to /web1/http.

Return to Server Configuration File Overview[9]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] / 10−02−95

9 Overview.html
10 ../../FeedBack.html

ServerType Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The ServerType directive sets how the server is executed by the system.

Syntax

`ServerType type`

`type` is one of:

- `inetd`

  To run the server from the system process `inetd`. The system then does all socket and child management.

- `standalone`

  To run the server as a daemon process. The server then does all socket and child management.

Only one ServerType directive is allowed in the configuration file.

Virtual Host

This directive is set for the entire server and cannot be used in the VirtualHost section.

Default

If you do not specify a ServerType, HTTPd assumes:

`ServerType standalone`

We recommend against running the server from inetd. Here's why[9].

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 WhyNotInetd.html

Examples

`ServerType standalone`

`ServerType inetd`

Return to Server Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10−02−95

10 Overview.html
11 ../../FeedBack.html

StartServers Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

The StartServers directive sets how many children NCSA HTTPd will launch at run time. NCSA HTTPd will continue launching children up to MaxServers[9]. For more information on the server hunt group, see the 1.4 Changes[10].

Syntax

`StartServers number`

Virtual Host

This directive is not used by virtual host configuration.

Default

If no number is give, HTTPd will start 5 children.

Compile Option: `DEFAULT_START_DAEMON`

Return to Server Configuration File Overview[11]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[12] / 10−02−95

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 MaxServer.html
10 ../../Upgrade.html#1.4
11 Overview.html
12 ../../FeedBack.html

TimeOut Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The TimeOut directive sets the amount of time the server will wait for a client to send its query once connected, or the maximum amount of time the server will spend waiting for a client to accept information.

Syntax

`TimeOut time`

Where `time` is the amount of time in seconds the server should wait.

Only one TimeOut directive is allowed in the configuration file.

Default

If you do not specify a TimeOut, HTTPd assumes:

`TimeOut 1200`

This is a timeout of twenty minutes.

Compile Option: `DEFAULT_TIMEOUT`

Example

`TimeOut 600`

This sets the timeout to ten minutes. For large files served to slow network clients, you may need every second of it.

Return to Server Configuration File Overview[9]

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 Overview.html

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[10] */ 10–02–95*

10 ../../FeedBack.html

TransferLog Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

The TransferLog directive[9] tells HTTPd where to record client accesses.

The logfile format is as follows. Each line consists of:

```
host rfc931 authuser [DD/Mon/YYYY:hh:mm:ss] "request" ddd bbbb "opt_referer" "o
```

- host: Either the DNS name or the IP number of the remote client
- rfc931: Any information returned by identd for this person, – otherwise.
- authuser: If user sent a userid for authentication, the user name, – otherwise.
- DD: Day
- Mon: Month (calendar name)
- YYYY: Year
- hh: hour (24–hour format, the machine's timezone)
- mm: minutes
- ss: seconds
- request: The first line of the HTTP request as sent by the client.
- ddd: the status code returned by the server, – if not available.
- bbbb: the total number of bytes sent, *not including the HTTP/1.0 header*, – if not available
- opt_referer: the referer field if given and if LogOptions[10] is Combined
- opt_agent: the user agent field if given and if LogOptions[11] is Combined

You can determine the name of the file accessed through `request`.

Syntax

`TransferLog file`

`file` is the name of the file to which transfers will be logged. It is either a full pathname, or a partial pathname relative to ServerRoot[12].

Virtual Host

This directive is valid in virtual host configuration.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../ConfigureInfo.html#Directive
10 LogOptions.html
11 LogOptions.html
12 ServerRoot.html

Default

If you do not specify a TransferLog, HTTPd assumes:

`TransferLog logs/access_log`

Compile Option: `DEFAULT_XFERLOG`

Examples

`TransferLog logs/downloads`

This logs transfers to the file logs/downloads in the ServerRoot[13] directory.

`TransferLog /tmp/httpd-accesses`

This logs transfers to the file `/tmp/httpd-accesses`.

`TransferLog /dev/null`

This effectively turns off transfer logging.

Return to Server Configuration File Overview[14]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[15] / 10–02–95

13 ServerRoot.html
14 Overview.html
15 ../../FeedBack.html

TypesConfig Directive

Top[1] *– Docs*[2] *– General*[3] *– Server*[4] *– Resource*[5] *– Access*[6] *– FAQ*[7] *– Tutorials*[8]

Purpose

The TypesConfig directive gives HTTPd the location of the typing configuration file.

This file is how HTTPd maps filename extensions to MIME types to return to HTTP/1.0 clients. You should not need to edit it. If you really want to, look at the file format[9].

Syntax

`TypesConfig file`

`file` is the name of the types file. It is either a full pathname, or a partial pathname relative to ServerRoot[10].

Only one TypesConfig directive is allowed in the configuration file.

Virtual Host

This directive is not valid for virtual host configuration.

Default

If you do not specify an TypesConfig, HTTPd assumes:

`TypesConfig conf/mime.types`

Compile Option: `TYPES_CONFIG_FILE`

Examples

`TypesConfig conf/mime-types`

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../TypesFormat.html
10 ServerRoot.html

HTTPd looks for types configuration in the file conf/mime–types in the ServerRoot[11] directory.

`TypesConfig /httpd/admin/types-local`

HTTPd looks for types configuration in the file `/httpd/admin/types-local`.

Return to Server Configuration File Overview[12]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[13] / 10–02–95

11 ServerRoot.html
12 Overview.html
13 ../../FeedBack.html

User Directive

Top[1] – *Docs*[2] – *General*[3] – *Server*[4] – *Resource*[5] – *Access*[6] – *FAQ*[7] – *Tutorials*[8]

Purpose

The User directive sets which user id the server will answer requests as.

This directive is only applicable if you are using a ServerType[9] of `standalone`.

In order to use this directive, the standalone server process must be initially started as `root`.

This directive does not mean that the original daemon process will run as the given user, it means that the children which answer requests run as the given user.

Syntax

`User id`

`id` is one of:

- A name

  Refer to the given user by name.

- # followed by a user number

  Refer to a user by their number.

Only one User directive is allowed in the configuration file.

Virtual Host

Due to the limits of the Unix `(TM)` security architecture, the server cannot run as more than one user with the current server model while maintaining sufficient security on other fronts.

Default

If you do not specify a User, HTTPd assumes:

[1] ../../../
[2] ../../
[3] ../ConfigureInfo.html
[4] ../httpd/Overview.html
[5] ../srm/Overview.html
[6] ../access/Overview.html
[7] ../../FAQ.html
[8] ../../tutorials/
[9] ServerType.html

`User #-1`

Compile Option: `DEFAULT_USER`

Examples

`User nobody`

This will cause the server to run as user nobody.

`User #-2`

This will cause the server to run as user number –2.

Return to Server Configuration File Overview[10]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[11] / 10–02–95

[10] Overview.html
[11] ../../FeedBack.html

VirtualHost Directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

This directive allows you to customize the server to respond to different IP addresses in different ways. See the tutorial[9] for more information.

Syntax

```
<VirtualHost servername errorlevel>
...
</VirtualHost>
```

Where :

- *servername*

  This is the DNS name or IP address of this virtual host. The DNS name should map to a single IP address.

- *errorlevel*

  This can be either `Required` or `Optional`. If set to Required, any misconfiguration or problem will cause the server start to fail. If this is set to Optional, the server will attempt to continue even if the configuration is invalid. This is an optional parameter, and if left off, the server will default to `Required`.

- ...

  This can be any number of Server[10] directives, including the SRMOptions[11] directive which allows Resource[12] directives in the server configuration file.

Virtual Host

You cannot use VirtualHost directives within VirtualHost directives. You can use as many VirtualHost directives as you like, up to the limits of your servers memory and operating system.

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../../tutorials/VirtualHost.html
10 Overview.html
11 SRMOptions.html
12 ../srm/Overview.html

Examples

```
<VirtualHost www.uiuc.net>
ServerName www.uiuc.net
ServerAdmin blong@uiuc.net
DocumentRoot /www
</VirtualHost>

<VirtualHost 127.0.0.1 Required>
        ServerName localhost
        ServerAdmin yow@wow.net
        <SRMOptions>
                DocumentRoot /www/local
                Alias /icons/ /usr/local/etc/httpd/icons/
                ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-local/
                # Local can use internal imagemap
                AddType text/x-imagemap .map
                AddType application/x-httpd-cgi .cgi
        </SRMOptions>
</VirtualHost>
```

Return to Configuration File Overview[13]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[14] / 10–02–95

13 Overview.html
14 ../../FeedBack.html

Why Not inetd?

Why should you run the server standalone as opposed to running it from inetd? If your server has a lot of CPU cycles to burn, then there is nothing wrong with running HTTPd from inetd. Most servers don't.

If you run the server from inetd, that means that whenever a request comes in, inetd must first fork a new process, then load the HTTPd binary. Once the HTTPd binary is loaded, HTTPd must then load and parse all of its configuration files (including httpd.conf, access.conf, and mime.types), which is quite a task to be done for each and every request that comes in.

Now, contrast this with running standalone. When HTTPd gets a request, it makes a copy of itself (which requires no loading of a binary since shared text pages are used), and the copy handles the request. The configuration files have already been loaded on startup, and so we don't reload them every time.

In fact, with NCSA HTTPd 1.4 and 1.5, it doesn't even have to make another copy of itself if there is an extra server already running. The parent will just pass off the request to the free child.

Return to ServerType instructions[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / 10−02−95

[1] ServerType.html
[2] ../../FeedBack.html

Setting up the Server Resource Map

When you unpacked HTTPd, there should have been a directory called `conf`. In there is a file called `srm.conf-dist`. This is a template for your resource configuration file. We suggest you use this file only for reference, and edit the file `conf/srm.conf` as your resource configuration file.

Here is a list of the directives[1] this file recognizes. Following the link will give you usage information, default values, and examples.

- DocumentRoot[2]
- UserDir[3]
- AccessFileName[4]
- Virtual URLs
  - Redirect[5]
  - Alias[6]
  - ScriptAlias[7]
- AddType[8]
- AddEncoding[9]
- DefaultType[10]
- Indexing Options
  - DirectoryIndex[11]
  - FancyIndexing[12]
  - DefaultIcon[13]
  - ReadmeName[14]
  - HeaderName[15]
  - AddDescription[16]
  - AddIcon[17]
  - AddIconByType[18]
  - AddIconByEncoding[19]
  - IndexIgnore[20]
  - IndexOptions[21]
- ErrorDocument[22]

---

1 ../ConfigureInfo.html#Directive
2 DocumentRoot.html
3 UserDir.html
4 AccessFileName.html
5 Redirect.html
6 Alias.html
7 ScriptAlias.html
8 AddType.html
9 AddEncoding.html
10 DefaultType.html
11 DirectoryIndex.html
12 FancyIndexing.html
13 DefaultIcon.html
14 ReadmeName.html
15 HeaderName.html
16 AddDescription.html
17 AddIcon.html
18 AddIconByType.html
19 AddIconByEncoding.html
20 IndexIgnore.html
21 IndexOptions.html
22 ErrorDocument.html

Return to Configuration Instructions[23]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[24] / Last Modified 7-12-95

23 ../Configure.html
24 mailto:httpd@ncsa.uiuc.edu

AccessFileName directive

Purpose

When returning a document to a client, the server looks for access control files[1] in the document's directory as well as its parent directories. This directive[2] sets the name of the file HTTPd should look for to find access control files.

Syntax

`AccessFileName file`

Where `file` is a file name.

Only one AccessFileName directive is allowed in the server configuration file.

File

srm.conf

Default

If you do not specify an AccessFileName, HTTPd assumes:

`AccessFileName .htaccess`

Example

`AccessFileName .ncsa-acl`

This would set AccessFileName to .ncsa–acl.

Return to Resource Configuration File Overview[3]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[4] / Last Modified 7–12–95

1 ../access/Overview.html
2 ../ConfigureInfo.html#Directive
3 Overview.html
4 mailto:httpd@ncsa.uiuc.edu

AddDescription directive

Purpose

Tells HTTPd how to describe a file or a file type while generating a directory index.

Syntax

`AddDescription "blah blah blah" fileid`

`fileid` is either a filename extension (like .html), a filename, a full real pathname to a file on your system, or a wildcard pattern[1] to match to filenames.

`blah blah blah` must be surrounded by quotes and is a short (preferably < 1 line) description of the file.

You may use as many AddDescription directives as you wish.

File

srm.conf

Default

There are no default descriptions.

Example

`AddDescription "a great batch of nonsense" fargle.bargle`

A portion of HTTPd's index for fargle.bargle's directory would look something like this:

foople.poople (8000 bytes)
fargle.bargle : a great batch of nonsense(10000 bytes)

Return to Resource Configuration File Overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified

[1] ../MatchExp.html
[2] Overview.html
[3] mailto:httpd@ncsa.uiuc.edu

7–12–95

AddEncoding directive

Purpose

Allows you to specify an encoding type for a document with a given filename extension.

In order to serve encoded documents to clients, the client must support the given encoding method, as well as the HTTP encoding extension.

Syntax

`AddEncoding type extension`

`type` is the encoding type for the document.

`extension` is the filename extension to map to this encoding.

You may use as many AddEncoding directives as you wish.

File

srm.conf

Default

There are no default encodings.

Example

`AddEncoding x-gzip gz`

This would cause any file ending in .gz to be marked as encoded using the x−gzip encoding method.

Reference

See HTTP/1.0 Section 3.5 Content Codings[1]

[1] http://www.w3.org/hypertext/WWW/Protocols/HTTP1.0/draft−ietf−http−spec.html#Content−Codings

Return to Resource Configuration File Overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7−12−95

[2] Overview.html
[3] mailto:httpd@ncsa.uiuc.edu

AddIcon directive

Purpose

Tells HTTPd what kind of an icon to show for a given filetype in a directory index.

You may wish to read the indexing tutorial[1] for more information.

Syntax

`AddIcon icon name1 name2...`

`icon` is a virtual path[2] to an image file which should be shown for files which match the pattern of `names`. Alternatively, this can be a group of the format (alt,icon) where alt is the text tag given for an icon for non−graphical browsers, and icon is a virtual path.

`name` is either `^^DIRECTORY^^` for directories (or `**DIRECTORY**` for backward compatibility), `^^BLANKICON^^`, specifying the blank icon used to format the list properly, a file extension (like .html), a partial filename, a wildcard expression[3], or a complete physical pathname.

You may use as many AddIcon directives as you wish.

File

srm.conf

Default

There are no default icons.

Example

`AddIcon /icons/image.xbm .gif .jpg .xbm`

When HTTPd is indexing and finds a file with the extension .gif, .jpg, or .xbm, it will reference `/icons/image.xbm` as an image to show next to the filename.

`AddIcon /icons/dir.xbm ^^DIRECTORY^^`

If a given directory entry is in fact a subdirectory, `/icons/dir.xbm` will be referenced as an image for that index entry.

[1] ../../tutorials/index−tutorial.html
[2] ../admin/AccessingFiles.html#path
[3] ../MatchExp.html

`AddIcon (SND,/icons/sound.xbm) *.au`

This would reference `/icons/sound.xbm` as the image to show next to any sound file, with the textual ALT tag of SND for non−image clients.

Return to Resource Configuration File Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

[4] Overview.html
[5] mailto:httpd@ncsa.uiuc.edu

AddIconByEncoding directive

Purpose

Tells HTTPd what kind of an icon to show for a given filetype in a directory index.

You may wish to read the indexing tutorial[1] for more information.

Syntax

`AddIconByEncoding icon name1 name2...`

`icon` is a virtual path[2] to an image file which should be shown for files which match the pattern of `names`. Alternatively, this can be a group of the format (alt,icon) where alt is the text tag given for an icon for non−graphical browsers, and icon is a virtual path.

`name` is a wildcard expression[3] specifying the content−encoding for which to display this icon.

You may use as many AddIconByEncoding directives as you wish.

File

srm.conf

Default

There are no default icons.

Example

`AddIconByEncoding /icons/compress.xbm x-compress`

When HTTPd is indexing, if it finds a file compressed with the UNIX compress command, it will display /icons/compress.xbm.

`AddIconByEncoding (CMP,/icons/compress.xbm) x-compress`

Same as above, but the ALT tag would be CMP for these items.

[1] ../../tutorials/index−tutorial.html
[2] ../admin/AccessingFiles.html#path
[3] ../MatchExp.html

Return to Resource Configuration File Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

[4] Overview.html
[5] mailto:httpd@ncsa.uiuc.edu

AddIconByType directive

Purpose

Tells HTTPd what kind of an icon to show for a given filetype in a directory index.

You may wish to read the indexing tutorial[1] for more information.

Syntax

`AddIconByType icon type1 type2...`

`icon` is a virtual path[2] to an image file which should be shown for files which match the pattern of `types`. Alternatively, this can be a group of the format (alt,icon) where alt is a 3−letter text tag given for an icon for non−graphical browsers, and icon is a virtual path.

`name` is a wildcard expression[3] of the MIME types for which to add this icon.

You may use as many AddIconByType directives as you wish.

File

srm.conf

Default

There are no default icons.

Example

`AddIconByType /icons/image.xbm image/*`

When HTTPd is indexing and finds a file which is an image, it will reference `/icons/image.xbm` as an image to show next to the filename.

`AddIconByType (SND,/icons/sound.xbm) audio/*`

This would reference `/icons/sound.xbm` as the image to show next to any sound file, with the textual ALT tag of SND for non−image clients.

[1] ../../tutorials/index−tutorial.html
[2] ../admin/AccessingFiles.html#path
[3] ../MatchExp.html

Return to Resource Configuration File Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

[4] Overview.html
[5] mailto:httpd@ncsa.uiuc.edu

AddType directive

Purpose

Allows you to add entries to the server's default typing information and cause an extension to be a certain type. These directives override any conflicting entries in the TypesConfig[1] file.

Syntax

`AddType type/subtype extension`

`type/subtype` is the MIME–like type for the document.

`extension` is the filename extension to map to this type. This can either be a filename extension, a full pathname, or a file name.

You may use as many AddType directives as you wish.

File

srm.conf

Default

The default types are in the types configuration file.

Example

`AddType text/plain doc`

This would cause any file ending in .doc to be served as type text/plain.

Return to Resource Configuration File Overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7–12–95

1 ../httpd/TypesConfig.html
2 Overview.html
3 mailto:httpd@ncsa.uiuc.edu

Alias directive

Purpose

The Alias directive[1] creates a virtual document (or directory) on your server. Any accesses to it will be satisfied by a different file or directory.

Syntax

`Alias virtual path`

`virtual` is the translated location[2] of the file/directory.

`path` is the full pathname of the file or directory which should be used to fulfill the request.

Several Alias directives may appear in the configuration file.

File

srm.conf

Example

`Alias /images /ftp/pub/images`

This would cause requests for `/images` to be satisfied from the directory `/ftp/pub/images`.

Thus, if someone requested `/images/foo.gif`, the server would return `/ftp/pub/images/foo.gif`.

Return to Resource Configuration File Overview[3]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[4] / Last Modified 7-12-95

[1] ../ConfigureInfo.html#Directive
[2] ../admin/AccessingFiles.html
[3] Overview.html
[4] mailto:httpd@ncsa.uiuc.edu

DefaultIcon directive

Purpose

This directive specifies what icon should be shown in an automatically generated directory listing for a file which has no icon information.

Syntax

`DefaultIcon location`

`location` is the virtual path[1] to the icon on your server.

Only one DefaultIcon directive should appear in the configuration file.

File

srm.conf

Default

If no DefaultIcon is present, HTTPd assumes nothing.

Example

`DefaultIcon /icons/unknown.xbm`

This would cause a file with no icon to be given the icon /icons/unknown.xbm.

Return to Resource Configuration File Overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7−12−95

[1] ../admin/AccessingFiles.html#path
[2] Overview.html
[3] mailto:httpd@ncsa.uiuc.edu

DefaultType directive

Purpose

If HTTPd can't type a file through normal means, it will type it as DefaultType.

Syntax

`DefaultType type/subtype`

`type/subtype` is the MIME–like type.

Only one DefaultType directive should appear in the configuration file.

File

srm.conf

Default

If no DefaultType is present, HTTPd assumes:

`DefaultType text/html`

Example

`DefaultType application/octet-stream`

This would cause a file with an unknown extension to be returned as type application/octet–stream.

Return to Resource Configuration File Overview[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / Last Modified 7–12–95

[1] Overview.html
[2] mailto:httpd@ncsa.uiuc.edu

DirectoryIndex directive

Top[1] – Docs[2] – General[3] – Server[4] – Resource[5] – Access[6] – FAQ[7] – Tutorials[8]

Purpose

When a client requests a directory, HTTPd can return a pre–written index, or generate one from the filesystem. The DirectoryIndex directive[9] sets the file HTTPd should look for as a prewritten index to a given directory.

Syntax

`DirectoryIndex file`

Where `file` is a file name.

Only one DirectoryIndex directive is allowed in the server configuration file.

File

srm.conf

Default

If you do not specify a DirectoryIndex, HTTPd assumes:

`DirectoryIndex index.html`

Example

`DirectoryIndex .index.html`

This would set DirectoryIndex to .index.html. A request for `/dir` would cause the server to look for the file `DocumentRoot`[10]`/dir/.index.html`. If found, the server would send it back to the client. Otherwise, it would create and return an index from the filesystem.

---

1 ../../../
2 ../../
3 ../ConfigureInfo.html
4 ../httpd/Overview.html
5 ../srm/Overview.html
6 ../access/Overview.html
7 ../../FAQ.html
8 ../../tutorials/
9 ../ConfigureInfo.html#Directive
10 DocumentRoot.html

Return to Resource Configuration File Overview[11]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[12] / Last Modified 7–12–95

11 Overview.html
12 mailto:httpd@ncsa.uiuc.edu

DocumentRoot directive

Purpose

The DocumentRoot directive[1] sets the directory from which HTTPd will serve files.

If you need to serve files outside this directory, you can use the Alias[2] directive, or create symbolic links.

This directive affects how you access files on your server[3].

Syntax

`DocumentRoot dir`

Where `dir` is an absolute path of the directory you want documents to be served from.

Only one DocumentRoot directive is allowed in the server configuration file.

File

srm.conf

Default

If you do not specify a DocumentRoot, HTTPd assumes:

`DocumentRoot /usr/local/etc/httpd/htdocs`

Example

`DocumentRoot /home/web`

This would set DocumentRoot to the directory /home/web.

Return to Resource Configuration File Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified

1 ../ConfigureInfo.html#Directive
2 Alias.html
3 ../admin/AccessingFiles.html
4 Overview.html
5 mailto:httpd@ncsa.uiuc.edu

7-12-95

ErrorDocument directive

Purpose

The ErrorDocument directive[1] points the server to a file to send in place of the builtin error message.

Syntax

```
ErrorDocument type filename
```

Where `type` is one of:

- 302 – REDIRECT
- 400 – BAD_REQUEST
- 401 – AUTH_REQUIRED
- 403 – FORBIDDEN
- 404 – NOT_FOUND
- 500 – SERVER_ERROR
- 501 – NOT_IMPLEMENTED

And `filename` is a CGI script[2] or text/html file with full path from document root. CGI scripts launched via these errors have 3 new environment variables, `REDIRECT_REQUEST`, `REDIRECT_URL` and `REDIRECT_STATUS`. They also take as input the error reason of the `form err_string=error_reason.` For an example script, see an example[3].

File

srm.conf

Default

If this directive is left out, the compiled error messages will be used.

Examples

```
ErrorDocument 403 /cgi-bin/notallowed.cgi
ErrorDocument 404 /cgi-bin/nph-error.pl
ErrorDocument 500 /serverError.html
ErrorDocument 501 /error/notImplemented.html
```

For more information on Error scripts[5]

[1] http://hoohoo.ncsa.uiuc.edu/docs/setup/ConfigureInfo.html#Directive</a
[2] http://hoohoo.ncsa.uiuc.edu/cgi/overview.html
[3] /cgi/ErrorCGI.html
[4] /cgi/nph-error.pl
[5] /cgi/ErrorCGI.html

Return to Resource Configuration File Overview[6]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[7] / Last Modified 7-12-95

[6] Overview.html
[7] mailto:httpd@ncsa.uiuc.edu

FancyIndexing directive

Purpose

This directive specifies whether you want fancy directory indexing (with icons and file sizes) or standard directory indexing.

If your documents are being served from an AFS drive, you will want to seriously consider turning fancy indexing off.

Syntax

`FancyIndexing setting`

`setting` is either `on` or `off`.

Only one FancyIndexing directive should appear in the configuration file.

File

srm.conf

Default

If no FancyIndexing directive is present, HTTPd assumes:

`FancyIndexing off`

Example

`FancyIndexing on`

Turns fancy indexing on.

Return to Resource Configuration File Overview[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / Last Modified 7−12−95

[1] Overview.html
[2] mailto:httpd@ncsa.uiuc.edu

HeaderName directive

Purpose

This directive specifies what filename HTTPd should look for when indexing a directory, in order to add a custom header. This can describe the contents of the directory.

Syntax

`HeaderName name`

`name` is the name of the file HTTPd should look for when trying to find a description file. HTTPd will first look for `name.html`, and if found, will display the HTML inlined with its own index. If it finds `name`, it will include the file as plaintext.

Only one HeaderName directive should appear in the configuration file.

File

srm.conf

Default

If no HeaderName is present, HTTPd assumes nothing.

Example

`HeaderName HEADER`

When generating an index for the directory `/foo`, HTTPd will look for `/foo/HEADER.html`, and will insert it at the top of the index if found. If not, it will then look for `/foo/HEADER` and insert it if found. If it finds nothing, it will include nothing.

Return to Resource Configuration File Overview[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / Last Modified 7-12-95

[1] Overview.html
[2] mailto:httpd@ncsa.uiuc.edu

IndexIgnore directive

Purpose

Tells HTTPd which files to ignore when generating an index of a directory.

Syntax

`IndexIgnore pat1 pat2...`

`pat` is a file extension or file name which should be ignored. When HTTPd is looking in a directory, it will try to match each of these strings to the right hand side of the entry's string, and if it matches it will ignore that entry in its directory index. If `pat` is a wildcard expression[1], HTTPd will match the filename against the given match expression.

File

srm.conf

Default

The only entry ignored by default is '.'.

Example

`IndexIgnore README README.html .htaccess # ~`

HTTPd will ignore files named README, README.html, and .htaccess when indexing a directory. It will also ignore emacs autosave files and emacs backup files.

Return to Resource Configuration File Overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7−12−95

1 ../MatchExp.html
2 Overview.html
3 mailto:httpd@ncsa.uiuc.edu

IndexOptions directive

Purpose

This directive specifies whether you want fancy directory indexing (with icons and file sizes) or standard directory indexing, and which options you want active for indexing.

If your documents are being served from an AFS drive, you will want to seriously consider turning fancy indexing off.

Syntax

`IndexOptions opt1 opt2...`

`opt` is an option name:

- `FancyIndexing` turns FancyIndexing on
- `IconsAreLinks` makes the icons part of the anchor for the filename
- `ScanHTMLTitles` will cause HTTPd to fill in the description field of any unknown HTML document with its title (title must be within 256 bytes of the start of the file). You should NOT turn this option on unless your server has CPU time to spare.
- `SuppressLastModified` will cause HTTPd not to print the last date of modification in index listings.
- `SuppressSize` will cause HTTPd not to print the size of the files in index listings.
- `SuppressDescription` will cause HTTPd not to print descriptions for any files.

Only one IndexOptions directive should appear in the configuration file.

File

srm.conf

Default

If no IndexOptions directive is present, HTTPd assumes that none of the options should be on.

Example

`IndexOptions FancyIndexing IconsAreLinks`

This will turn on fancy indexing with icons, and the icons will be part of the links to the files.

Return to Resource Configuration File Overview[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / Last Modified 7-12-95

[1] Overview.html
[2] mailto:httpd@ncsa.uiuc.edu

ReadmeName directive

Purpose

This directive specifies what filename HTTPd should look for when indexing a directory, in order to add a paragraph of description to the end of the index it automatically generates. Generally these paragraphs are used to give a general overview of what's in a directory.

Syntax

`ReadmeName name`

`name` is the name of the file HTTPd should look for when trying to find a description file. HTTPd will first look for `name.html`, and if found, will display the HTML inlined with its own index. If it finds `name`, it will include the file as plaintext.

Only one ReadmeName directive should appear in the configuration file.

File

srm.conf

Default

If no ReadmeName is present, HTTPd assumes nothing.

Example

`ReadmeName README`

When generating an index for the directory `/foo`, HTTPd will look for `/foo/README.html`, and will insert it if found. It will then look for `/foo/README` and insert it if found. If it finds nothing, it will include nothing.

Return to Resource Configuration File Overview[1]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[2] / Last Modified 7−12−95

[1] Overview.html
[2] mailto:httpd@ncsa.uiuc.edu

Redirect directive

Purpose

The Redirect directive[1] creates a virtual document on your server, and any accesses to it will be redirected to a new URL.

Syntax

`Redirect virtual URL`

`virtual` is the translated location[2] which should trigger a redirect..

`URL` is the URL[3] of the new document.

Several Redirect directives may appear in the configuration file.

File

srm.conf

Example

`Redirect /dir1 http://newserver.widget.com/dir1`

This would cause requests for /dir1 to be redirected to the new location, `http://newserver.widget.com/dir1`.

Return to Resource Configuration File Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

1 ../ConfigureInfo.html#Directive
2 ../admin/AccessingFiles.html
3 http://www.ncsa.uiuc.edu/demoweb/url−primer.html
4 Overview.html
5 mailto:httpd@ncsa.uiuc.edu

NCSA HTTPd ScriptAlias directive

Purpose

The ScriptAlias directive[1] creates a virtual directory on your server. Any accesses to that virtual directory will be satisfied by returning the output of a CGI server script[2] in that directory.

Syntax

`ScriptAlias virtual path`

`virtual` is the translated location[3] of the script directory.

`path` is the full pathname of the directory which contains server scripts which fulfill the request.

Note: You should always place a trailing / after ScriptAlias directives which reference directories, to prevent similar entries from conflicting with each other.

Several ScriptAlias directives may appear in the configuration file.

File

srm.conf

Example

`ScriptAlias /cgi-bin/ /usr/local/etc/httpd/cgi-bin/`

This would cause requests such as `/cgi-bin/foo` to be satisfied by running the script `/usr/local/etc/httpd/cgi-bin/foo`.

Thus, if someone requested `/cgi-bin/a-script`, the server would run `/usr/local/etc/httpd/cgi-bin/a-script` and send its output to the client.

Return to Resource Configuration File Overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified

1 ../ConfigureInfo.html#Directive
2 /cgi/
3 ../admin/AccessingFiles.html
4 Overview.html
5 mailto:httpd@ncsa.uiuc.edu

7-12-95

UserDir directive

Purpose

The UserDir directive[1] sets the real directory in a user's home directory to index upon receiving a request from a user supported directory[2].

Syntax

`UserDir dir`

Where `dir` is a partial path relative to the users home directory (as given in `/etc/passwd`). The server takes the path it creates with this information, and looks for the user's document in the resulting full path.

The keyword `DISABLED` will disable the user supported directories feature.

Only one UserDir directive is allowed in the server configuration file.

File

srm.conf

Default

If you do not specify a UserDir, HTTPd assumes:

`UserDir public_html`

Example

`UserDir web-docs`

This would set UserDir to web–docs. A request for `/~robm/foo.gif` would cause the server to retrieve `~robm/web-docs/foo.gif`.

Return to Resource Configuration File Overview[3]

1 ../ConfigureInfo.html#Directive
2 ../../info/UserDirectories.html
3 Overview.html

Access Configuration

Overview

When creating a document tree, NCSA HTTPd allows you to control certain aspects of the branches (directories) of that tree.

- Access

  Restrict access to a branch to allowed hosts or authenticated users.

- Server features

  Branches considered unsafe (such as users' home directories[1]) can be made more secure by disabling certain server functions in those directories.

Related tutorials

There are many aspects of server administration regarding security and access control. Please read this tutorial[2] on setting up a secure server. Also, here is a quick overview[3] of setting up user authentication, which you may find helpful. If multiple users are using your server, this tutorial on user management[4] may also be helpfull.

Methods

There are two methods for controlling access to directories.

- Global Access Configuration file

  A document in your server's `conf` directory, specified by the Server Configuration directive AccessConfig[5], controls access to any directory in your tree.

  HTTPd requires that you set up the Global ACF.

- Per–directory Access Configuration file

  Within your document tree, files with the name specified by the AccessFileName[6] directive in the Resource Configuration File[7] control access to the directory they are in as well as any subdirectories.

  Per–directory ACFs are optional. They can be restricted or completely forbidden by the Global ACF.

1 ../../info/UserDirectories.html
2 ../../tutorials/security.html
3 ../../tutorials/user.html
4 ../admin/UserManagement.html
5 ../httpd/AccessConfig.html
6 ../srm/AccessFileName.html
7 ../srm/Overview.html

Directives

In addition to the general information[8] which applies to all NCSA HTTPd configuration files, the access control files support the notion of sectioning directives[9].

This is a list of the directives[10] and sectioning directives used when writing an ACF. Each directive specifies where it can be used, and gives examples of usage.

- Directory[11]
- Options[12]
- AllowOverride[13]
- AddType[14]
- DefaultType[15]
- AddEncoding[16]
- AddDescription[17]
- AddIcon[18]
- IndexIgnore[19]
- DefaultIcon[20]
- ReadmeName[21]
- AuthName[22]
- AuthType[23]
- AuthUserFile[24]
- AuthGroupFile[25]
- Limit[26]
  - order[27]
  - deny[28]
  - allow[29]
  - require[30]
  - satisfy[31]

Example Configuration Files

For the security demo[32], I used this Global access configuration file[33].

8 ../ConfigureInfo.html
9 SectionDirectives.html
10 ../ConfigureInfo.html#Directive
11 Directory.html
12 Options.html
13 AllowOverride.html
14 AddType.html
15 DefaultType.html
16 AddEncoding.html
17 AddDescription.html
18 AddIcon.html
19 IndexIgnore.html
20 DefaultIcon.html
21 ReadmeName.html
22 AuthName.html
23 AuthType.html
24 AuthUserFile.html
25 AuthGroupFile.html
26 Limit.html
27 order.html
28 deny.html
29 allow.html
30 require.html
31 satisfy.html
32 ../../info/Security.html
33 access.conf.txt

The same effect could be achieved by taking everything within each Directory[34] section, and placing the directives in a ACF in each particular directory.

Return to Configuration Instructions[35]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[36] / Last Modified 7−12−95

[34] Directory.html
[35] ../Configure.html
[36] mailto:httpd@ncsa.uiuc.edu

AddDescription directive

Purpose

The AddDescription directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to only the per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

1 ../ConfigureInfo.html#Directive
2 ../srm/Overview.html
3 ../srm/AddDescription.html
4 Overview.html#Directives
5 mailto:httpd@ncsa.uiuc.edu

AddEncoding directive

Purpose

The AddEncoding directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to both the Global ACF as well as per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

1 ../ConfigureInfo.html#Directive
2 ../srm/Overview.html
3 ../srm/AddEncoding.html
4 Overview.html#Directives
5 mailto:httpd@ncsa.uiuc.edu

AddIcon directive

Purpose

The AddIcon directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to both the Global ACF as well as per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

[1] ../ConfigureInfo.html#Directive
[2] ../srm/Overview.html
[3] ../srm/AddIcon.html
[4] Overview.html#Directives
[5] mailto:httpd@ncsa.uiuc.edu

AddType directive

Purpose

The AddType directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to both the Global ACF as well as per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

1 ../ConfigureInfo.html#Directive
2 ../srm/Overview.html
3 ../srm/AddType.html
4 Overview.html#Directives
5 mailto:httpd@ncsa.uiuc.edu

AllowOverride directive

Purpose

The AllowOverride directive[1] controls which access control directives can be overruled by a per–directory ACF.

The global ACF cannot be restricted by this directive.

Scope

This directive may appear only in the global ACF.

Syntax

`AllowOverride or1 or2 ... orn`

Each `or` is one of the following:

- None

  ACFs are not allowed in this directory.

- All

  ACFs are unrestricted in this directory.

- Options

  Allow use of the Options[2] directive.

- FileInfo

  Allow use of the AddType[3] and AddEncoding[4] directives.

- AuthConfig

  Allow use of these directives:
  - AuthName[5]
  - AuthType[6]
  - AuthUserFile[7]
  - AuthGroupFile[8]

1 ../ConfigureInfo.html#Directive
2 Options.html
3 AddType.html
4 AddEncoding.html
5 AuthName.html
6 AuthType.html
7 AuthUserFile.html
8 AuthGroupFile.html

- Limit

  Allow use of the Limit[9] sectioning directive.

- Redirect

  Allow use of the Redirect[10] directive.

File

access.conf

Default

If no AllowOverride directives are given for this directory or any of its parents, HTTPd assumes:

`AllowOverride All`

Example

`AllowOverride Limit FileInfo`

ACFs in this directory are allowed to use `Limit`, `AddType`, and `AddEncoding`.

Return to access configuration overview[11]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[12] / Last Modified 7–12–95

9 Limit.html
10 ../srm/Redirect.html
11 Overview.html#Directives
12 mailto:httpd@ncsa.uiuc.edu

AuthGroupFile

Purpose

The AuthGroupFile directive[1] sets the file to use as a list of user groups for user authentication.

Scope

This directive applies to both global and per−directory ACFs.

This directive must be accompanied by AuthName[2], AuthType[3], and AuthUserFile[4] directives in order for user authentication to work properly.

Syntax

```
AuthGroupFile path
```

`path` is the absolute path of group file[5] to use in this directory.

File

access.conf

Default

There is no default.

Example

```
AuthGroupFile /usr/local/etc/httpd/conf/.htgroup
```

Sets the authorization group file for this directory to `/usr/local/etc/httpd/conf/.htgroup`.

Return to access configuration overview[6]

1 ../ConfigureInfo.html#Directive
2 AuthName.html
3 AuthType.html
4 AuthUserFile.html
5 ../admin/UserManagement.html#group
6 Overview.html#Directives

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[7] / Last Modified 7−12−95

7 mailto:httpd@ncsa.uiuc.edu

AuthName

Purpose

The AuthName directive[1] sets the name of the authorization realm for this directory. This realm is a name given to users so they know which username and password to send.

Scope

This directive applies to both global and per−directory ACFs.

This directive must be accompanied by AuthType[2], AuthUserFile[3], and AuthGroupFile[4] directives in order for user authentication to work properly.

Syntax

`AuthName name`

Where name is a short name describing this authorization realm. Can contain spaces.

File

access.conf

Default

There is no default.

Example

`AuthName PhysicsCollab`

Sets the authorization name of this directory to `PhysicsCollab`.

Return to access configuration overview[5]

[1] ../ConfigureInfo.html#Directive
[2] AuthType.html
[3] AuthUserFile.html
[4] AuthGroupFile.html
[5] Overview.html#Directives

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu / Last Modified 7−12−95[6]

[6] mailto:httpd@ncsa.uiuc.edu

AuthType

Purpose

The AuthType directive[1] sets the type of authorization used in this directory.

Scope

This directive applies to both global and per−directory ACFs.

This directive must be accompanied by AuthName[2], AuthUserFile[3], and AuthGroupFile[4] directives in order for user authentication to work properly.

Syntax

`AuthType type`

`type` is the authentication type to use for this directory. Only `Basic` is currently implemented.

File

access.conf

Default

There is no default.

Example

`AuthType Basic`

Sets the authorization type of this directory to `Basic`.

Return to access configuration overview[5]

1 ../ConfigureInfo.html#Directive
2 AuthName.html
3 AuthUserFile.html
4 AuthGroupFile.html
5 Overview.html#Directives

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu / Last Modified 7−12−95[6]

6 mailto:httpd@ncsa.uiuc.edu

AuthUserFile

Purpose

The AuthUserFile directive[1] sets the file to use as a list of users and passwords for user authentication.

Scope

This directive applies to both global and per−directory ACFs.

This directive must be accompanied by AuthName[2], AuthType[3], and AuthGroupFile[4] directives in order for user authentication to work properly.

Syntax

`AuthUserFile path`

`path` is the absolute path of a user file created with the `htpasswd`[5] support program.

File

access.conf

Default

There is no default.

Example

`AuthUserFile /usr/local/etc/httpd/conf/.htpasswd`

Sets the authorization user file for this directory to `/usr/local/etc/httpd/conf/.htpasswd`.

1 ../ConfigureInfo.html#Directive
2 AuthName.html
3 AuthType.html
4 AuthGroupFile.html
5 ../admin/UserManagement.html#htpasswd

Return to access configuration overview[6]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu / Last Modified 7−12−95[7]

6 Overview.html#Directives
7 mailto:httpd@ncsa.uiuc.edu

DefaultIcon directive

Purpose

The DefaultIcon directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to both the Global ACF as well as per–directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu / Last Modified 7–12–95[5]

1 ../ConfigureInfo.html#Directive
2 ../srm/Overview.html
3 ../srm/DefaultIcon.html
4 Overview.html#Directives
5 mailto:httpd@ncsa.uiuc.edu

DefaultType directive

Purpose

The DefaultType directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies only to per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu / Last Modified 7−12−95[5]

1 ../ConfigureInfo.html#Directive
2 ../srm/Overview.html
3 ../srm/DefaultType.html
4 Overview.html#Directives
5 mailto:httpd@ncsa.uiuc.edu

Directory directive

Purpose

Directory is a sectioning directive[1] which controls the directory to which access control directives apply.

Scope

This directive applies only to the Global ACF. All directives[2] in the Global ACF must be contained in a Directory section.

Syntax

Opening Directive:

```
<Directory dir>
```

`dir` is the absolute pathname of the directory you are protecting. If you are using NCSA HTTPd 1.2 or later, this can also be a wildcard expression[3] of a set of directories you wish to protect.

Closing directive:

```
</Directory>
```

File

access.conf

Example

```
<Directory /u/Web>
Options None
</Directory>
```

The directives contained within the Directory section above would only apply to the server directory /u/Web.

Return to access configuration overview[4]

1 SectionDirectives.html
2 ../ConfigureInfo.html#Directive
3 ../MatchExp.html
4 Overview.html#Directives

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu / Last Modified 7–12–95[5]

5 mailto:httpd@ncsa.uiuc.edu

IndexIgnore directive

Purpose

The IndexIgnore directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to both the Global ACF as well as per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

[1] ../ConfigureInfo.html#Directive
[2] ../srm/Overview.html
[3] ../srm/IndexIgnore.html
[4] Overview.html#Directives
[5] mailto:httpd@ncsa.uiuc.edu

Limit directive

Purpose

Limit is a sectioning directive[1] which controls which clients can access a directory.

Scope

This directive applies to both the global and per–directory ACFs.

Syntax

Opening Directive:

```
<Limit meth1 meth2 ... methn>
```

Each `meth` is one of the following methods:

- `GET`

  Allows clients to retrieve documents and execute scripts.
- `PUT`

  Not Implemented.
- `POST`

  Not fully implemented. Currently, allows clients to use POST scripts.

Closing directive:

```
</Limit>
```

Only the following directives are allowed inside Limit sections:

- order[2]
- deny[3]
- allow[4]
- require[5]

File

[1] SectionDirectives.html
[2] order.html
[3] deny.html
[4] allow.html
[5] require.html

access.conf

Example

```
<Limit GET>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
require group sdg
</Limit>
```

The only clients allowed to use the `GET` method in this directory must be from ncsa.uiuc.edu, and authenticate to group sdg.

Return to access configuration overview[6]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[7] / Last Modified 7–12–95

[6] Overview.html#Directives
[7] mailto:httpd@ncsa.uiuc.edu

Options directive

Purpose

The Options directive[1] controls which server features are available in a given directory.

Scope

This directive applies to both the Global ACF as well as per–directory ACFs.

Syntax

`Options opt1 opt2 ... optn`

Each `opt` is one of the following:

- None

  No features are enabled for in directory.

- All

  All features are enabled for in directory.

- FollowSymLinks

  The server will follow symbolic links in this directory.

- SymLinksIfOwnerMatch

  The server will only follow symbolic links for which the target file/directory is owned by the same user id as the link.

- ExecCGI

  Execution of CGI scripts[2] is allowed in this directory.

- Includes

  Server side include files[3] are enabled in this directory.

- Indexes

  The server allows users to request indexes[4] in this directory. Disabling this

[1] ../ConfigureInfo.html#Directive
[2] ../../tutorials/cgi.html
[3] ../../tutorials/includes.html
[4] ../../info/Indexes.html

  option disables ONLY the server–generated indexes. It does not stop the server from sending any precompiled index file it may find in there (the name of which depends on DirectoryIndex[5]).

- IncludesNoExec

  This enables server side includes[6] in the directory, but disables the exec[7] feature.

File

access.conf

Default

If no Options directives are given for this directory or any of its parents, HTTPd assumes:

`Options All`

Example

`Options Indexes FollowSymLinks`

The server would allow users to index this directory and its subdirectories, and would allow symbolic links to be used within.

Return to access configuration overview[8]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[9] / Last Modified 7–12–95

[5] ../srm/DirectoryIndex.html
[6] ../../tutorials/includes.html
[7] ../../tutorials/includes.html#exec
[8] Overview.html#Directives
[9] mailto:httpd@ncsa.uiuc.edu

ReadmeName directive

Purpose

The ReadmeName directive[1] acts exactly as it does in the resource configuration file[2].

Scope

This directive applies to both the Global ACF as well as per−directory ACFs.

Syntax

See the description[3] from the resource configuration section.

File

access.conf

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7−12−95

1 ../ConfigureInfo.html#Directive
2 ../srm/Overview.html
3 ../srm/ReadmeName.html
4 Overview.html#Directives
5 mailto:httpd@ncsa.uiuc.edu

Sectioning directives

Definition

Some access control directives require that certain information apply to all directives in a given section. These directives are called sectioning directives.

The formatting of these directives is similar to HTML tags[1].

For each sectioning directive, there must be two components: an opening directive, and a closing directive.

For instance, for the sectioning directive `Foo`, with one argument `data`, the opening directive would be:

```
<Foo data>
```

The closing argument would be:

```
</Foo>
```

The information given in the opening directive will affect all other directives between the opening and closing sectioning directive.

File

access.conf

Example

The emphasized words are here for clarification; they would not appear in the actual configuration file.

```
<Directory /u/Web>                       The opening directive
require group physics
</Directory>                             The closing directive
```

`Directory` is the sectioning directive. Therefore, the information given by the opening directive, `/u/Web`, applies to the `require` directive within.

Return to access configuration overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7-12-95

[1] http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html#MinimalHTML
[2] Overview.html#Directives
[3] mailto:httpd@ncsa.uiuc.edu

allow directive

Purpose

The allow directive affects which hosts can access a given directory with a given method.

Scope

This directive is only available within Limit[1] sections.

Syntax

`allow from host1 host2 ... hostn`

`host` is one of the following:

- A domain name

  A domain name, like .ncsa.uiuc.edu, which host names must end in to be allowed.

- A host name A full host name.
- A full IP address

  An IP address of a host.

- A partial IP address

  The first 1−3 bytes of an IP address, for subnet restriction.

- The keyword `all`

  This means that all hosts will be allowed.

File

access.conf

Default

No default applies.

[1] Limit.html

Example

```
<Limit /u/Web>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
</Limit>
```

In the /u/Web directory, the server evaluates the deny directive first. So, everyone is denied. It then evaluates the allow directive, and decides to allow clients from .ncsa.uiuc.edu.

Return to access configuration overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7−12−95

[2] Overview.html#Directives
[3] mailto:httpd@ncsa.uiuc.edu

NCSA HTTPd deny directive

Purpose

The deny directive affects which hosts can access a given directory with a given method.

Scope

This directive is only available within Limit[1] sections.

Syntax

`deny from host1 host2 ... hostn`

`host` is one of the following:

- A domain name

  A domain name, like .ncsa.uiuc.edu, which host names must end in to be allowed.
- A host name A full host name.
- A full IP address

  An IP address of a host.
- A partial IP address

  The first 1−3 bytes of an IP address, for subnet restriction.
- The keyword `all`

  This means that all hosts will be denied.

File

access.conf

Default

No default applies.

[1] Limit.html

Example

```
<Limit /u/Web>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
</Limit>
```

In the /u/Web directory, the server evaluates the deny directive first. So, everyone is denied. It then evaluates the allow directive, and decides to allow clients from .ncsa.uiuc.edu.

Return to access configuration overview[2]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[3] / Last Modified 7−12−95

[2] Overview.html#Directives
[3] mailto:httpd@ncsa.uiuc.edu

order directive

Purpose

The order directive affects the order in which deny[1] and allow[2] directives are evaluated within a Limit section.

Scope

This directive is only available within Limit[3] sections.

Syntax

`order ord`

`ord` is one of the following:

- `deny,allow`

  In this case, the deny directives are evaluated before the allow directives.

- `allow,deny`

  In this case, the allow directives are evaluated before the deny directives.

- `mutual-failure`

  This order is a bit unorthodox. With this order, you specify specific hosts which must be allowed or denied. Any host appearing on the allow list is allowed, and any list on the deny list is denied. Any appearing on neither is denied.

File

access.conf

Default

If no order is given, HTTPd assumes:

`order deny,allow`

[1] deny.html
[2] allow.html
[3] Limit.html

Example

```
<Limit GET>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
</Limit>
```

In the /u/Web directory, the server evaluates the deny directive first. So, everyone is denied. It then evaluates the allow directive, and decides to allow clients from .ncsa.uiuc.edu.

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7-12-95

[4] Overview.html#Directives
[5] mailto:httpd@ncsa.uiuc.edu

require directive

Purpose

The require directive affects which authenticated users can access a given directory with a given method.

Scope

This directive is only available within Limit[1] sections.

Syntax

`require entity en1 en2 ... enn`

`en` are entity names, separated by spaces.

`entity` is one of the following:

- `user`

  Only the named users[2] can access this directory with the given methods.

- `group`

  Only users in the named groups[3] can access this directory with the given methods.

- `valid-user`

  All of the users defined in the AuthUserFile[4] are allowed access upon providing a valid password.

Default

No default applies.

File

access.conf

[1] Limit.html
[2] ../admin/UserManagement.html
[3] ../admin/UserManagement.html#group
[4] AuthUserFile.html

Example

```
<Limit GET PUT>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
require user ls
require group sdg
</Limit>
```

In this directory, the server evaluates the deny directive first. So, everyone is denied. It then evaluates the allow directive, and decides to allow clients from .ncsa.uiuc.edu. Now, it uses user authentication and only allows users who are named ls or are in the group sdg.

Return to access configuration overview[5]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[6] / Last Modified 7-12-95

[5] Overview.html#Directives
[6] mailto:httpd@ncsa.uiuc.edu

NCSA HTTPd Satisfy directive

Purpose

The satisfy directive allows you to specify how access is allowed if you use both allow[1] and require[2] directives.

Scope

This directive is only available within Limit[3] sections.

Syntax

`Satisfy all | any`

- `all`

  In order to gain access to a directory, the user must satisfy both the allow and require directives.
- `any`

  Users are only required to meet one of the specified allow or require directives.

Default

The default is all.

File

access.conf

Example

```
<Limit GET PUT>
order deny,allow
deny from all
allow from .ncsa.uiuc.edu
require user ls
satisy any
</Limit>
```

[1] allow.html
[2] require.html
[3] Limit.html

In this directory, the server evaluates the deny directive first. So, everyone is denied. It then evaluates the allow directive, and decides to allow clients from .ncsa.uiuc.edu. With the satisfy any, if you are from .ncsa.uiuc.edu, access is granted. Otherwise, it uses user authentication and only allows users who are named ls.

Return to access configuration overview[4]

NCSA HTTPd Development Team / httpd@ncsa.uiuc.edu[5] / Last Modified 7-12-95

[4] Overview.html#Directives
[5] mailto:httpd@ncsa.uiuc.edu