Ari Luotonen, CERN
Tim Berners-Lee, CERN

CERN httpd Reference Manual

A Guide To A World-Wide Web HyperText Daemon

CERN Server User Guide

Generated from the Hypertext

May 4, 1994

Table of Contents

1. CERN httpd 3.0 Guide for Prereleases

CERN WWW Server [`httpd`, HyperText Transfer Protocol Daemon]is a generic, full featured server for serving files using the HTTP protocol. This is a TCP/IP based protocol running by convention on port 80.

Files can be real or synthesized, produced by scripts generating virtual documents. It handle clickable images, fill-out forms, and searches etc.

CERN `httpd` can also be run as a proxy server to allow people behind firewalls to use the Web as if the firewall was not present. A powerful feature is caching performed by the proxy, which makes `cern_httpd` as proxy attract even those not inside a firewall.

- This documentation is also available in PostScript.
- Documentation for older versions is still available: [2.14 or older][2.15][2.16][2.17 & 2.18].
- If you upgrade see also release notes for [2.15][2.16][2.17][2.18][3.0pre1-3].
- **Current VMS Version is 2.16beta. See distribution.** See also Foteos Macrides' fixes.

1.1 In This Guide...

Installation	The steps necessary to install CERN server.
Administration	How to set up document protection, index search, clickable images, server-side scripts, ...

1.2 About documents generated from hypertext

Paper manuals generated from hypertext are made for convenience, for example for reading when one has no computer to turn to. We have tried to make the hypertext into fairly conventional paper documents, but they may seem a little strange in some ways.

All the links have been removed. Therefore, it is worth looking at the table of contents to see what there is in the manual. Something which is not explained in place may be explained in detail elsewhere.

We have tried to keep related matter together, but sometimes necessarily you might have to check the table of contents to find it.

Please remember that these are for the most part "living documents". That is, they are constantly changing to reflect current knowledge. If you see a statement such as "Product xxx does not support this feature", remember that it was the case when the document was generated, and may not be the same now. So if in doubt, check the online version. Of course, the living document may be out of date too, in which case it is helpful to mail its author.

2. Installing CERN Server

VMS note: There are special instructions if you are installing under VMS.

2.1 Getting the Program

CERN server distribution is available from `info.cern.ch` anonymous ftp account. Often you don't need to compile the server yourself, precompiled binaries are available for many Unix platforms. If there is no precompiled version for your platform, of if it doesn't work (e.g. the name resolution doesn't work), you should get the source code and compile it yourself.

- Precompiled versions can be found under directory
  `ftp://info.cern.ch/pub/www/bin` (in the subdirectory corresponding your machine architecture).
- Source code `ftp://info.cern.ch/pub/www/src/cern_httpd.tar.Z`.
  Compilation:
  - Uncompress and untar the distribution tar file:
    ```
    uncompress cern_httpd.tar.Z
    tar xvf cern_httpd.tar
    ```
  - Go to newly-created `WWW` directory, and give command `./BUILD`:
    ```
    cd WWW
    ./BUILD
    ```
  - Executable `httpd` appears in directory `.../WWW/Daemon/sun4` (if you have a Sun4 machine), or in another subdirectory corresponding to your machine architecture. The utility programs go to the same directory (`htadm`, `htimage`, `cgiparse` and `cgiutils`).

2.2 Configuration File

- `httpd` requires a configuration file, the default configuration file is `/etc/httpd.conf`. If this doesn't suit you, you can specify another location to it using the `-r` option:
  ```
  httpd -r /other/place/httpd.conf
  ```
- Sample configuration files are available from
  - directory `cern_httpd/config` inside the binary distribution, or
  - under `WWW/server_root` inside the source code distribution.
  - If this is missing you can get them from
    `ftp://info.cern.ch/pub/www/src/server_root.tar.Z`

If you have all your documents in a single directory tree, say `/Public/Web`, the easiest way to make them available to the world is to specify the following rule in your configuration file:

```
Pass /* /Public/Web/*
```

This maps all the requests under the directory `/Public/Web` and accepts them.

The default welcome document (what you get with URL of form `http://your.host/`) is now `Welcome.html` in the directory `/Public/Web`.

2.3 First Trying It Out In Verbose Mode

Often it is easy to make mistakes in the configuration file that makes configuring `httpd` feel tedious - this doesn't have to be so. In the beginning start `httpd` by hand in verbose mode to listen to some port, and look what happens when you make a request to that port with your browser.

Typically test servers are run on a non-priviledged port above 1024 (you don't have to be `root` to bind to them), often 8001, 8080, or such. Official HTTP port is 80.

The server port is defined in the configuration file with the `Port` directive, but you can override it with the `-p` command line option while testing; e.g.

```
httpd -v -r /home/you/httpd.conf -p 8080
```

This will start `httpd` in verbose mode, use configuration file `httpd.conf` in your home directory, and accept connections to port 8080.

You can now try to request a document form your server using a URL of form:

```
http://your.host:8080/document.html
```

where `document.html` is relative to the directory that you have exported in your configuration file.

If you get an error message back see the verbose output to find out what is going wrong - it is usually self-explanatory.

And remember, you should always feel free to ask advice from **httpd@info.cern.ch**.

2.4 The Actual Installation of httpd

In Unix you can run the server either as stand-alone, or from Internet Daemon `(inetd)`. A stand-alone server is typically started once at system-boot time. It waits for incoming connections, and forks itself to serve a request. **This is much faster** than letting `inetd` spawn `httpd` every time a request comes. **We therefore recommend that you run CERN httpd in stand-alone mode.**

2.4.1 Stand-alone Installation

A stand-alone server is started from the bootstrap command file (for example `/etc/rc.local)` so that it runs continuously like the `sendmail` daemon, for example.

This method has the advantage over using the `inetd` that the response time is reduced.

Add a line starting `httpd` to your system startup file (usually `/etc/rc.local` or `/etc/rc`). If you have the configuration file in the default place, `/etc/httpd.conf`, and if it specifies the port to listen to via the `Port` directive, you don't need any command line options:

```
/usr/etc/httpd &
```

`httpd` will automatically go background so there is really no need for an ampersand in the end (as long as your configuration file `/etc/httpd.conf` really exists).

Or a little more safely in case httpd is removed:

```
if [ -f /usr/etc/httpd ]; then
    (/usr/etc/httpd  && (echo -n ' httpd') ) & >/dev/console
fi
```

Naturally you can use any of the command line options, if necessary.

2.5 Registering Your Server

Once you have your `httpd` up and running, and you have documents to show the word, announce your server, so that others can find it.

2.6 If It Doesn't Work...

...first run it in verbose mode with the `-v` option and try to figure out what goes wrong. See also the debugging chart and the FAQ. If you can't figure out what's going wrong, feel free to send mail to **httpd@info.cern.ch**

2.7 Installing httpd Under inetd

This is how to to set up `inetd` to run `httpd` whenever a request comes in. (These steps are the same for any daemon under unix: you will probably find a similar thing has been done for the FTP daemon, `ftpd,` for example.)

2.7.1 Step 1: Install httpd Binary

Copy `httpd` into a suitable directory such as `/usr/etc.` Make it owned by `root`, and make it writable only to `root,` for example by saying:

```
chmod 755 httpd
```

2.7.2 Step 2: Add http Service to /etc/services

Put "http" in the `/etc/services` file, or use the name of a specific service of your own if you want to use a special port number. Standard port number for HTTP is 80.

```
http    80/tcp          # WWW server
```

Exceptions:

- On a NeXT, see using the NetInfomanager
- On any machine running NIS (yellow pages), see specicial instructions.

2.7.3 Step 3: Add a Line to /etc/inetd.conf

Put a line in the internet daemon configuration file, `/etc/inetd.conf.`

```
http  stream  tcp  nowait  root  /usr/etc/httpd  httpd
```

First word is the same as in `/etc/services` file.
If you want to pass command line options or parameters to `httpd,` they would listed be in the end of line, for example to set the rule file to something else than the default `/etc/httpd.conf:`

```
http  stream  tcp  nowait  root  /usr/etc/httpd  httpd -r /my/own/rules
```

Note: For `httpd` version 2.15 and later we recommend that it is run as user `root.`
Running `httpd` as `root` is safe, since it automatically resets its user-id to `nobody.` However, if you decide to use access authorization features, and you need to serve protected files, `httpd` will have to be able to set its user-id to some other uid as well. In any case, `httpd` always sets its user-id to something other than `root` before serving the file to the client.
Note: `/etc/inetd.conf` syntax varies from system to system, for example all systems don't have the field specifying the user name, in which case the default is `root.` If in doubt, sopy the format of other lines in your existing `inetd.conf.`
Note: There seems to be a limit of 4 arguments passed across by `inetd,` at least on the NeXT.

2.7.4 Step 4: Send HUP Signal to inetd

When you have updated `inetd.conf,` find out the process number of `inetd,` and send a "HUP" signal to it.
For example on BSD unix do this:

```
> ps -aux | grep inetd | grep -v grep
root     85   0.0  0.9 1.24M  304K ?  S  0:01 /usr/etc/inetd
> kill -HUP 85
```

For system V, use `ps -el` instead of `ps -aux`.
Be aware that on some systems your local file /etc/services may not be consulted by your system (see notes on debugging).

2.7.5 Test It!

2.7.6 Using NIS (Yellow Pages)

If your machine is running Sun's "Network Information Service", originally know as "yellow pages", read this.
You must:

- First make an addition to the `/etc/services` file just as for a normal unix system.
- Then, change directory to `/var/yp` and run `make`.

This will load the `/etc/services` file info the NIS information system.
Some people have found that they needed to reboot he system afterward for the change to take effect.

2.7.7 Adding a Service on the NeXT

The NeXT uses the the "netinfo" database instead of the `/etc/services` file. This is managed with the `/NextAdmin/NetInforManager` application. Here's how to add the service `http`:

- Start the NetInfomanager by double-clicking on its icon.
- If you are operating in a cluster, open either your local domain `(/hostname)` or if you have authority, the whole cluster domain `(/)`. If you're not in a cluster, just use the domain you are presented with.
- Select `"services"` from the browser tree.
- Select `"ftp"` from the list of services.
- Select `"dupliacte"` from the edit menu.
- Select `"copy of ftp"` and double-click on its icon to get the property editor.
- Click on `"name"` and then on the value `"copy of ftp"`. Change this to `"http"` by typing "http" in the window at the botton, and hitting return.
- Click on `"port"`, and then on the value `21`. Change it to `80`.
- Use `"Directory:Save"` menu `(Command/s)` to save the result. You will have to give a root password or netinfo manager password.

2.8 Priviliged ports

The TCP/IP port numbers below 1024 are special in that normal users are not allowed to run servers on them. This is a security feaure, in that if you connect to a service on one of these ports you can be fairly sure that you have the real thing, and not a fake which some hacker has put up for you.
The normal port number for W3 servers is port 80. This number has been assigned to WWW by the Internet Assigned Numbers Authority, IANA.
When you run a server as a test from a non-priviliged account, you will normally test it on other ports, such as 2784, 5000, 8001 or 8080.

2.8.1 Under Unix

The Internet Daemon `inetd` (running as root) can listen for incomming conections on port 80 and pass them down to a process with a safer uid for the server itself. However, the `httpd` versions 2.14 and later can be safely run as `root` since they automatically change their user-id to `nobody` or some other user-id depending on server setup.

2.8.2 Under VMS

Under UCX, the process running as a server needs BYPASS privilege to listen to ports below 1024. This might mean you have to install the server. With other TCP/IP packages, privilege of some sort is similarly required.

2.9 Debugging httpd

Suppose you think you have installed `httpd` but it doesn't work. Here we assume you have used port 80. If you have a situation not handled by this problem-solving guide, please mail `httpd@info.cern.ch`.
Type

```
www http://myhost.domain/
```

What happens?

2.9.1 Connection Refused

The browser tries to connect to the daemon but gets this status in the trace.
This means that nobody was listening on that port number. Check the port numbers match between server and client. Make sure you specify the port number explicitly in the document address for `www`.
If you are running the daemon standalone (as you should be), check that it is actually running by taking a list of processes, and that it is listening to the correct port (specified with `-p` *port* option), or try running it from the terminal with `-v` option as well. The trace for the server should say `"socket, bind and listen all ok".` If it does, and you still get "`connection refused`", then you must be talking to the wrong host (or, conceivably, different ethernet adapters on the same host).
If you are running with the inet daemon, then check both the services file `(/etc/services)` or database (yellow pages, netinfo) if your system uses it, and the `/etc/inetd.conf` file. Check the service name matches between these two (e.g. `http`).
Did you remember to kill -HUP the `inetd` when you changed the `inetd.conf` file?
Be aware that on some systems your local file `/etc/services` will not be consulted E.g. when `ypbind` is running on Suns, then you should type

```
ypwhich -m services
```

and ask the administrator of the machine named to change its own `/etc/services`.
Try running the deamon from a shell window to see better what happens.

2.9.2 Cannot Connect To Information Server

The usual cause of this is that the server is not running, or it's running on a different port.

There is more information you can get. Use the "verbose" option on the LineMode browser to find out what went wrong:

```
www -v http://myhost.domain:80/
```

What do you get? A load of trace messages. There are several cases.

- The browser can't look up the name of the host. If it can, it will display "Parsed address as" message. If not, try fixing your name server or `/etc/hosts` file, or quoting the IP number of the host in decimal notation (like 128.141.77.45) instead.
- The browser can get to the host but gets `Connection refused` status back.
- Your browser gets an error number but prints "error message not translated". This is because when it was compiled on your platform it didn't know what form the error message table took. Try the same thing form a unix platform for example.
- You get some network error like "network unreachable". Depending on whether the IP network is your responsibility or not, and your attitude to life, either fix it, try again in an hour's time, or complain to someone.

2.9.3 Unable To Access Document

Typical cause of this is that the configuration file is incorrect, or files are not readable by the user-id under which the server runs. When you are running the server as `root,` it will automatically switch it to `nobody` just before serving the document. This can be changed with the `UserId` configuration directive.

2.9.4 An Empty Document Is Displayed

The document sent back is empty, but there is no error message.

The `inetd` has started a process to run your server but it immediately failed. Possibilities include:

- When running from `inetd`, the daemon may not be in the file specified, or may not be executable by the specified user (or, if a user id is not specified in your variety of `inetd.conf`, `root`).
- For some reason server crashes when it's trying to serve the request. If you can, try to tract down when this happens, and send mail to `httpd@info.cern.ch`. Try running the daemon from a terminal window to see what happens.
- Script fails to produce any result, which may be due to the fact that there is no empty line after the header section output by the script, causing server to read the entire generated document as the header section.

2.9.5 Document Address Invalid Or Access Not Authorized...

...or some similar kind of error message. This means either:

- You have been passed a bad document address. If you are following a link, check with the author of the document which contained the link.

- The document has been moved. Check with the server administrator. You should be able to find out who runs the server by going to the welcome page (type "g /" with the line mode browser) and seeing a link to information about the maintainers.

If you are the server administrator, and you can't understand why the daemon refuses to deliver the file,

- Check the configuration file (rule file, by default `/etc/httpd.conf`) if you have one. Think out way the document name will be mapped successively by each line, and what the result will be.
- Run the daemon in debug mode from a terminal session to get trace information.

2.9.6 Bad Output

A document is displayed, but not the one you wanted.
These are some ideas:

- Try running the server from the terminal.
- Check the HTML source the daemon produces with

```
www -source http://my.host.domain/
```

- Try telnetting to httpd and simulating the client:

```
> telnet my.host.domain 80
Connected to my.host.domain on port 80
Escape is ^[
GET /document/name
```

2.9.7 Running Under Shell

You don't have to run the daemon under the `inetd` if it doesn't work (and we recommend running it standalone anyway). You can run it from a shell session.
Run `httpd` from your terminal turned on, with a different port number like 8080:

```
httpd -p 8080
```

Note: You must be `root` (under VMS, have some privilege) to run with a port number below 1024. If you select a port above 1024, then you can run as a normal user. This way, anyone can publish files on the net. Howeever, it isn't very reliable, as your server will not automatically come back up if the machine is rebooted. In the long term it is best to install it to be started from the system startup file `/etc/rc` or `/etc/rc.local`.
You may not be able to use a port number which has been used by a daemon process recently (port may still be bound), so you may have to switch port number if you ˆC and restart `httpd`. When it is running like this, you can also read the debugging messages (when running with `-v` option), and use a debugger on it if necessary. (See also: telnetting to the server).

Debugging using Trace

If you can't understand why a server refuses to give back a document, then run with the `-v` option to turn on debugging messages. Use `-v` as the very first command line option (this way debugging is turned on right away). You will see the daemon setting up the rules for translating requests into local URLs, and you will see its attept to access the file (assuming you map requests onto files).

```
httpd -v -p 8080
```

Try to access the document from a client using another terminal window. Look at the debugging output. It will probably explain what is happening. If you still can't figure out the problem, mail your local guru help desk or if desperate `httpd@info.cern.ch` **enclosing** a copy of debugging output.

Even simpler

For testing a daemon very simply, without using a client, you can make the terminal be the client. With `httpd` try just running it with the terminal and typing `GET` */document/url* into its input:

```
httpd -v
GET /document/url
```

2.9.8 Telnetting to httpd

Most implementations of telnet allow you to specify a port number. Under unix this is often just a second parameter, under VMS a `/PORT` option.

The HTTP protocol is a telnet protocol, so you can simulate it just by typing things in. This will help you to see exactly what a sending back, and it will check you that it really is the server not the browser which has a problem.

Here is a simple example (keybord input is in **boldface**):

```
> telnet myhost.domain 80
Connected to myhost.domain on port 80
Escape is ^[
GET /document/url
...document or error message...
```

3. Command Line of CERN httpd

The command line syntax for `httpd` allows a number of options and an optional directory argument:

```
httpd  [-opt -opt -opt ...] [directory]
```

The directory argument, if present, indicates the directory to be exported. If not present, either a rule file is be used, to export combinations of directories, or else the default is to export the `/Public` directory tree.

3.1 Options

`-r` ***rulefile*** Use *rulefile* as configuration file. **This is the only necessary command line option** if you don't have the default configuration file, `/etc/httpd.conf`. All the other options can be given as directives in the configuration file.

`-p` ***port*** Listen to port *port*. Without this argument `httpd` assumes that it has been run by `inetd`, and uses `stdin` and `stdout` as its communication channel. **Note** that port numbers under 1024 are privileged.

`-l` ***logfile*** Use *logfile* to log the requests.

`-restart` Restart an already running `httpd`. `httpd` finds the out the process number of the running server from `PidFile` and sends it the `HUP` signal (HangUP). This will cause `httpd` to reload its configuration files and reopen its log files. **Important:** To find out the `PidFile` `httpd` will have to read the same configuration file as the running `httpd` has, so you have to specify the same `-r` options on the command line as for the actual `httpd`.

`-gc_only` [only for proxies]Do only garbage collection and then exit. This can be used to run `httpd` periodically by `cron` to do garbage collection on a cache that is used by `httpd` run from the `inetd` daemon rather than standalone. When `httpd` is not running standalone it cannot monitor the cache, nor perform automatic garbage collection.

`-v` Verbose, turn on debugging messages.

`-vv` Very Verbose, turn on even more verbose debugging messages.

`-version` Print version number of `httpd` and `libwww` (the WWW Common Library).

3.1.1 Directory Browsing

You can set these also with the `DirAccess` configuration directive.

`-dy` Enable direcory browsing. Directories are returned as hypertext documents. See browsing directories. *Default.*

`-dn` Disable directory browsing. An attempt to access a directory will generate an error response.

`-ds` Selective directory browsing; enabled only for directories containing a file named `.www_browsable`

3.1.2 README Feature

It is common practice to put a file named `README` into a directory containing instructions or notices to be read by anyone new to the directory. `httpd` will by default embed any `README` file in the hypertext version of a directory.

You can set these also with the `DirReadme` configuration directive.

`-dt` For any browsable directory which contains a `README` file, include the text of the `README` file at the top of the document before the listing. *Default.*

-db As `-dt` but put the `README` at the bottom, after the listing. The `-db` and `-dt` options may be combined with `-dy` as `-dyb`, `-dty` etc.

-dr Disables the `README` inclusion feature.

3.2 Examples

```
httpd -r /usr/etc/httpd.conf -p 80
```

This is a standalone server running on port 80. Configuration file is `/usr/etc/httpd.conf` instead of the default, `/etc/httpd.conf`.

Note that if the `Port` directive is given in the configuration file the `-p` option is not necessary (it can be used to override the value set in the configuration file).

```
httpd
```

`httpd` uses its default configuration file `/etc/httpd.conf`. If that file doesn't exist, `httpd` exports the `/Public` directory tree. This tree may contain soft links to other directory trees.

If the configuration file `/etc/httpd.conf` didn't define the port number to listen to this is an `httpd` reading its `stdin` and writing to its `stdout`, so it is run by `inetd`.

```
httpd -r /usr/local/lib/httpd.conf
```

The same as before, but uses `/usr/local/lib/httpd.conf` as a rule file instead of the default `/etc/httpd.conf`.

4. Configuration File of CERN httpd

The configuration file (often referred to as the rule file) defines how `httpd` will translate a request into a document name. The directives controlling `httpd` features are also put into the configuration file, as well as protection configuration. This is essential to prevent unauthorized access to your private documents.

4.1 Default Configuration File

By default, the configuration file `/etc/httpd.conf` is loaded, unless specified otherwise with the `-r` command line option:

```
httpd -p 80 -r /your/own/httpd.conf
```

See also example configuration files.

4.2 Comments in Configuration File

Each line consists of an operation code and one or two parameters, referred to as the template and the result. Lines starting with a hash sign # are ignored, as are empty lines.

4.3 Restarting the Server

When you are running the server in standalone mode (not from `inetd`), and modify the configuration file, send the `HUP` signal to `httpd` to make it re-read the configuration file. You can find out the process number from the pid file written by httpd, e.g.

```
> cat /server_root/httpd-pid
2846
> kill -HUP 2846
>
```

You must specify the configuration file as an **absolute pathname** for the `-r` option because when the server is started in standalone mode it changes its current directory to / so after startup it cannot reload configuration files that were specified with relative filenames.

To make restarting easier `httpd` has a `-restart` option, which will automatically send the HUP signal to another `httpd` process. **Important:** To find out the `PidFile` `httpd` will have to read the same configuration file as the running `httpd` has, so you have to specify the same `-r` options on the command line as for the actual `httpd`, e.g.

```
> httpd -r /usr/etc/httpd.conf -restart
Restarting.. httpd
Sending..... HUP signal to process 21379
>
```

4.4 Exhaustive List of Configuration Directives

- General settings:
  - `ServerRoot`
  - `HostName`

  - `Port`
  - `PidFile`
  - `UserId`
  - `GroupId`
  - `Enable`
  - `Disable`
  - `IdentityCheck`
  - `Welcome`
  - `AlwaysWelcome`
  - `UserDir`
  - `MetaDir`
  - `MetaSuffix`
  - `MaxContentLengthBuffer`

- URL translation rules:
  - `Map`
  - `Pass`
  - `Fail`
  - `Redirect`
  - `Protect`
  - `DefProt`
  - `Exec`

- Filename suffix definitions:
  - `AddType`
  - `AddEncoding`
  - `AddLanguage`
  - `SuffixCaseSense`

- Accessory scripts:
  - `Search`
  - `POST-Script`
  - `PUT-Script`
  - `DELETE-Script`

- Directory listings:
  - `DirAccess`
  - `DirReadme`
  - `DirShowIcons`
  - `DirShowBrackets`

  - `DirShowMinLength`
  - `DirShowMaxLength`
  - `DirShowDate`
  - `DirShowSize`
  - `DirShowBytes`
  - `DirShowHidden`
  - `DirShowOwner`
  - `DirShowGroup`
  - `DirShowMode`
  - `DirShowDescription`
  - `DirShowMaxDescrLength`
  - `DirShowCase`
- Icons in directory listings:
  - `AddIcon`
  - `AddBlankIcon`
  - `AddUnknownIcon`
  - `AddDirIcon`
  - `AddParentIcon`
- Logging:
  - `AccessLog`
  - `ErrorLog`
  - `LogFormat`
  - `LogTime`
  - `NoLog`
  - `CacheAccessLog`
- Timeouts:
  - `InputTimeOut`
  - `OutputTimeOut`
  - `ScriptTimeOut`
- Proxy Caching:
  - `Caching`
  - `CacheRoot`
  - `CacheSize`
  - `NoCaching`
  - `CacheOnly`
  - `CacheClean`

  - CacheUnused
  - CacheDefaultExpiry
  - CacheLastModifiedFactor
  - CacheTimeMargin
  - CacheNoConnect
  - CacheExpiryCheck
  - Gc
  - GcDailyGc
  - GcMemUsage
  - CacheLimit_1
  - CacheLimit_2
  - CacheLockTimeOut
  - CacheAccessLog
- Going through many proxies:
  - http_proxy
  - ftp_proxy
  - gopher_proxy
  - wais_proxy
  - no_proxy

4.5 General CERN httpd Configuration Directives

- ServerRoot
- HostName
- Port
- PidFile
- UserId
- GroupId
- Enable
- Disable
- IdentityCheck
- Welcome
- AlwaysWelcome
- UserDir
- MetaDir
- MetaSuffix
- MaxContentLengthBuffer

4.5.1 ServerRoot

Server's "home" diretory is specified via `ServerRoot` directive. If server root is specified, but no `AddIcon` directive has been used in configuration file to set up icons, the default icon directory is under server root `icons`. The default icons that should be present are:

- `blank.xbm` blank icon for aligning the header with listing
- `directory.xbm` for directories
- `back.xbm` for parent directory
- `unknown.xbm` for unknown types
- `binary.xbm` for binary files
- `text.xbm` for text files
- `image.xbm` for image files
- `movie.xbm` for movies
- `sound.xbm` for audio files
- `tar.xbm` for tar files
- `compressed.xbm` for compressed files

If these defaults don't please you you can define all from the scratch. As an example of `AddIcon` directive, the defaults would be specified as follows:

```
Pass  /httpd-internal-icons/*  /server_root/icons/*

AddBlankIcon   /httpd-internal-icons/blank.xbm
AddDirIcon     /httpd-internal-icons/directory.xbm  DIR
AddParentIcon  /httpd-internal-icons/back.xbm       UP
AddUnknownIcon /httpd-internal-icons/unknown.xbm
AddIcon        /httpd-internal-icons/binary.xbm     BIN  binary
AddIcon        /httpd-internal-icons/text.xbm       TXT  text/*
AddIcon        /httpd-internal-icons/image.xbm      IMG  image/*
AddIcon        /httpd-internal-icons/movie.xbm      MOV  video/*
AddIcon        /httpd-internal-icons/sound.xbm      AU   audio/*
AddIcon        /httpd-internal-icons/tar.xbm        TAR  multipart/*tar
AddIcon        /httpd-internal-icons/compressed.xbm CMP  x-compress x-gzip
```

On Proxy Server

On proxy server the icon URLs **must be full URLs**, because otherwise clients would translate them relative to remote host. This means that in the above example all the `AddIcon*` directives have to read:

```
AddIcon  http://your.server/httpd-internal-icons/...
```

and you have to pass also the full icon URL:

```
Pass  http://your.server/httpd-internal-icons/*  /server_root/icons/*
```

Since future smart browsers might notice that the icon server is the same one as the proxy server it may be best in this case to also `Pass` the partial URL as above:

```
Pass  /httpd-internal-icons/*  /server_root/icons/*
```

4.5.2 HostName

On some hosts the hostname lookup fails producing only the name without the domain part. Full hostname is necessary when `httpd` is generating references to itself (redirection responses to clients). If necessary, provide full server hostname with `HostName` directive:

```
HostName   full.server.host.name
```

You may want to use this also when the real host name is different from what you want the clients to see (you have a DNS alias for the host).

4.5.3 Default Port Setting

For standalone server (the one running continuously, listening to a certain port, and forking a child to handle the request) the port to listen to can be defined via `Port` configuration directive instead of the `-p` *port* command line option. Normally:

```
Port 80
```

`-p` *port* command line line option still overrides this default.

4.5.4 PidFile

`httpd` re-reads its configuration file when it receives a `HUP` signal [HANGUP], the signal number 1. To make it easy to find out the parent `httpd` process id, it writes it to a file.

By default, if `ServerRoot` is specified, this is the file `httpd-pid` under server root; if not, it defaults to `/tmp/httpd-pid`.

The `PidFile` directive can be used to set the process id file name; it can be either an absolute path, or a relative one. Relative path is relative to `ServerRoot`, or if not defined, relative to `/tmp`.

Example

```
ServerRoot   /Web/serverroot
PidFile      logs/httpd-pid
```

would cause the process id to be written to `/Web/serverroot/logs/httpd-pid`.

4.5.5 Default User Id

`UserId` directive sets the default user to run as instead of `nobody`. This directive is only meaningful when running server as `root`.

```
UserId whoever
```

4.5.6 Default Group Id

`GroupId` directive sets the default group to run under instead of `nogroup`. This directive is only meaningful when running server as `root`.

```
GroupId whichever
```

4.5.7 Enabling and Disabling HTTP Methods

You can enable/disable methods that you do/don't want your server to accept:

```
Enable  METHOD
Disable METHOD
```

By default `GET`, `HEAD` and `POST` are enabled, and the rest are disabled.

Examples

```
Enable POST
Disable DELETE
```

4.5.8 IdentityCheck

If `IdentityCheck` configuration directive is turned `On`, `httpd` will connect to the ident daemon (RFC931) of the remote host and find out the remote login name of the owner of the client socket. This information is written to access log file, and put into the `REMOTE_IDENT` CGI environment variable.

Default setting is `Off`:

```
IdentityCheck Off
```

and if you don't need this information you will save the resources by keeping it off. Furthermore, this information does not provide any more security and should not be trusted to be used in access control, but rather just for informational purposes, such as logging.

WARNING

On some systems there is a kernel bug that causes all the connections to the remote node to be broken if the remote ident request is not answered (ident daemon not running, for example). This is reported for at least SunOS 4.1.1, NeXT 2.0a, ISC 3.0 with TCP 1.3, and AIX 3.2.2, and later are ok. Sony News/OS 4.51, HP-UX 8-?? and Ultrix 4.3 still have this bug. A fix for Ultrix is availabe (CSO-8919).

[Thanks to Per-Steinar Iversen from Norway for pointing this out!]

If the operating system on your server host has this bug, **do not use IdentityCheck!**

4.5.9 Welcome

`Welcome` directive specifies the default file name to use when only a directory name is specified in the URL. There may be many `Welcome` directives giving alternative welcome page names. The one that was defined earlier will have precedence.

Default values are `Welcome.html`, `welcome.html` and `index.html`. `index.html` is there only for compatibility with NCSA server; the word "Welcome" is more descriptive, and has precedence.

All default values will be overridden if `Welcome` directive is used.

Default values could be defined as:

```
Welcome Welcome.html
Welcome welcome.html
Welcome index.html
```

4.5.10 AlwaysWelcome

By default there is no difference between directory names with and without a trailing slash when it comes to welcome pages. The one without a trailing slash will cause an automatic redirection to the one with a trailing slash, which then gets mapped to the welcome page.

If it is desirable to have plain directory names to produce a directory listing, and only the ones with a trailing slash cause the welcome page to be returned, set the `AlwaysWelcome` directive to off:

```
AllwaysWelcome Off
```

Default value is `On`.

4.5.11 User-Supported Directories

User-supported directories, URLs of form **/ũsername**, are enabled by `UserDir` directive:

```
UserDir dir-name
```

The *dir-name* argument is the directory in each user's home directory to be exported, for example `WWW`:

```
UserDir WWW
```

4.5.12 Meta-Information

It is possible to tell `httpd` to add meta-information to response. Meta-information is stored in a directory specified by `MetaDir` directive, under the same directory as the file being retrieved:

```
MetaDir  dir-name
```

Meta-information is stored in a file with the same name as the actual document, but appended with a suffix specified via `MetaSuffix` directive:

```
MetaSuffix  .suffix
```

Meta-information files contain RFC822-style headers.

Default settings are:

```
MetaDir    .web
MetaSuffix .meta
```

meaning that meta-information files are located in the `.web` subdirectory, and they end in `.meta` suffix, i.e. the metafile for file:

```
/Web/Demo/file.html
```

would be:

```
/Web/Demo/.web/file.html.meta
```

4.5.13 MaxContentLengthBuffer

`httpd` normally gives a content-lenght header line for every document it returns. When it's running as a proxy it buffers the document received from the remote server before sending it to the client. This directive can be used to set the value of this buffer - if it is exceeded the document will be returned without a content-lenght header field.

Default setting is 50 kilobytes:

```
MaxContentLengthBuffer 50 K
```

4.6 Rules In The Configuration File

Rules define the mapping between virtual URLs and physical file names. Currently the following rules are understood:

- `Map` - Map URLs to actual files
- `Pass` - Accept a request
- `Fail` - Fail a request
- `Redirect` - Redirect a request
- `Protect` - Set up protection
- `DefProt` - Default protection setup
- `Exec` - Executable server scripts

4.6.1 Mapping, Passing and Failing

There are three main rules: `Map,` `Pass` and `Fail.` The server uses the top rule first, then **each successive rule** unless told otherwise by a `Pass` or a `Fail` rule.

`Map` ***template result***	If the address matches the *template*, use the *result* string from now on for future rules.
`Pass` ***template***	If the address maches the *template*, use it as it is, porocessing no further rules.
`Pass` ***template result***	If the string matches the *template*, use the *result* string as it is, processing no futher rules.
`Fail` ***template***	If the address matches the *template*, prohibit access, processing no futher rules.

The *template* string may contain wildcards (asterisks) `*.` (Versions earlier than 3.0 support only a single wildcard.) The *result* string may have wildcards only if the *template* has them. In this case they expand to matched strings in respective order.
Whitespace, (literal) asterisks and backslashes are allowed in templates if they are preceded by a backslash.
The tilde character (see user-supported directories) just after a slash (in other words in the beginning of a directory name) has to be explicitly matched, i.e. wildcard does not match it.
When matching,

- Rules are scanned from the top of the file to the bottom.
- If a request matches a `Map` template exactly, the result string is used instead of the original string and applied to successive rules.
- If the request maches a `Map` *template* with wildcard, then the text of the request which matches the wildcard is inserted in place of the wildcard in the *result* string to form the translated request. If the result string has no wildcard, it is used as it is.
- When a `Map` substitution takes place, the rule scan continues with the next rule using the new string in place of the request. This is not the case if a `Pass` or `Fail` is matched: they terminate the rule scan.

4.6.2 Redirecting Requests Elsewhere

When documents, or entire trees of documents, are moved from one server to another, you can use `Redirect` rule to tell `httpd` to redirect the request to another server. If the client program is smart enough user won't even notice that the document is retrieved from a different server.

`Redirect` ***template result*** Document matching *template* is redirected to *result*, which must be a **full URL** (i.e. containing `http:` and the host name).

Example

```
Redirect  /hypertext/WWW/*  http://www.cern.ch/WebDocs/*
```

This redirects everything starting with `/hypertext/WWW` to host `www.cern.ch` into virtual directory `/Web-Docs`. For example, `/hypertext/WWW/TheProject.html` would be redirected to `http://www.cern.ch/WebDocs/ThePr`

4.6.3 Setting Up User Authentication and Document Protection

Documents are protected by `Protect` and `DefProt` rules. Their syntax is the following:

`DefProt` ***template setup-file*** **`[`*uid.gid*`]`** Any document matching the *template* is associated with protection *setup-file*. The documents are not yet taken to be protected, but they may become protected by an existing access control list file in the same directory as the requested file, or by later matching a `Protect` rule. If that `Protect` rule doesn't specify *setup-file*, the one from the latest `DefProt` rule is used.

`Protect` **`[`*template setup-file* `[`*uid.gid*`]]`** Any document matching *template* is protected. The type of protection is defined in finer detail in *setup-file*.
If *setup-file* is not specified the one from previous matched `DefProt` rule will be used.
If none have matched access to the file is forbidden.

setupfile is always a full pathname for the protection setup file which specifies the actual protection parameters.

Setup file can be omitted from `Protect` rule, but it is obligatory in `DefProt` rule. If setup file is omitted it is not possible to give the *uid.gid* part, either.

uid.gid are the Unix user id and group id (either by name or by number, separated by a dot) to which the server should change when serving the request. These are only meaningful when the server is running as `root.` If they are missing they default to *nobody.nogroup*.

Note: Uid and gid are inherited from `DefProt` rule to `Protect` rule **only** when the *setup-file* is also inherited. If *setup-file* is specified for `Protect` rule but *uid.gid* is not, they default to *nobody.nogroup* regardless of the previous `DefProt` rule.

This is to avoid accidentally running the server under wrong user id with wrong setup file. This information should logically go into the protection setup file, but for safety reasons it cannot be done, because a non-trustworthy collaboration could specify it to be `root`. This way only the main `webmaster` can control user and group ids.

4.6.4 Executable Server Scripts

Document address is mapped into a script call by `Exec` rule:

```
Exec template script
```

In both *template* and *script* there **must be a `*` wildcard, that matches everything starting from the script filename.** This is to enable `httpd` to know what is the script name and what is the extra path information to be passed to the script.

Example

You want to map everything starting with `/your/url/doit` to execute the script `/usr/etc/www/htbin/doit.` You do this by saying:

```
Exec  /your/url/*  /usr/etc/www/htbin/*
```

Here asterisk mathes the script name `doit` (and everything else that follows it). Usually people use some fixed keyword in front of the pathname in URL to point out that the document is actually a script call. Often this keyword is `/htbin`. That is, usually your `Exec` rule looks like this:

```
Exec   /htbin/*   /usr/etc/www/htbin/*
```

and all the URLs pointing to the scripts start with `/htbin`, for example `/htbin/doit` in the previous example.

Historical Note (HTBin Rule)

CERN `httpd` versions 2.13 and 2.14 had a hard-coded handling of URL pathnames starting `/htbin` that mapped them to scripts in a directory specified via `HTBin` rule:

```
HTBin /your/htbin/directory
```

This is still handled automatically by `httpd`, by translating it to its equivalent `Exec` form:

```
Exec /htbin/*   /your/htbin/directory/*
```

Always use `Exec` instead −− it is more general.

4.7 Suffix Definitions for CERN httpd

`cern_httpd` uses suffixes to discover the content-type, content-encoding and content-language of a file. Default values are so extensive that `httpd` knows the usual file types. The following configuration directives can be used to add new suffix bindings and override existing defaults:

- `AddType` - Filename suffix mappings to MIME Content-Types
- `AddEncoding` - Filename suffix mappings to MIME Content-Encodings
- `AddLanguage` - Multilanguage support, suffix mappings to different Content-Languages
- `SuffixCaseSense` - Set suffix case sensitivity

4.7.1 Binding Suffixes to MIME Content-Types

As well as any mapping lines in the rule file, the rule file may be used to define the data types of files with particular suffixes. CERN `httpd` has an extensive set of predefined suffixes, so usually you don't need to specify any.
The syntax is:

```
AddType .suffix representation encoding [quality]
```

The parameters are as follows:

suffix The last part of the filename. There are two special cases. `*.*` matches to all files which have not been matched by any explicit suffixes but do contain a dot. `*` by itself matches to any file which does not match any other suffix.

representation A MIME Content-Type style description of the repreentation in fact in use in the file. See the HTTP spec. This need not be a real MIME type - it will only be used if it matches a type given by a client.

encoding A MIME content transfer encoding type. Much more limited in variety than representations, basically whether the file is ASCII (7bit or 8bit) or binary. A few other encodings are allowed, and maybe extension to compression.

quality Optional. A floating point number between 0.0 and 1.0 which determines the relative merits of files `xxx.*` which differ in their suffix only, when a link to `xxx.multi` is being resolved. Defaults to 1.0.

Examples

```
AddType .html text/html               8bit     1.0
AddType .text text/plain              7bit     0.9
AddType .ps   application/postscript  8bit     1.0
AddType *.*   application/binary      binary   0.1
AddType *     text/plain              7bit
```

Historical Note (Suffix Directive)

`AddType` was previously called `Suffix`. The old name is still understood, but may be misleading since suffixes are also used to determine Content-Encoding and language. Always use `AddType` instead.

4.7.2 Binding Suffixes to MIME Content-Endocings

Suffixes are also used to determine the Content-Encoding of a file (`.Z` suffix for `x-compressed`, for example). Syntax is:

```
AddEncoding .suffix  encoding
```

Example

```
AddEncoding .Z  x-compress
```

4.7.3 Multilanguage Support

Multilanguage support is also built on using suffixes to determine the language of a document. Suffix is bound to a language by `AddLanguage` rule (`.en` suffix for english, for example). Syntax is:

```
AddLanguage .suffix  encoding
```

Examples

```
AddLanguage .en  en
AddLanguage .uk  en_UK
```

4.7.4 Suffix Case Sensitivity

Suffix case sensitivity is by default *off*. You can make suffixes case sensitive with `SuffixCaseSense` directive:

```
SuffixCaseSense On
```

4.8 Accessory Scripts

In addition to having a fully configurable CGI script interface to handle form requests, CERN `httpd` has a few special directives to handle certain tasks always via CGI scripts:

- keyword searches
- general `POST`
- general `PUT`
- general `DELETE`

4.8.1 Keyword Search Facility

Server automatically calls a script to perform search, if the **absolute pathname** of search script is supplied by a `Search` directive in the configuration file:

```
Search /search/script/pathname
```

This script is called with the vital information in the following CGI environment variables:

PATH_INFO	contains the virtual URL of the file from where the query was issued from.
PATH_TRANSLTED	contains the physical filename of the document corresponding to the virtual URL in `PATH_INFO`.
QUERY_STRING	contains the (URL-encoded) keywords, which are also available decoded as command line parameters, one in each of `argv[1]`, `argv[2]`, ...

Search script must conform to CGI/1.1 rules, that is, it has to start its output with a MIME header **followed by a blank line**, after which comes the actual document. MIME header **must** contain either a `Location:` field, or a `Content-Type:` field, typically:

```
Content-Type: text/html
```

if the document is an HTML document.

4.8.2 General POST Method Handler Script

`POST` requests are handled by calling the script defined by `POST-Script` directive:

```
POST-Script  /absolute/path/post-handler
```

POST handler script is called in the normal CGI manner, and its output must be CGI compliant.

Only such `POST` requests are handled by the POST handler that haven't already matched an `Exec` rule (which causes a specified script to be called).

4.8.3 General PUT Method Handler Script

`PUT` requests are handled by calling the script defined by `PUT-Script` configuration directive:

```
PUT-Script  /absolute/path/put-handler
```

PUT handler script is called in the normal CGI manner, and its output must be CGI compliant.

By default `PUT` method is disabled; you must explicitly enable it in the configuration file:

```
Enable PUT
```

This is to enhance security.

Since `PUT` can be a very dangerous method because it allows files to be written back to the server, it is not possible to use `PUT` without access authorization module being activated. This means that you have to have at least a `DefProt` rule specifying a default protection setup, which then in turn defines the `PutMask` containing the list of allowed users and hosts to perform PUT operation.

4.8.4 General DELETE Method Handler Script

DELETE requests are handled by calling the script defined by DELETE-Script configuration directive:

```
DELETE-Script   /absolute/path/put-handler
```

DELETE handler script is called in the normal CGI manner, and its output must be CGI compliant.
By default PUT method is disabled; you must explicitly enable it in the configuration file:

```
Enable DELETE
```

This is to enhance security.
Since DELETE can be a very dangerous method because it allows files to be deleted from the server, it is not possible to use DELETE without access authorization module being activated. This means that you have to have at least a DefProt rule specifying a default protection setup, which then in turn defines the DeleteMask containing the list of allowed users and hosts to perform DELETE operation.

4.9 Directory Browsing

By default references to directories which don't include a welcome page cause httpd to generate a hypertext view of the directory listing. There are numerous configuration directives controlling this feature:

- DirAccess - Enable/Selective/Disable directory listings
- DirReadme - Configure/disable README-feature
- Controlling the appearance of directory listings:
  - DirShowIcons - Show icons in directory listings
  - DirShowDate - show last-modified date
  - DirShowSize - show file sizes
  - DirShowBytes - show byte count for small files
  - DirShowDescription - show descriptions for files
  - DirShowMaxDescrLength - maximum description length
  - DirShowBrackets - use brackets around ALTernative text used instead of an icon
  - DirShowMinLength - minimum width to reserve for filenames
  - DirShowMaxLength - maximum width to reserve for filenames
  - DirShowHidden - show also files starting with a dot (hidden Unix files)
  - DirShowOwner - show owner of the file
  - DirShowGroup - show group of the file
  - DirShowMode - show permissions of the file
  - DirShowCase - do sorting in a case-sensitive manner
- Icons:
  - AddIcon - bind icon URL to a MIME Content-Type or Content-Encoding
  - AddBlankIcon - icon URL used in the heading of the listing to align it
  - AddUnknownIcon - icon URL for unknown file types
  - AddDirIcon - icon URL for directories
  - AddParentIcon - icon URL for parent directory

4.9.1 Controlling Directory Browsing

`DirAccess on` Enable directory browsing in all directories (which are not forbidden by rules). Synonym with `-dy` command line option. *Default.*

`DirAccess off` Disable directory browsing. Synonym with `-dn` command line option.

`DirAccess selective` Enable selective directory browsing - only directories containing the file `.www_browsable` are allowed. Synonym with `-ds` command line option.

4.9.2 README Feature

`DirReadme top` For any browsable directeory containing a `README` file, include the text at the top of the directory listing. Synonym with `-dt` command line option. *Default.*

`DirReadme bottom` Same as previous, but contents of `README` appear on the bottom. Synonym with `-db` command line option.

`DirReadme off` Disables the `README` inclusion feature. Synonym with `-dr` command line option.

4.9.3 Controlling The Look of Directory Listings

The following `On/Off` directives control how the directory listings look like. The default is to show icons, use brackets around ALTernaltive text, show last-modifid, size and description, and allow filename field width to vary between 15-22 characters, and reserve 25 characters for description.

`DirShowIcons` Generate inlined image calls in front of each line. Icons visualize the content-type of the file, and they are defined by `AddIcon` configuration directive. *Default.*

`DirShowDate` Show last modification date. *Default.*

`DirShowSize` Show the size of files. *Default.*

`DirShowBytes` By default files smaller than 1K are shown as just 1K. Setting this directive to `On` will cause the exact byte count to appear.

`DirShowDescription` Show description if available. *Default.*
At the time of release of 2.17 there was no consensus about where the descriptions come from, and the mechanism is currently undocumented. For HTML files description it the TITLE element; for other files the description field is left empty.

`DirShowMaxDescrLenght` The maximum number of characters to show in the description field.

`DirShowBrackets` Use brackets around ALTernative text used by browsers not capable of displaying images. *Default.*

`DirShowHidden` Show hidden Unix files (the ones starting with a dot).

`DirShowOwner` Show the owner of the file.

`DirShowGroup` Show the group of the file.

`DirShowMode` Show the permissions of files.

`DirShowCase` Sort entries in a case-sensitive manner, i.e. all capital letters before lower-case letters.

4.9.4 Filename Length

There is a minimum and maximum width for the filename field. Entries longer than the maximum value will be truncated. Default values are 15 and 25, and they can be changed with these directives:

`DirShowMinLength` ***num*** At least this amount of characters is always reserved for filenames. If the longest filename in the directory is longer than *num* the field will be extended, but no more than the maximum limit (see next directive).

`DirShowMaxLength` ***num*** Filenames longer than *num* will be truncated to fit in length.

Example

The default values would be set by saying:

```
DirShowMinLength  15
DirShowMaxLength  25
```

4.10 Icons In The Directory Listings

`cern_httpd` directory icons are used, if enabled, for both regular directory listings, and FTP listings (when runnins as a proxy).

- `AddIcon` - bind icon URL to a MIME Content-Type or Content-Encoding
- `AddBlankIcon` - icon URL used in the heading of the listing to align it
- `AddUnknownIcon` - icon URL for unknown file types
- `AddDirIcon` - icon URL for directories
- `AddParentIcon` - icon URL for parent directory

These directives are specified in the configuration file.

4.10.1 AddIcon Directive

The `AddIcon` directive binds an icon to a MIME Content-Type or Content-Encoding:

```
AddIcon  icon-url ALT-text template
```

icon-url	is the URL of the icon.
ALT-text	is the alternative text to use on character terminal browsers.
template	is either a Content-Type template or a Content-Encoding template. Content-Type template must always contain a slash, whereas Content-Encoding template never has it.

The following important remarks serve also as examples.

CERN httpd as a Normal HTTP Server

Understand that the *icon-url* is a virtual URL - one that will be translated through the rules. Therefore you must make sure that your configuration rules allow the icon URLs to be passed, e.g.:

```
AddIcon  /icons/UNKNOWN.gif  ???  */*
AddIcon  /icons/TEXT.gif     TXT  text/*
AddIcon  /icons/IMAGE.gif    IMG  image/*
AddIcon  /icons/SOUND.gif    AU   audio/*
AddIcon  /icons/MOVIE.gif    MOV  video/*
AddIcon  /icons/PS.gif       PS   application/postscript
Pass /icons/*  /absolute/icon/dir/*
...other rules...
```

CERN httpd as a Proxy

When using `httpd` as a proxy the icon URL **must be** an absolute URL pointing to your server; otherwise clients would translate it relative to the remote host.

Furthermore, you must have a mapping from this absolute URL to your local file system, e.g.:

```
AddIcon  http://your.server/icons/UNKNOWN.gif  ???  */*
AddIcon  http://your.server/icons/TEXT.gif     TXT  text/*
AddIcon  http://your.server/icons/IMAGE.gif    IMG  image/*
AddIcon  http://your.server/icons/SOUND.gif    AU   audio/*
AddIcon  http://your.server/icons/MOVIE.gif    MOV  video/*
AddIcon  http://your.server/icons/PS.gif       PS   application/postscript

Pass http://your.server/icons/*  /absolute/icon/dir/*
Pass /icons/*                    /absolute/icon/dir/*
Pass http:*
Pass ftp:*
Pass gopher:*
```

Both the full and partial icon URLs are `Pass`'ed because smart clients may be configured to connect to local servers directly, instead of through the proxy, and in that case the proxy server (which is then just a normal HTTP server from client's point of view) will be requested for `/icons/...` instead of `http://your.server/icons/...`. The proxy server has no way of knowing which will happen.

4.10.2 Icons in Gopher Listings

There are special internal (to `httpd`) MIME content types that can be bound to icons for gopher listings (the names should be self-explanatory):

- `application/x-gopher-index`
- `application/x-gopher-cso`
- `application/x-gopher-telnet`
- `application/x-gopher-tn3270`
- `application/x-gopher-duplicate`

4.10.3 Special Icons

`httpd` needs some special icons:

`AddBlankIcon` Icon URL used in the heading of the listing to align it. This is typically a blank icon, but may contain some nice image that you wish to have on top of all your listings. The only criterion is that it must be the same size as the other icons.

`AddUnknownIcon` Icon URL used for unknown file types, i.e. files for which no other icon binding applies. If you have an exhaustive set of `AddIcon` directives this needs not be used.

`AddDirIcon` Icon URL for directories.

`AddParentIcon` Icon URL for parent directory.

Example For a Regular HTTP Server

Remember to `Pass` the icon URLs!

```
        AddBlankIcon    /icons/BLANK.gif
        AddUnknownIcon  /icons/UNKNOWN.gif  ???
        AddDirIcon      /icons/DIR.gif      DIR
        AddParentIcon   /icons/PARENT.gif   UP

Pass  /icons/*  /absolute/icon/dir/*
        ...other rules...
```

Example For a Proxy Server

Icon URLs **must be absolute URLs**, and you must have a mapping from the absolute form to local form, and remember to `Pass` them:

```
        AddBlankIcon    http://your.server/icons/BLANK.gif
        AddUnknownIcon  http://your.server/icons/UNKNOWN.gif  ???
        AddDirIcon      http://your.server/icons/DIR.gif      DIR
        AddParentIcon   http://your.server/icons/PARENT.gif   UP

        Pass http://your.server/icons/*  /absolute/icon/dir/*
        Pass /icons/*                    /absolute/icon/dir/*
        Pass  http:*
        Pass  ftp:*
        Pass  gopher:*
```

4.11 Logging Control In CERN httpd

`cern_httpd` logs all the incoming requests to an access log file. It also has an error log where internal server errors are logged.

- `AccessLog` - Set access log file name
- `ErrorLog` - Set error log file name
- `LogFormat` - Set access log file format
- `LogTime` - Set time zone for log files
- `NoLog` - No log entries for listed hosts/domains
- `CacheAccessLog` - Log cache accesses to a different log file

4.11.1 Access Log File

Access log file contains a log of all the requests. The name of the log file is spesified either by `-l` *logfile* command line option, or with `AccessLog` directive:

```
        AccessLog /absolute/path/logfile
```

4.11.2 Error Log File

Error log contains a log of errors that might prove useful when figuring out if something doesn't work. Error log file name is set by `ErrorLog` directive:

```
ErrorLog /absolute/path/errorlog
```

If error log file is not specified, it defaults to access log file name with `.error` extension. If the filename extension already exists, `.error` will replace it.

4.11.3 Log File Format

Previously every server used to have its own logfile format which made it difficult to write general statistics collectors. Therefore there is now a *common logfile format* (which will eventually become the default). Currently it is enabled by

```
LogFormat  Common
```

The old CERN `httpd` format can be used by

```
LogFormat  Old
```

4.11.4 Log Time Format

Times in the log file are by default local time. That can be changed to be GMT time by `LogTime` directive:

```
LogTime  GMT
```

Default is:

```
LogTime  LocalTime
```

4.11.5 Suppressing Log Entries For Certain Hosts/Domains

It's not always necessary to collect log information of accesses made by local hosts. The `NoLog` directive can be used to prevent log entry being made for hosts matching a given IP number or host name template:

```
NoLog  template
```

Examples

```
NoLog 128.141.*.*
NoLog *.cern.ch
NoLog *.ch  *.fr  *.it
```

4.12 Timeout Settings

Something may go wrong with the connection to the client causing `httpd` to hang infinitely doing nothing. This can be avoided by setting timeouts on different tasks that the server performs. All of these timeouts have relatively good default values by default and they don't usually need to be changed.
All the times for these directives are of form:

```
45 secs
10 mins
2 mins 30 secs
1 hour
```

4.12.1 InputTimeOut

`InputTimeOut` diretictive specifies the time to wait for the client to send the request (the MIME-header part of it, not the message body). Default value is:

```
InputTimeOut  2 mins
```

4.12.2 OutputTimeOut

`OutputTimeOut` diretictive specifies the time to allow for sending the response. Default value is:

```
OutputTimeOut  20 mins
```

If you are serving huge files for clients behind slow connections you may want to increase this value if you hear of connections being cut in the middle of transfer.

4.12.3 ScriptTimeOut

`ScriptTimeOut` diretictive specifies the time to allow for server scripts to finish. If a script doesn't return in the time specified `httpd` will send `TERM` and `KILL` signals to it (with 5 seconds in between to let scripts do cleanup upon exit). Default value is:

```
ScriptTimeOut  5 mins
```

4.13 Proxy Caching

When `cern_httpd` is run as a proxy it can perform caching of the documents retrieved from remote hosts to make futher requests faster.

- `Caching` - Turn caching on
- `CacheRoot` - Set cache root directory for a proxy server
- `CacheSize` - Specify cache size (in megabytes)
- `NoCaching` - No caching for URLs matching a given mask
- `CacheOnly` - Cache only if URL matches a given set of URLs
- `CacheClean` - Remove everything older than this (in days)
- `CacheUnused` - Remove if has been unused this long (in days)
- `CacheDefaultExpiry` - Default expiry time if not given by remote server (in days)
- `CacheLastModifiedFactor` - Factor used in approximating expiry date
- `CacheTimeMargin` - Time accuracy between hosts
- `CacheNoConnect` - Standalone cache mode - no external document retrievals
- `CacheExpiryCheck` - Turn off expiry checking for standalone operation
- `Gc` - Enable and disable garbage collection
- `GcDailyGc` - Time for daily garbage collection

- `GcTimeInterval` - Interval to do cache garbage collection (in hours)
- `GcReqInterval` - Number of requests between garbage collections
- `GcMemUsage` - Garbage collector memory usage directive
- `CacheLimit_1` - First cache file size limit (kilobytes)
- `CacheLimit_2` - Second cache file size limit (kilobytes)
- `CacheLockTimeOut` - Break cache locks after this timeout
- `CacheAccessLog` - Log cache accesses to a different log file

4.13.1 Turning Caching On and Off

Caching is normally turned implicitly on by specifying the Cache Root Directory, but it can be explicitly turned on and off by `Caching` directive:

```
Caching On
```

4.13.2 Setting Cache Directory

Caching is enabled on a server running as a gateway (proxy) by `CacheRoot` directive, which is used to set the absolute path of the cache directory:

```
CacheRoot /absolute/cache/directory
```

4.13.3 Cache Size

`CacheSize` directive sets the maximum cache size in megabytes. Default value is 5MB, but its preferable to have several megabytes of cache, like 50-100MB, to get best results. Cache may, however, temporarily grow a few megabytes bigger than specified.

Example

```
CacheSize 20 M
```

sets cache size to 20 megabytes.

4.13.4 NoCaching

URLs matching a template given by `NoCaching` directive will never be cached, e.g.:

```
http://really.useless.site/*
```

From version 3.0 on templates can have any number of wildcard characters `*`.

4.13.5 CacheOnly

Only the URLs matching templates given by `CacheOnly` directives will be cached, e.g.:

```
http://really.important.site/*
```

From version 3.0 on templates can have any number of wildcard characters `*`.

4.13.6 Maximum Time to Keep Cache Files

All cached documents matching a specified template and that are older than specified by `CacheClean` directive will be removed. This value overrides expiry date in that no file can be stored longer than this value specifies, regardless of expiry date.

Examples

```
CacheClean http:*      1 month
CacheClean ftp:*      14 days
CacheClean gopher:*    5 days 12 hours
```

4.13.7 Maximum Time to Keep Unused Files

Cache files matching a template and having been unused longer than specified by `CacheUnused` directive will be removed.

Examples

```
CacheUnused *                       4 days 12 hours
CacheUnused http://info.cern.ch/*   7 days
CacheUnused ftp://some.server/*    14 days
```

Note that the last matching specification will have precedence; therefore HTTP files from `info.cern.ch` will be kept 7 days, and **not** 4.5 days.

4.13.8 Default Expiry Time

Files for which the server gave neither `Expires:` nor `Last-Modified:` header will be kept at most the time specified by `CacheDefaultExpiry` directive. Default values are zero for HTTP (script replies shouldn't be cached), and 1 day for FTP and Gopher.

Example

```
CacheDefaultExpiry ftp:*     1 month
CacheDefaultExpiry gopher:*  10 days
```

Default expiry for HTTP will almost always cause problems because there are currently many scripts that don't give an expiry date, yet their output expires immediately. Therefore, it is better to keep the default value for `http:` in zero.

4.13.9 CacheLastModifiedFactor

Currently HTTP servers give usually only the `Last-Modified` time, but not `Expires` time. `Last-Modified` can often be successfully used to approximate expiry date. `CacheLastModifiedFactor` gives the fraction of time since last modification to give the remaining time to be up-to-date.

Default value is `0.1`, which means that e.g. file modified 20 days ago will expire in 2 days.

Examples

```
CacheLastModifiedFactor  0.2
```

would cause files modified 5 months ago to expire after one month.
This feature can be turned off by specifying:

```
CacheLastModifiedFactor  Off
```

4.13.10 CacheTimeMargin

Sometimes inaccurate times on other hosts cause confusion in caching. It often also makes sense not to cache documents that will expiry in a couple of minutes anyway. `CacheTimeMargin` defines this time margin, by default:

```
CacheTimeMargin  2 mins
```

No document expiring in less than two minutes will be written to disk.

4.13.11 CacheNoConnect

This directive puts proxy to standalone cache mode, i.e. only the documents found in the cache are returned, and ones no in the cache will return error rather than connection to the outside world. This is useful for demo-purposes and in other cases without network connection:

```
CacheNoConnect On
```

Default setting is naturally `Off`.
This directive is typically used with expiry checking also turned `Off`.

4.13.12 CacheExpiryCheck

If (for demo-reasons etc) it's desired that the proxy always returns documents from the cache, even if they have expired, `CacheExpiryCheck` can be turned off:

```
CacheExpiryCheck  Off
```

Default setting is `On`, meaning that proxy never returns an expired document.
This is usually used in standalone cache mode (`CacheNoConnect` diretive turned `On`).

4.13.13 Garbage Collection

When caching is enabled garbage collection is also activated by default. This can be explicitly turned off with `Gc` directive:

```
Gc  Off
```

4.13.14 When to Do Garbage Collection

Garbage collection is launched right away when cache size limit is reached. However, to keep cache smaller it might be desirable to remove expired files even if there is still cache space remaining. It is possible to to launch garbage collection at a certain time, usually outside the busy hours:l

```
GcDailyGc      time
```

`GcDailyGc` specifies the time to do daily garbage collection, normally during the night. Default value is 3:00. Daily garbage collection can be disabled by specifying `Off`.

Example

Default value would be specified as:

```
GcDailyGc       3:00
```

Another example: turning daily gc off:

```
GcDailyGc       Off
```

4.13.15 Memory Usage of Garbage Collector

Garbage collector performs its job best if if can read information about the whole cache into memory at once. This is not possible if the machine doesn't have enough main memory.

`GcMemUsage` directive advices garbage collector about how much memory to use. You may imagine this is the number of kilobytes to use for gc data, but it may vary greatly according to dynamic things, like the directory structure of cached files.

Default is 500; if gc fails because memory runs out make this smaller. If your machine has so much memory that it just can't run out, make this very big.

Example

```
GcMemUsage 100
```

if you have very little memory.

4.13.16 Cache File Sizes

There are two limits controlling the size factor of a file when its value is being calculated. `CacheLimit_1` sets the lower limit; under this all the files have equal size factor. `CacheLimit_2` sets up higher limit; files bigger than this get extremely bad size factor (meaning they get removed right away because they are too big).

Sizes are specified in kilobytes, and defaults values are 200K and 4MB, respectively.

Examples

```
CacheLimit_1 200 K
CacheLimit_2 4000 K
```

would set the same values as the defaults, 200K and 4MB.

4.13.17 Cache Lock Timeout

During retrieval cache files are locked. If something goes wrong a lock file may be left hanging. `CacheLockTimeOut` directive sets the amount of time after which lock can be broken. Time is specified like all the other times in the configuration file, and default value is 20 minutes, the same as default `OutputTimeOut`. **CacheLockTimeOut should never be less than OutputTimeOut!**

Example

```
CacheLockTimeOut   30 mins
```

would set lock timeout to half an hour.

4.13.18 CacheAccessLog

Cache accesses can be logged to a different log file instead of the normal access log. The `CacheAccessLog` directive takes an absolute pathname of the cache access log file:

```
CacheAccessLog  /absolute/path/file.log
```

4.14 Configuring Proxy To Connect To Another Proxy

If there is a need to make an (inner) proxy cern_httpd connect to the outside world via another (outer) proxy server, you can use the same environment variables as are used to redirect clients to the proxy to make inner proxy use the outer one:

- `http_proxy`
- `ftp_proxy`
- `gopher_proxy`
- `wais_proxy`

E.g. your (inner) proxy server's startup script could look like this:

```
#!/bin/sh
http_proxy=http://outer.proxy.server:8082/
export http_proxy
/usr/etc/httpd -r /etc/inner-proxy.conf -p 8081
```

This is a little ugly, so there are also the following directives in the configuration file:

- `http_proxy` *http://outer.proxy.server/*
- `ftp_proxy` *http://outer.proxy.server/*
- `gopher_proxy` *http://outer.proxy.server/*
- `wais_proxy` *http://outer.proxy.server/*

4.14.1 no_proxy

In the same way that clients can specify a set of domains for which the proxy should not be consulted, `httpd` has a `no_proxy` configuration directive to tell it that it should not connect to another proxy for certain URLs:

```
no_proxy  cern.ch,ncsa.uiuc.edu,some.host:8080
```

The argument string is a comma-separated list and should **not contain spaces!**

5. Configuration File Examples

httpd.conf	sample configuration file for running as a normal HTTP server.
prot.conf	sample configuration file for running as a normal HTTP server with access control.
proxy.conf	sample configuration file for running as a proxy **without caching.**
caching.conf	sample configuration file for running as a proxy **with caching.**

5.1 Normal HTTP Server Configuration

```
#
# Sample configuration file for cern_httpd for running it
# as a normal HTTP server.
#
# See:
# <http://info.cern.ch/hypertext/WWW/Daemon/User/Config/Overview.html>
#
# for more information.
#
# Written by:
# Ari Luotonen  April 1994  <luotonen@dxcern.cern.ch>
#

#
# Set this to point to the directory where you unpacked this
# distribution, or wherever you want httpd to have its "home"
#
ServerRoot /where/ever/server_root

#
# The default port for HTTP is 80; if you are not root you have
# to use a port above 1024; good defaults are 8000, 8001, 8080
#
Port 80

#
# General setup; on some systems, like HP, nobody is defined so
# that setuid() fails; in those cases use a different user id.
#
UserId nobody
GroupId nogroup

#
# Logging; if you want logging uncomment these lines and specify
# locations for your access and error logs
#
# AccessLog /where/ever/httpd-log
# ErrorLog /where/ever/httpd-errors
LogFormat Common
LogTime LocalTime
```

```
#
# User-supported directories under ~/public_html
#
UserDir public_html

#
# Scripts; URLs starting with /cgi-bin/ will be understood as
# script calls in the directory /your/script/directory
#
Exec /cgi-bin/* /your/script/directory/*

#
# URL translation rules; If your documents are under /local/Web
# then this single rule does the job:
#
Pass /* /local/Web/*
```

5.2 Normal HTTP Server With Access Control

```
#
# Sample configuration file for cern_httpd for running it
# as a normal HTTP server WITH access control.
#
# See:
# <http://info.cern.ch/hypertext/WWW/Daemon/User/Config/Overview.html>
#
# for more information.
#
# Written by:
# Ari Luotonen  April 1994  <luotonen@dxcern.cern.ch>
#

#
# Set this to point to the directory where you unpacked this
# distribution, or wherever you want httpd to have its "home"
#
ServerRoot /where/ever/server_root

#
# The default port for HTTP is 80; if you are not root you have
# to use a port above 1024; good defaults are 8000, 8001, 8080
#
Port 80

#
# General setup; on some systems, like HP, nobody is defined so
# that setuid() fails; in those cases use a different user id.
#
```

```
UserId nobody
GroupId nogroup

#
# Logging; if you want logging uncomment these lines and specify
# locations for your access and error logs
#
# AccessLog /where/ever/httpd-log
# ErrorLog /where/ever/httpd-errors
LogFormat Common
LogTime LocalTime

#
# User-supported directories under ~/public_html
#
UserDir public_html

#
# Protection setup by usernames; specify groups in the group
# file [if you need groups]; create and maintain password file
# with the htadm program
#
Protection PROT-SETUP-USERS {
UserId nobody
GroupId nogroup
ServerId YourServersFancyName
AuthType Basic
PasswdFile /where/ever/passwd
GroupFile /where/ever/group
GET-Mask user, user, group, group, user
}

#
# Protection setup by hosts; you can use both domain name
# templates and IP number templates
#
Protection PROT-SETUP-HOSTS {
UserId nobody
GroupId nogroup
ServerId YourServersFancyName
AuthType Basic
PasswdFile /where/ever/passwd
GroupFile /where/ever/group
GET-Mask @(*.cern.ch, 128.141.*.*, *.ncsa.uiuc.edu)
}

Protect /very/secret/URL/*    PROT-SETUP-USERS
Protect /another/secret/URL/* PROT-SETUP-HOSTS
```

```
#
# Scripts; URLs starting with /cgi-bin/ will be understood as
# script calls in the directory /your/script/directory
#
Exec /cgi-bin/* /your/script/directory/*

#
# URL translation rules; If your documents are under /local/Web
# then this single rule does the job:
#
Pass /* /local/Web/*
```

5.3 Proxy Configuration With Caching

The configuration **without caching** is otherwise the same, just leave out all the directives starting with "`Cache`" or "`Gc`".

```
#
# Sample configuration file for cern_httpd for running it
# as a proxy server WITH caching.
#
# See:
# <http://info.cern.ch/hypertext/WWW/Daemon/User/Config/Overview.html>
#
# for more information.
#
# Written by:
# Ari Luotonen  April 1994  <luotonen@dxcern.cern.ch>
#

#
# Set this to point to the directory where you unpacked this
# distribution, or wherever you want httpd to have its "home"
#
ServerRoot /where/ever/server_root

#
# Set the port for proxy to listen to
#
Port 8080

#
# General setup; on some systems, like HP, nobody is defined so
# that setuid() fails; in those cases use a different user id.
#
UserId nobody
GroupId nogroup
```

```
#
# Logging; if you want logging uncomment these lines and specify
# locations for your access and error logs
#
# AccessLog /where/ever/proxy-log
# ErrorLog /where/ever/proxy-errors
LogFormat Common
LogTime LocalTime

#
# Proxy protections; if you want only certain domains to use
# your proxy, uncomment these lines and specify the Mask
# with hostname templates or IP number templates:
#
# Protection PROXY-PROT {
#  ServerId YourProxyName
#  Mask @(*.cern.ch, 128.141.*.*, *.ncsa.uiuc.edu)
# }
# Protect  *  PROXY-PROT

#
# Pass the URLs that this proxy is willing to forward.
#
Pass http:*
Pass ftp:*
Pass gopher:*
Pass wais:*

#
# Enable caching, specify cache root directory, and cache size
# in megabytes
#
Caching On
CacheRoot /your/cache/root/dir
CacheSize 5

#
# Specify absolute maximum for caching time
#
CacheClean * 2 months

#
# Specify the maximum time to be unused
#
CacheUnused http:* 2 weeks
CacheUnused ftp:* 1 week
CacheUnused gopher:* 1 week

#
```

```
# Specify default expiry times for ftp and gopher;
# NEVER specify it for HTTP, otherwise documents generated by
# scripts get cached which is usually a bad thing.
#
CacheDefaultExpiry ftp:* 10 days
CacheDefaultExpiry gopher:* 2 days

#
# Garbage collection controls; daily garbage collection at 3am;
#
Gc On
GcDailyGc 3:00
```

6. CERN Server CGI/1.1 Script Support

Server scripts are used to handle searches, clickable images and forms, and to produce synthesized documents on the fly. See calendar and finger gateway for examples.

6.1 In This Section...

- Using `Exec` rule to allow scripts
- CGI Interface −− Script Input
- CGI Interface −− Script Output
- NPH-Scripts −− No Parsing of Headers
- Setting up a search script

6.2 Important Note!

CERN `httpd` versions 2.15 and newer have **two** script interfaces. The other one is the official CGI, Common Gateway Interface, which enables scripts to be shared between different server implementations (NCSA server, Plexus, etc). The other one is the original, very easy-to-use, interface, that was introduced in version 2.13.

Use of CGI instead of the old interface is strongly encouraged.

IMPORTANT: If you have, or wish to write, scripts that use the old interface, your script name has to end in `.pp` suffix (comes from "Pre-Parsed"). URLs referring to these scripts should not contain this suffix. This is to make it easier to later upgrade to CGI scripts, so you only need to change the script name in the file system, and not the documents pointing to it. If you absolutely want to use the old interface (which is nice for quick hacks that don't need to be portable), see the doc.

6.3 Setting Up httpd To Call Scripts

The server knows that a request is actually a script request by looking at the beginning of the URL pathname. You can specify these special strings in the configuration file `(/etc/httpd.conf)` by `Exec` rules:

```
Exec /url-prefix/*  /physical-path/*
```

Where */url-prefix/* is the special string that signifies a script request, and */physical-path/* is the absolute filesystem pathname of the **directory** that contains your scripts.

6.3.1 Example

```
Exec  /htbin/*  /usr/etc/cgi-bin/*
```

makes URL paths starting with `/htbin` to be mapped to scripts in directory `/usr/etc/cgi-bin.` I.e. requesting

```
/htbin/myscript
```

causes a call to script

```
/usr/etc/cgi-bin
```

6.3.2 Historical Note

In `httpd` versions before 2.15 there was an `HTBin` directive:

```
HTBin  /physical-path
```

which is now obsolite, but understood by the server to mean

```
Exec  /htbin/*  /physical-path/*
```

Use of `Exec` rule instead is recommended for its generality.

6.4 Information Passed to CGI Scripts

CGI scripts get their input mainly from environment variables and standard input (when using `POST` method). Search scripts get keywords also as command line arguments.

Most important environment variables are:

QUERY_STRING The query part of URL, that is, everything that follows the question mark. This string is URL-encoded, meaning that special characters like spaces and newlines are encoded into their hex notation (%xx), and characters like `+ = &` have a special meaning. The contents of this variable can be easily parsed using the `cgiparse` program.

PATH_INFO Extra path information given after the script name, for example with `Exec` rule:

```
Exec  /htbin/*  /usr/etc/cgi-bin/*
```

a URL with path

```
/htbin/myscript/extra/pathinfo
```

will execute the script `/usr/etc/cgibin/myscript` with `PATH_INFO` environment variable set to `/extra/pathinfo`.

PATH_TRANSLATED Extra pathinfo translated through the rule system. (This doesn't always make sense.)

See also NCSA's primer to writing CGI scripts.

6.5 Results From Scripts

Scripts return their results either outputting a document to their standard output, or by outputting the location of the result document (either a full URL or a local virtual path).

6.5.1 Outputting a Document

Script result must begin with a `Content-Type:` line giving the document content type, followed by **an empty line**. The actual document follows the empty line. Example:

```
Content-Type: text/html

<HEAD>
<TITLE>Script test>
</HEAD>
<BODY>
<H1>My First Virtual Document</H1>
....
</BODY>
```

6.5.2 Giving Document Location

If the script wants to return an existing document (local or remote), it can give a `Location:` header followed by an empty line: Example:

```
Location: http://info.cern.ch/hypertext/WWW/TheProject.html
```

This causes the server to send a redirection to client, which then retrieves that document. If `Location` starts with a slash (is not a full URL), it is taken to be a virtual path for a document on the same machine, and server passes this string right away through the rule system and serves that document as if it had been requested in the first place. In this case clients don't do the redirection, but the server does it "on the fly".
Example:

```
Location: /hypertext/WWW/TheProject.html
```

Understand, that this is a **virtual path**, so after translations it might be, for example, `/Public/Web/TheProject.html`.
Important: Only **full** URLs in `Location` field can contain the *#label* part of URL, because that is meant only for the client-side, and the server cannot possibly handle it in any way.

6.5.3 NPH-Scripts (No-Parse-Headers)

Script wishing to output the entire HTTP reply (including status line and all response headers) should be named to begin with `nph-` prefix. This makes `httpd` connect script's output stream directly to requesting client reducing the overhead of server needlessly parsing the response headers.

Example Of NPH-Script Output

```
HTTP/1.0 200 Script results follow
Server: MyScript/1.0 via CERN/3.0
Content-Type: text/html

<HEAD>
<TITLE>Just testing...</TITLE>
</HEAD>
<BODY>
<H1>Output From NPH-Script</H1>
Yep, seems to work.
</BODY>
```

6.6 Setting Up A Search Script

There is a special `Search` directive in the configuration file givin the **absolute** pathname of the script performing the search:

```
Search /absolute/path/search
```

Every time a document is searched, this script is called with

Command line	containing the search keywords decoded, one in each of `argv[1]`, `argv[2]`, ...
`QUERY_STRING`	containing the query string encoded, as it came in the URL after the question mark.
`PATH_INFO`	Virtual path of the document that the search was issued from.

PATH_TRANSLATED Absolute filesystem path of the document.

Search results are output in the usual way:

```
Content-Type: text/html

...generated document...
```

7. cgiparse Manual

`cgiparse` handles `QUERY_STRING` environment variable parsing for CGI scripts. It comes with CERN server distributions **2.15** and newer.

If the `QUERY_STRING` environment variable is not set, it reads `CONTENT_LENGTH` characters from its standard input.

7.1 Command Line Options

7.1.1 Main Options

`cgiparse -keywords` Parse `QUERY_STRING` as search keywords. Keywords are decoded and written to standard output, one per line.

`cgiparse -form` Parse `QUERY_STRING` as form request. Outputs a string which, when `eval`'ed by Bourne shell, will set shell variables beginning with `FORM_` appended with field name. Field values are the contents of the variables.

`cgiparse -value` *fieldname* Parse `QUERY_STRING` as form request. Prints only the value of field *fieldname*.

`cgiparse -read` Just read `CONTENT_LENGTH` characters from `stdin` and write them to `stdout`.

`cgiparse -init` If `QUERY_STRING` is not defined, read `stdin` and output a string that when `eval`'d by Bourne shell it will set `QUERY_STRING` to its correct value. This can be used when the same script is used with both `GET` and `POST` method. Typical use in the beginning of Bourne shell script:

```
eval `cgiparse -init`
```

After this command the `QUERY_STRING` environment variable will be set regardless of whether `GET` or `POST` method was used. Therefore `cgiparse` may be called multiple times in the same script (otherwise with `POST` it could only be called once because after that the `stdin` would be already read, and the next `cgiparse` would hang).

7.1.2 Modifier Options

`-sep` ***separator*** Specify the string used to separate multiple values. With

- `-value` default is newline
- `-form` default is ", "

`-prefix` ***prefix***

- Only with `-form`. Specify the prefix to use when making up environment variable names. Default is "*FORM_*".

`-count` With

- `-keywords` outputs the number of keywords
- `-form` outputs the number of unique fields (multiple values are counted as one)
- `-value` *fieldname* gives the number of values of field *fieldname* (no such field is zero, one field gives 1, one multiple 2, etc).

−*number*, e.g. `-2` With

- `-keywords` gives *n*'th keyword
- `-form` gives all the values of *n*'th field
- `-value` *fieldname* gives *n*'th of the multiple values of field *fieldname* (first value is number 1).

`-quiet` Suppress all error messages. (Non-zero exit status still indicates error.)

All options have one-character equivalents: `-k -f -v -r -i -s -p -c -q`

7.2 Exit Statuses

- `0` Success
- `1` Illegal command line
- `2` Environment variables not set correctly
- `3` Failed to get requested information (no such field, `QUERY_STRING` contains keywords when form field values requested, etc).

7.3 Examples

Note: In real life, of course, `QUERY_STRING` is already set by the server.
Here `$` is the Bourne shell prompt.

7.3.1 Keyword Search

```
$ QUERY_STRING="is+2%2B2+really+four%3F"
$ export QUERY_STRING
$ cgiparse -keywords
is
2+2
really
four?
$
```

7.3.2 Parsing All Form Fields

```
$ QUERY_STRING="name1=value1&name2=Second+value%3F+That%27s right%21"
$ export QUERY_STRING
$ cgiparse -form

FORM_name1='value1'; FORM_name2='Second value? That'\''s right!'

$ eval `cgiparse -form`
$ set
...
FORM_name1=value1
FORM_name2=Second value? That's right!
...
$
```

7.3.3 Extracting Only One Field Value

```
QUERY_STRING as in previous example.
$ cgiparse -value name1
value1
$ cgiparse -value name2
Second value? That's right!
$
```

8. cgiutils Manual

`cgiutils` program is provided to make it easier to produce easily a full HTTP1 response header by NPH [No-Parse-Headers]scripts. It can also be used to just calculate the `Expires:` header, given the time to live in a human-friendly way, like

```
1 year 3 months 2 weeks 4 days 12 hours 30 mins 15 secs
```

8.1 Command Line Options

`cgiutils -version`	print the version information.
`-nodate`	don't produce the `Date:` header.
`-noel`	don't print the empty line after headers [in case you want to output other MIME headers yourself after the initial header lines].
`-status` ***nnn***	give full HTTP1 response, instead of just a set of HTTP headers, with HTTP status code *nnn*.
`-reason` ***explanation***	specify the reason line for HTTP1 response [can only be used with the `-status` *nnn* options.
`-ct` ***type/subtype***	specify the MIME content-type.
`-ce` ***encoding***	specify the content-encoding [e.g. `x-compress`, `x-gzip`].
`-dl` ***language-code***	specify the content-languge code.
`-length` ***nnn***	specify the MIME content-length value.
`-expires` ***time-spec***	specify the time to live, like `"2 days 12 hours"`, and `cgiutils` will compute the `Expires:` field value [which is the actual expiry date and time in GMT and in format specified by HTTP spec].
`-expires now`	means immediate expiry. Often this is exactly what the scripts should output.
`-uri` ***URI***	specify the *URI* for the returned document.
`-extra` ***xxx: yyy***	specify an extra header which cannot otherwise be specified for `cgiutils`.

Make sure that you quote the option arguments that are more than one word:

```
cgiutils -expires "2 days 12 hours 30 mins"
```

8.2 Examples

```
        cgiutils -status 200 -reason "Virtual doc follows" -expires now
  ==>
        HTTP/1.0 200 Virtual doc follows
        MIME-Version: 1.0
        Server: CERN/2.17beta
        Date: Tuesday, 05-Apr-94 03:43:46 GMT
        Expires: Tuesday, 05-Apr-94 03:43:46 GMT
```

There is an empty line after the output to mark the end of the MIME header section; if you don't want this [you want to output some more headers yourself], specify the `-noel` (NO-Empty-Line) option.

Note also that `cgiutils` gives automatically the `Server:` header because it is available in the CGI environment. The `Date:` field is also automatically generated unless `-nodate` option is specified.

To get only the expires field don't specify the `-status` option. If you don't want the empty line after the header line use also the `-noel` option:

```
      cgiutils -noel -expires "2 days"
==>
      Expires: Thursday, 07-Apr-94 03:44:02 GMT
```

9. CERN Server Clickable Image Support

CERN Server versions 2.14 and newer have a `htimage` program in the distribution, which is an `/htbin` program handling clicks on sensitive images. For versions 2.15 and newer it is a CGI program (uses the Common Gateway Interface to communicate with `httpd`). See demo.

9.1 In This Section...

- `htimage` installation
- Writing documents that contain clickable images
- Image configuration file
- Output of `htimage`

9.2 Installing htimage Binary

After compiling `htimage` you should move the executable binary to the same directory as your other server scripts are, and remember to set up an exec rule. For example if your scripts are in `/usr/etc/cgi-bin`, you could have an `Exec` rule like this:

```
Exec   /htbin/*   /usr/etc/cgi-bin/*
```

Often `htimage` is one of the most often used scripts, and it would therefore be nice to refer to it with as short a name as possible, like `/img`, so you could have a `Map` rule just before the `Exec`:

```
Map    /img/*     /htbin/htimage/*
Exec   /htbin/*   /usr/etc/cgi-bin/*
```

9.3 Writing a Document With Clickable Images

To create a clickable image in your HTML document, you'll need to:

- specify `ISMAP` in your inlined image call, and
- make that image an anchor, with an `HREF` to the script handling the request `(htimage)` with image configuration file name appended to it.

Each clickable image has to be described to `htimage` via an image configuration file. These files are referred to by the extra path information in the URL causing the call to `htimage`:

```
<A HREF="/htbin/htimage/image/config/file">
<IMG SRC="Image.gif" ISMAP></A>
```

Image configuration file can be:

- either a virtual path, that is translated through rule system,
- or an absolute path in your filesystem.

`htimage` will look for both of these (afterall, it gets both `PATH_INFO` and `PATH_TRANSLATED` environment variables from `httpd` anyway).

You can even do some very smart mappings in the rule file to allow very short references to `htimage` and picture configuration files. Let's suppose all your image configuration files are in directory `/usr/etc/images`. Then you can use the following two rules in your server's configuration file (by default `/etc/httpd.conf`):

```
Map   /img/*    /htbin/htimage/usr/etc/images/*
Exec  /htbin/*  /usr/etc/cgi-bin/*
```

In this case you can refer to your image mapper very easily; if you have an image configuration file `Dragons.conf` in `/usr/etc/images` directory, all you need to say in the anchor is this:

```
<A HREF="/img/Dragons.conf">
<IMG SRC="Image.gif" ISMAP></A>
```

9.4 Image Configuration File

There are four keywords:

`default` ***URL*** — *URL* which is used if click is in none of the given shapes. This should always be set!

`circle` ***(x,y) r URL*** — Circle with center point *(x,y)* and radius *r*.

`rectangle` ***(x1,y1) (x2,y2) URL*** — Rectangle with (any) two opposite corners having coordinates *(x1,y1)* and *(x2,y2)*.

`polygon` ***(x1,y1) (x2,y2) ... (xn,yn) URL*** — Polygon having adjacent vertices *(xi,yi)*. If the path given is not closed (first and last coordinate pairs aren't the same) the first and last coordinate pairs will be connected by `htimage.` So first point is added also as the last one if necessary.

These can be abbreviated as `def, circ, rect, poly.`

Shapes are checked in the order they appear in config file, and the URL corresponding to the first match is returned. If none match, the `default` URL is returned.

*URL*s are

- either full URLs (with access method, machine name and path), in which case server sends a redirection to client,
- or a partial URL containing only pathname part of it (always starting with a slash), in which case server considers that as the original request, translates it through the rule system, access authorization and serves it normally (faster than sending redirection).

9.5 Output Produced by htimage

`htimage` prints a single `Location:` field to its `stdout`, or an error message with preceding `Content-Type: text/html` so in fact `htimage` behaves exactly as any other CGI/1.0 program (script), and is not in any way handled specially by the server. Therefore, you can rename `htimage` to whatever you prefer, like we called it `/img` in the above example.

Server understands this `Location:` field, and either directly sends that file to the client (non-full URL), or sends a redirection to client causing it to fetch the document, maybe even from another machine.

Note that URLs returned by `htimage` may well be other script requests - there is no reason for being limited to just regular documents.

10. Protected CERN Server Setup

Access can be restricted according to user name, internet address, or both. Access control can be tree-level, file level, or both.

10.1 In This Section...

- Password File
- Group File
- Protect Directive in Configuration File
- Protection Setup File
- Protecting a Tree of Documents
- Protecting Individual Files
- Using Two-Level Protection
- Embedding the Protection Setup in the Configuration File Itself
- Access Control List File

10.2 Password File

If user-wise access control is used there has to be a password file listing all the users and their encrypted passwords. Password file can be maintained by `htadm` program which is a part ot CERN `httpd` distribution.

Unix password files are understood by CERN daemon (but not vice versa). However, **Unix users are in no way connected to the WWW access authorization.**

10.3 Group File

Group file contains declarations of groups containing users and other groups, with possibly an IP address template. Group declarations as viewed from top-level look like this:

```
groupname: item, item, item
```

The list of items is called a group definition. Each `item` can be a username, an already-defined groupname, or a comma-separated list of user and group names in parentheses. Any of these can be followed by an at sign `@` followed by either a single IP address template, or a comma-separated list of IP address templates in parentheses. The following are valid group declarations:

```
authors: john, james
trusted: authors, jim
cern_people: @128.141.*.*
hackers: marca@141.142.*.*, sanders@153.39.*.*,
         (luotonen, timbl, hallam)@128.141.*.*,
         cailliau@(128.141.201.162, 128.141.248.119)
cern_hackers: hackers@128.141.*.*
```

If an item contains only IP address template part all users from those addresses are accepted (e.g. `cern_people` above). Note the last two declarations: `cern_hackers` group is made up of the `hackers` group by restricting it further according to IP address.

Group definition can be continued to next line after any comma in the definition. Forward references in group file are illegal (i.e. to use group name before it is defined).

Group definition syntax is valid not only in group file, but also in

- `GetMask` in protection setup file, and
- in last field in ACL entries.

10.4 Server Configuration File

Typically you protect a tree of documents by `protect` rule in rule file, and specify authorized persons and IP addresses in the protection setup file or access control list file:

```
Protect /very/secret/*  /WWW/httpd.setup
```

If there are Unix file system protections set up so that there is no world read-permission the daemon naturally has to run as the owner or the group member of those files.

However, if there are protected trees owned by different people this doesn't work. In that case *the daemon has to run as `root`, and the user and group ids have to be specified in the `protect` rule,* e.g.:

```
Protect /kevin/secret/*   /WWW/httpd.setup1  kevin.www
Protect /marcus/secret/*  /WWW/httpd.setup2  marcus.nogroup
```

10.5 Protection Setup File

Each `protect` rule has an associated protection setup file. It specifies valid authentication schemes, password and group files, and password server-id:

```
AuthType      Basic
ServerId      OurCollaboration
PasswordFile  /WWW/Admin/passwd
GroupFile     /WWW/Admin/group
```

Password server id needs not be a real machine name. It's only purpose is to inform the browser about which password file it is using (different protection setups on the same machine can use different password file and that would otherwise confuse pseudo-intelligent clients trying to figure out which password to send).

Same server-ids on different machines are considered different by clients (otherwise this would be a security hole).

10.5.1 Protecting Entire Tree As One Entity

If you want to control access only to entire trees of documents and don't care to restrict access differently to individual files, it suffices to give a `GetMask` in setup file (and you don't need any ACL files):

```
GetMask    group, user, group@address, ...
```

Group definition has the same syntax as in group file.

10.5.2 Protecting Individual Files Differently

When each individual file needs to be protected separately you should use an ACL (access control list) file in the same directory as the protected files. After that no file in that directory can be accessed unless there is a specific entry in ACL allowing it.

In this case you don't need the `GetMask` in setup file.

10.5.3 Restricting Access Even Further

There may be both `GetMask` *and* an ACL, in which case both conditions must be met. This is typically used so that `GetMask` defines a general group of people allowed to access the tree, and ACLs restrict access even further.

10.6 Protection Setup Embedded in the Configuration File

Often it is not necessary to have the protection information in a different file; as a new feature `cern_httpd` allows protection setup to be "embedded" inside the configuration file itself.

Instead of writing the setup in a different file and referring to it by the filename, you can use the `Protection` directive to define the protection setup and bind it to a name, and later refer to this setup via that name.

The previous example could be written into the main configuration as follows:

```
Protection  PROT-NAME  {
    UserId        marcus
    GroupId       nogroup
    AuthType      Basic
    ServerId      OurCollaboration
    PasswordFile  /WWW/Admin/passwd
    GroupFile     /WWW/Admin/group
    GetMask       group, user, group@address, ...
}
Protect  /private/URL/*       PROT-NAME
Protect  /another/private/*   PROT-NAME
```

Note that since the protection setup is in the same file as the other configuration directives, it is also possible to specify the `UserId` and `GroupId` for the server to run as, without it being a security hole. With external protection setup this is made impossible because of security reasons; that is why there is an extra field after the protection setup filename specifying the user and group ids in that case:

```
    Protect /kevin/secret/*    /WWW/httpd.setup1   kevin.www
    Protect /marcus/secret/*   /WWW/httpd.setup2   marcus.nogroup
```

If you need a given protection setup only once there is no need to first bind it to a name and then refer to it by that name, but rather just combine the two:

```
Protect  /private/URL/*  {
    UserId        marcus
    GroupId       nogroup
    AuthType      Basic
    ServerId      OurCollaboration
    PasswordFile  /WWW/Admin/passwd
    GroupFile     /WWW/Admin/group
    GetMask       group, user, group@address, ...
}
```

`httpd` is not very robust in parsing this particular directive; make sure you have a space between the URL template and the curly brace, and that the ending curly brace is alone on that line. Also, comments are **not** allowed inside the protection setup definition.

10.7 Access Control List File

ACL file is a file named `.www_acl` in the same directory as the files the access of which it is controlling. It looks typically something like this:

```
secret*.html : GET,POST : trusted_people
minutes*.html: GET,POST : secretaries
*.html : GET : willy,kenny
```

It is worth noticing that all the templates are matched agaist (unlike in rule file where translation of rules stops in `pass` and `fail.`. So in the previous example all the HTML files are accessible to `willy` and `kenny,` even those matching the two previous templates.

The last field is just a list of users and group (possibly at required IP addresses), and in fact this field is in same syntax as group file.

When `PUT` method will be implemented it can appear in the middle field separated by a comma from `get`:

```
*.html : GET,PUT : authors
```

10.8 Manual Page For htadm

CERN `httpd` password file can be maintained with `htadm` program which is a part ot CERN `httpd` distribution.

10.8.1 Command Line Options and Parameters

`htadm -adduser` *passwordfile* **[***username* **[***password* **[***realname* **]]]** adds a user into the password file (fails if there is already a user by that name).

`htadm -deluser` *passwordfile* **[***username* **]** deletes a user from the password file (fails if there is no user by that name).

`htadm -passwd` *passwordfile* **[***username* **[***password* **]]** changes user's password (fails if there is no such user).

`htadm -check` *passwordfile* **[***username* **[***password* **]]** checks user's password (fails if there is no such user). Writes either `Correct` or `Incorrect` to standard output. Also indicates password correctness by a zero return value.

`htadm -create` *passwordfile* creates an empty password file.

If *`password`* or even *`username`* is missing in either of the previous cases they are prompted interactively. *`passwordfile`* must be always specified. Missing real name is also prompted when adding a new user.

Do NOT use `htadm` to add new users to the actual Unix password file `/etc/passwd,` entries written by `htadm` are missing some necessary fields to Unix.

Passwords should not be longer than 8 characters (this is a restriction from linemode clients using C library function `getpass()` to read the password –– there is no other cause for this restriction; the maximum hardcoded password size is actually much larger, and if you only use GUI or other clients that are able to read this long passwords, feel free to use them).

`htadm` destroys the password from command line as soon as possible so that it is very unlikely to see somebody's password by looking at the process listing on the machine (with `ps`, for example).

11. Proxies

Proxy is a HTTP server typically running on a firewall machine, providing with access to the outside world for people inside the firewall. `cern_httpd` can be configured to run as a proxy. Furthermore, it is able to perform caching of documents, resulting in faster response times.

I (Ari Luotonen, CERN) and Kevin Altis from Intel have written a joint paper about proxies which will be presented in the WWW94 Conference.

11.1 In This Section...

- Server setup
- Proxy protection
- Configuring proxy to use another proxy
- Caching
- Client setup

11.2 Setting Up cern_httpd To Run as a Proxy

`cern_httpd` runs as a proxy if its configuration file allows URLs starting with corresponding access method to be passed. Typical proxy configuration file reads:

```
pass http:*
pass ftp:*
pass gopher:*
pass wais:*
```

Note that `cern_httpd` is capable of running as a regular HTTP server at the same time; just add your normal rules after those ones.

The `proxy_xxx` environment variables that are used to redirect clients to use a proxy also affect the proxy server itself. If this is not your intention make sure that those variables are not set in `httpd`'s environment.

11.3 Proxy Protection

`cern_httpd` 2.17 and newer provide a mechanism to protect the proxy against unauthorized use (in fact, the machinery behind this is the same that is used to set up document protection when running as a regular HTTP server).

11.3.1 Enabling and Disabling HTTP Methods

By default only `HEAD`, `GET` and `POST` methods are allowed to go through the proxy. You can enable more methods using the `Enable` directive in the configuration file:

```
Enable PUT
Enable DELETE
```

The `Disable` directive disables methods:

```
Disable POST
```

11.3.2 Defining Allowed Hosts

A certain protection setup is defined to the proxy as a single entity that is given a name. Later, when protecting certain URLs this name is used to refer to the protection setup. (The name can also be the absolute pathname of the file that defines the protection, if one wishes to store protection information in a different file.)

Protection is defined as follows:

```
Protection  protname  {
    Mask @(*.cern.ch, *.desy.de)
}
```

This defines a protection that allows all request methods from domains `cern.ch` and `desy.de`, and none from elsewhere. This protection can be referred to by *protname*.

You can also use IP number templates:

```
Protection  protname  {
    Mask  @(128.141.*.*, 131.169.*.*)
}
```

Note that IP number templates always have four parts separated by dots.

If allowed methods are different according to domain, e.g. `GET` should be allowed from both of these domains, but `POST` and `PUT` only from `cern.ch`, you can use `GetMask`, `PostMask`, `PutMask` and `DeleteMask` directives instead:

```
Protection  protname  {
    GetMask  @(*.cern.ch, *.desy.de)
    PostMask @*.cern.ch
    PutMask  @*.cern.ch
}
```

Note that parentheses are necessary only if there is more than one domain name template.

11.3.3 Actual Protection

The `Protect` rule actually associates protection with a URL. In case of proxy protection you would typically say:

```
Protect  http:*   protname
Protect  ftp:*    protname
Protect  gopher:* protname
Protect  news:*   protname
Protect  wais:*   protname
```

which would restrict all proxy use to the allowed hosts defined previously in the protection setup *protname*. **Note** that *protname* must be defined before it is referenced!

11.4 Caching

`cern_httpd` running as a proxy can also perform caching of files retrieved from remote hosts. See the configuration diretives controlling this feature.

12. CERN Server FAQ

If you have problems, first make sure you're using the newest version. You'll find that out by peeking into ftp://info.cern.ch/pub/www/src.

When something goes wrong you should run server in verbose mode (the `-v` flag) to see exactly what is the problem. If you usually run it from inet daemon start it now standalone to some other port (with `-p` *port* flag) with otherwise the same parameters as in `/etc/inetd.conf`.

12.1 My Scripts Get Served As Text Files...

...or are completely unaccessible.

It's important to understand that rules in the configuration file (`Map`, `Pass`, `Exec`, `Fail`, `Protect`, `DefProt` and `Redirect`) are translated from top to bottom, and the first matching `Pass`, `Exec` or `Fail` will **terminate** rule translation.

So, make sure that your `Exec` rule is before any general `Map`pings.

12.2 How do I...

- Set up access authorization?
- Write server-side scripts?
- Get the server to perform searches?
- Make clickable images?
- Handle forms?
- Set up a proxy
- Set up proxy caching

12.3 Zombies

There used to be one zombie when running `cern_httpd` standalone; this was fixed in version 2.17beta. If you still see zombies (more than two that don't go away in a few minutes) it is a bug.

12.4 Inet daemon complains about looping...

...and terminates WWW service. `:-(`

This is a hard-coded `inetd` limitation on at least SunOS-4.1.* and NeXT, which limits maximum allowed connections from a given host to 40 per minute. This can be exceeded by scripts doing Web-roaming, or documents having masses of small inlined images.

There is a fix for at least SunOS `inetd` (100178-08), and in Solaris this is fixed. You can also run `httpd` standalone (preferably with the `-fork` command line option).

Most importantly, you should stop running `httpd` from `inetd` and rather run it standalone. This is because running from `inetd` is inefficient.

12.5 Server looks at funny directories and finds nothing

From version 2.0 until 2.15, you need to have an explicit map to file system in your rule file, e.g.:

```
Map    /*    file:/*
```

but 2.15 doesn't have this limitation anymore.

12.6 But the document says rule file is no longer needed

True, but it also says you must remember to give your Web directory as a parameter to `httpd,` e.g.

```
httpd  /home/me/MyGloriousWeb
```

13. CERN httpd 2.15 Release Notes

There is one single thing that needs to be done when changing over from `httpd` 2.14 to 2.15:

```
Rename your old /htbin scripts to end in .pp suffix!
```

13.1 General Notes

- Code tested under Purify –– all detected memory leaks and bugs fixed.
- Forking code enhanced –– no longer crashes when running standalone. Everybody should start running CERN httpd standalone instead of from inetd
- Documentation redesigned, but still under construction
- Contains Solaris port, but not VMS

13.2 CGI/1.0, Common Gateway Interface

- CGI/1.0 interface fully implemented
- **Old CERN httpd scripts will continue working if you rename them to end with .pp suffix.** Links referencing these scrips do NOT need to be changed. (This feature does not add any overhead to CGI/1.0 script calls.)
- New product cgiparse for CGI/1.0 scripts to parse QUERY_STRING env.var and to read CONTENT_LENGTH characters from stdin
- `htimage` upgraded to CGI/1.0
- The whole server-environment is propagated to CGI script, except for variables that are reserved for CGI/1.0.
- Scripts are spawned by doing a fork() and exec() instead of system() –– more efficient and secure

13.3 Firewall Gateway Modifications

- Access authorization works thru firewalls
- So does POST, therefore forms also
- -disable/-enable command line options and Disable/Enable configuration directives for dis/enabling HTTP methods. GET, HEAD and POST are enabled by default.
- Fix: text/html and text/plain not passed multiply to servers when running as gateway
- Fix: */*, image/* etc not expanded by the gateway
- Fix: try local search ONLY when accessing local files

13.4 Other New Features

- When started standalone in non-verbose mode automatically disconnects from terminal session and goes background
- User-supported directories enabling URLs starting with **/ũsername**
- Redirection
- Meta-information files to allow RFC-822-style headers to be appended to server response header section
- New, common logfile format, localtime default, `GMT` as an option
- Ability to suppress logging for certain hosts/domains according to given hostname template or IP number mask, like `*.cern.ch` or `128.141.*.*`
- -setuid option to set server uid to authenticated uid (local)
- Multilanguage support: same URL can be used to retrieve a document in different languages
- AddLanguage, AddEncoding and AddType directives to configuration file (AddType replaces Suffix)
- Better multiformat algorithm
- HostName directive to configuration file for servers that want to give CGI/1.0 scripts a different hostname than the actual. Useful if machine has many aliases, or if httpd fails to get the full domainname.
- Exec rule obsoliting HTBin directive −− now multiple script directories possible, with arbitrary mappings
- Get-Mask, Post-Mask and Put-Mask for protection setup files. Get-Mask obsolites Mask-Group
- Groups All/Users and Anybody/Anyone/Anonymous automatically defined. All means anybody that has been authenticated, and Anybody is just anybody
- Server:
- Last-Modified:
- Content-Length:
- Content-Language:
- Content-Encoding:
- Scripts can output also Uri: and Expires: headers (this will eventually be made more general)
- HEAD works, also with stupid scripts that also output the body

13.5 Enhancements, Fixes

- The final explicit Map to filesystem in configuration file no longer required, because it was causing confusion
- Assume Basic authentication scheme even if not explicitly mentioned in setup file
- Get client DNS hostname, for the logfile among other things
- Fail made the default when rules are translated to the end without coming accross with a Pass, Exec or Fail rule (this is to enhance security, it was too easy to forget the Fail * from the end of config file)
- Made config (rule) file understand different ways of writing keywords, e.g.: UserDir, userdir, User-Dir, user_dir, UserDirectory and so on
- The eight misplaced server-side access authorization files moved away from libwww
- Fix: directory indexing works with a trailing slash
- Fix: HTSimplify() might have behaved unexpectably on some systems (called strcpy() with overlapping args)

14. CERN httpd 2.16beta Release Notes

- If you are upgrading from 2.15beta, you need to make **no changes**.
- If you are upgrading from 2.14, there is one single thing that needs to be done:

```
Rename your old /htbin scripts to end in .pp suffix!
```

14.1 Firewall Gateway (Proxy) Additions, Fixes

- `ftp` with binary files work
- `x-compress` and `x-gzip` work correctly over proxy
- Firewalling now works through arbitrary number of proxies; `http_proxy, ftp_proxy, gopher_proxy` and `wais_proxy` configuration directives cause proxy to connect to the outside world through another proxy. Environment variables with the same names have same effects, but config file is user-friendlier for this.
- Now sends all the headers sent by client.
- Proxy log file now gives byte count.
- Proxy log file now gives correct status code also on error.

14.2 Firewall Gateway (Proxy) Caching

- `CacheRoot` directive specifies cache root directory, and turns on proxy caching. Cache root directory must be dedicated to `httpd` - all files in there are subject to garbage collection.
- Cache size (in megabytes) is specified by `CacheSize` directive; cache size should be several megabytes, 50-100MB should give good results. Cache may, however, temporarily grow a few megabytes bigger than specified. Also, space taken up by directories is not calculated in the current version.
- `http, ftp, gopher` with `GET` method get cached.
- However, not caching:
  - HTTP0 responses (you never know if it failed; also confused HTTP1 servers sometimes output garbage in front of HTTP1 headers).
  - Protected documents (request had `Authorization:` field).
  - Queries - they have too often side-effects. (POST should be **always** used with forms, and all script responses should have `Expires:` header when necessary. Until then, we don't cache them.)
- Expiry date is extracted:
  - From `Expires:` header.
  - If not present `Last-Modified:` is used to approximate expires. If a file hasn't changed in five months the chances are it won't change during the next week. On the other hand, if a file has changed yesterday, it will probably change again pretty soon. I know this is heuristic but until all the servers give `Expires:` this works much better than not using it, so no flames about it.
  - If `Last-Modified:` not given use the time given by `CacheDefaultExpiry` directive, default 7 days.

- Format of cache files and directory structure under cache root is subject to change if necessary. No application should yet rely on any certain cache format. Eventually I can see clients accessing cache files directly, bypassing proxy server.
- Caching system understands both time formats, also the one output by old NCSA httpds.
- Cache files get locked during transfer. Lock files time out if something goes wrong. Timeout can be set by `CacheLockTimeOut` directive (default 20 minutes). During the lock is in effect, further requests to the same file get retrieved from the remote host.
- Garbage collection directives:
  -
  - `GcMemoryUsage` to advice gc about how radical to be in memory use (more memory => smarter gc).
  - `GcTimeInterval`, how often to do gc.
  - `GcReqInterval`, after how many requests to do gc.
  - (gc is also automatically started if cache size limit is reached.)
  - `CacheLimit_1`, size in KB until which files are equally valuable despite their size (200K).
  - `CacheLimit_2`, size in KB after which files get discarded because they are too big (4MB).
  - `CacheClean`, remove all files older than this (default 21 days).
  - `CacheUnused`, remove all files that have not been used in this long time (default 14 days).
- Garbage collector always removes all expired, too long unused, and too old files.
- If cache size limit is reached some files need to be sacrified; the current algorithm takes into account:
  - Time remaining to unconditional removal; if it expires tomorrow it might as well be removed today.
  - Time last accessed; if it hasn't been accessed in 5 days, it probably won't be accessed anymore before it expires.
  - Size; huge files get removed move easily.
  - Time it took to load it from the remote host; files that were time-consuming to transfer have much higher value. This compensates the size factor. Load delay is the single most significant value.
  - Time it has already been in cache; ancient files get removed more easily than fresh ones.

14.3 Other New Features

- Error log file.
- `Referer:` field ends up in error log when a request fails.
- `UserId` and `GroupId` to set default uid and gid (used instead of nobody and nogroup).
- Timeout for input and output; default time to wait for a request is 2 minutes, and to send response 20 minutes. Timeout causes a note to error log, and terminates child (no more hanging httpds). **Note:** the one zombie is normal; don't report to me about it, I may do something about it some day, or maybe I won't. Zombie doesn't take up any other system resources except the one process table entry.
- Suffixes are no longer case-sensitive by default; this may be changed via the `SuffixCaseSense` configuration directive.

- Lou Montulli's news and proxy diffs added to the library.
- Most command line options now also available as configuration directives:
  - `DirAccess`
  - `DirReadme`
  - `AccessLog`
  - `ErrorLog`
  - `LogFormat`
  - `LogTime`
- `-vv` command line option for Very Verbose trace output. Outputs also request headers as they came in. Otherwise like `-v` flag.

14.4 Enhancements, Fixes

- NPH-scripts now work from automatically backgrounded standalone server.
- Fixed the many problems with `Content-Transfer-Encoding`:
  - Mosaic uses `Content-Encoding`, although spec says `Content-Transfer-Encoding`; I now output both
  - `Content-Transfer-Encoding` sometimes didn't show up although it should have, fixed.
  - `Content-Transfer-Encoding` didn't come up correctly with ftp, fixed.
- Strange escaping fixed with directory indexing (legal characters got escaped randomly by a gcc-compiled version).
- Timezone bug around midnight with the new logfile format fixed. (New logfile format is not yet default, use `-newlog` command line option, or `LogFormat` directive in configuration file.)
- Dashes for non-existent status codes and byte counts now show up correctly in the log.
- Forking code once again enhanced - fixed a possible hanging situation.
- Log time fixed to be the time of incoming request, not the time of request served.
- Zombies now correctly waited away on HP (this was in fact fixed already in 2.15beta binaries distributed after February 17th - **note,** that this bug had no effect on any other platforms).
- Directory listings no longer have `Content-Length:` (because it was wrong).
- Now understands also the old Accept: syntax, with spaces as separators between actual content-type and its parameters. This will eventually be taken out.
- `htadm` now uses the same file creation mask as in the original password file.

15. CERN httpd 2.17beta Release Notes

15.1 General New Features

- `PUT` and `POST` can be configured to be handled by external CGI scripts; `PUT-Script` and `POST-Script` directives
- BodyTimeOut for timing out scripts waiting for input that never comes from clients
- `IdentityCheck` directive to turn on RFC931 remote login name checking
- `REMOTE_IDENT` for CGI giving remote login name; this was the only feature missing to be fully CGI/1.0 compiant
- CGI/1.1 upgrade:
  - all the headers without a special meaning to CGI from CGI scripts get passed to the client
  - Status: header to specify the HTTP status code and message for client when not using NPH scripts
  - all HTTP request header lines which are not otherwise available to the scripts get passed as HTTP_XXX_YYY environment variables
- Understands conditional `GET` request with `If-Modified-Since` header
- `kill -HUP` causes `httpd` to re-read its configuration file
- `PidFile` directive for specifying the file to write the process id [makes it easy to send the `HUP` signal
- `ServerRoot` directive to specify a "home directory" for `httpd`
- Directory listings with icons; by default icons are in `icons` subdirectory under `ServerRoot`
- The precompiled binaries are distributed in a `tar` packet that contains a set of default icons; the easiest way to configure the icons is to just set the `ServerRoot` to point to the binary distribution directory [its name is `cern_httpd`]
- Welcome directive to specify the name of the overview page of the directory; default values are `Welcome.html`, `welcome.html` and, for compatibility with NCSA server, `index.html`. Use of `Welcome` directive will override all the defaults.
- `AlwaysWelcome` directive to configure if `/directory` and `/directory/` are to be taken to mean the same thing, or should only `/directory/` be mapped to the overview page and `/directory` produce the directory listing.
- /ũser causes an automatic redirection to /ũser/
- Now gives also the `Date:` header.
- `Port` directive to config file specifying the port number to listen to.

15.2 Access Authorization Enhancements / Proxy Protections

- Now also domain name templates, like *.cern.ch, can be used in specifying allowed hosts, not only IP number masks
- `ACLOverRide` directive to allow ACLs to override the `Masks` set in the protection setup [without this feature ACLs cannot allow anything more than what the `Masks` allow, only restrict access further]. This directive disables `Mask` checking if an ACL file is present.
- Since setting up protection seemed to be unnecessarily hard, it is now possible to give the protection setup in the main configuration file instead of having to use a different file; it is still ok to use a different file.
  - `Protection` directive defines a protection setup and associates a name with it:

    ```
    Protection  prot-name  {
    AuthType    Basic
    ServerId    Test-Server
    PasswdFile  /where/ever/passwd
    GroupFile   /where/ever/group
    UserId      someuser
    GroupId     somegroup
    GET-Mask    list, of, users, and, groups
    POST-Mask   list, of, users, and, groups
    PUT-Mask    list, of, users, and, groups
    }
    ```

    The content between the curly braces is the same as used to go the the protection setup file. What's new is the possibility to specify the `UserId` and `GroupId` for the clild process when serving the request in protected mode. This is not possible with external files for security reasons [it is not possible inside the external file, but it is not possible if the ids are set when calling that file; see doc for more details].
  - A single `Mask` directive for cases when `GET-Mask`, `POST-Mask` and `PUT-Mask` are the same.
  - In `Protect` rule the *prot-name* is specified instead of the file name; what's more is that `Protect` can now be used to protect also proxied URLs:

    ```
    Protect http:*   prot-name
    Protect ftp:*    prot-name
    Protect gopher:* prot-name
    ```

15.3 Enhancements, Fixes

- Incorporated Ian Dunkin's <imd1707@ggr.co.uk> SOCKS modifications (thank you, Ian!); read the `README-SOCKS` file in the source code distribution for more information.
- `SIGPIPE` causes a normal child to exit; proxy child will correctly stop writing to client socket but still writes to cache file [previously just kept on writing to the socket, too]
- 401, 402, 403, 404 errors don't go to error log anymore
- error log contains now the host name and request
- no longer sends `Content-Transfer-Encoding`, we agreed upon using `Content-Encoding` for compression

- fixed funny panic message from format module in verbose mode even though everything was ok [only aesthetic]
- now gives again "not authorized" rather than not found if trying to access a protected but nonexistant file; this way even filenames don't leak
- all time specifications in configuration file have more readable forms:

```
1 year
2 months
3 weeks 2 days
5 days 20 hours 30 mins 2 secs
20:30
20:30:01
2 weeks 20:30
```

- Case-sense bug with `LogTime`, `LogFormat`, `DirAccess` and `DirReadme` fixed; now paramters really are handled in a case-insensitive manner.

15.4 Proxy Additions, Fixes

- Proxy protections, see above
- Made proxy do smart guesses about the content of an unknown file while retrieving from the remote; this will end the problems of some files not being transferred to WinMosaic or Lynx. **IMPORTANT: Everybody, remove the rule [if you have it]**:

```
AddType  *.*  text/plain
```

  because it would disable this smart feature.
- Fixed a bug with unknown binary gopher files being truncated
- Fixed the bug with trailing slashes in ftp directory listings
- Fixed the bug with requests not being URL-encoded when forwarding the request
- Fixed a bug with filenames in directory listings not being URL-encoded
- Fixed stupid "mail-us" situation in certain situations when ftp load fails

15.5 Proxy Caching

- Cache is refreshed using the conditional `GET` method [use of `If-Modified-Since` header]
- Standalone cache mode with `CacheNoConnect` directive [causes an error rather than document fetch when the document is not in the cache]
- Possibility to disable garbage collection altogether
- Possibility to disable expiry checking
- Caching Off to explicitly turn off caching even if there are other caching directives specified

- `-gc_only` command line option to do garbage collection as a `cron` job for sites that run `httpd` as a proxy from `inetd`. However, since `httpd` now re-reads its configuration files when it receives a `HUP` signal, it makes standalone operation now even more easy, and `inetd` should no longer be much more convenient.
- Host names are converted to all-lower-case to avoid doing multiple caching for a single site.
- Files expiring immediately never get written to the cache; not even part of it.
- By default HTTP-retrieved documents without an `Expires:` and `Last-Modified:` field never get cached [because they are usually generated by scripts and should never be cached]; therefore I strongly advice against the use of `CacheDefaultExpiry` for HTTP.
- Caching control directives have changed to take a URL template as a first argument, and a more readable time format:

```
CacheDefaultExpiry  ftp:*     2 weeks 4 days
CacheDefaultExpiry  gopher:*  6 days
CacheUnused         http:*    1 month
CacheUnused         ftp:*     2 weeks
CacheUnused         gopher:*  1 week 5 days 2 hours 1 min 30 secs
```

- Made the expiry date approximation configurable; by default documents with `Last-Modified:` but without `Expires:` expire after 10% of the time that they have been unmodified. `CacheLastModifiedfactor` can be used to change this value, or turn this feature `Off`. Default value is 0.1 [=10%].
- Understands yet another date format:

```
Thu, 10 Feb 1994 22:23:32 GMT
```

  This date format is **not** conforming to the spec, so use of it is discouraged! This is only to make the proxy more robust.
- `NoCaching` directive to prevent certain URLs from being cached at all.
- Time margin to get rid of problems with machine clocks having inaccurate times and confusing caching.
- `GcDailyGc` to specify a daily garbage collection time, by default 3:00. [Can be turned `Off`, too.]
- Now possible to disable `GcReqInterval` and `GcTimeInterval` [by default disabled].
- Expired cache lock files get removed also during gc.
- `CacheAccessLog` to specify a different log file for cache accesses; also possible to make a separate log for each remote host.

15.6 cgiutils

A new product `cgiutils` for producing HTTP1 replies from CGI scripts, and for easily generating the `Expires:` header given the time to live, e.g. "2 weeks 4 hours 30 mins".

16. CERN httpd 2.18beta Release Notes

16.1 New Features

- Long FTP directory listing with last modification dates and sizes

16.2 Fixes

- Fixed a bad bug with `Port` directive -- server didn't fork but rather the parent process served which caused the service to eventually hang (this is the main reason for this release).
- `CLIENT_CONTROL` removed from SOCKS mods since `httpd` has now native proxy protection support.
- No longer fails to sometimes create `.gc_info` file.

17. CERN httpd 3.0 PreRelease Notes

17.1 3.0 Prerelease 3

- No longer strips hyphens from content-types and content-encodings that are given in the configuration file (broken in pre1).
- GMT-to-localtime transformation works now on all platforms in caching (was broken on others than Sun).
- Binary-FTP works again (broken pre2).
- Unescaping bug fixed in news module (caused many articles to fail to be retrieved).
- News module now gives appropriate error reponses for unavailable articles and non-existent news groups.
- FTP and HTTP modules now give better error responses.
- Fixed the cache access log to show the correct content-lengths.

17.2 3.0 Prerelease 2

- Respects UserId and GroupId directives again.
- FTP module no longer prints messages to stderr in non-verbose mode.
- ũsername form understood with ServerRoot, Search, PutScript, PostScript, DeleteScript, AccessLog, ErrorLog, CacheAccessLog directives.
- Opens cache access log only if caching is turned on.
- Binary distribution now contains a template configuration file that has all the configuration directives understood by httpd (thanks to Sean Gonzalez for it!).

17.3 3.0 Prerelease 1

- If-Modified-Since GET request now works correctly with proxy (client can do conditional GET/proxy can do conditional GET plus all the combinations of these).
- `Pragma: no-cache` supported; by sending this header to the proxy the client will force it to refresh its cache from remote server. Pragma headers are also forwarded to the remote server.
- Server now resets its state correctly when it receives the HUP signal (directory listing icons used to stop working).
- `-restart` option - `httpd` will find out the actual server process number and send s HUP signal to it to make it reload its configuration files; note that `httpd` must still have the same configuration file command line parameters (`-r` options) as the actual server (so it finds out the ServerRoot and PidFile).
- Now makes appropriate entry to error log when restarting.
- Made common logfile format default, the old format can still be used with the `LogFormat` directive:

  ```
  LogFormat old
  ```

- Multiple wild-card (asterisk) matching in configuration file works; it is a bit different from typical regular expression matching in that the wildcard matches the *shortest* possible amount of characters instead of the longest matching string; this is the best choise in most of the cases. Consider:

  ```
  Pass  http://*/*  /mirror/*/http/*
  ```

  Clearly the first asterisk should rather match only the hostname, and **not** the entire path except the filename.

- Rules can now have asterisks and whitespace in them: precede them with a backslah; as a result also the backslash itself has to be escaped with another backslash.

- The tilde character after a slash has to be explicitly matched:

  ```
  Map /* /foo/bar/*
  ```

  does *not* match user-supported directories, but:

  ```
  Map /~* /Webs/users/*
  ```

  does match them.

- Fixed the problem that user-supported directories could not be mapped or `Protect`'ed.

- Hostname matching made case-insensitive in access control/caching

- Added suffixes `.htm` and `.htmls` to the default set of known suffixes.

- Fixed some of the mysterious caching problems (all that were reported to me and that I could reproduce).

- Made it possible to specify the various byte/kilo/mega sizes in cache configuration with letters after the number (so it's no longer necessary to remember if the default is kilobytes or megabytes):

  ```
  CacheSize 150 M
  CacheLimit_1 100 K
  CacheLimit_2 2 M
  ```

  The numbers still have to be cardinals.

- Content-Length given for *all* documents, including (non-nph-)script responses, generated directory listings, error responses, all the documents retrieved over another protocol by the proxy (FTP, Gopher, ...), including HTTP responses from servers that didn't give it originally.

- `MaxContentLengthBuffer` directive to specify the maximum bytecount for the proxy to buffer in order to find out the content-length for the client - content-length is *always* calculated for the logs, but the user migth interrupt the connection if nothing seems to be happening, even though it is the proxy that is just buffering the entire file in order to find out the content-length before actually sending it to the client.

- Caching module now checks that it receives the correct content-length; if not it discards the cached document. This rules out the possibility to cache a truncated document from a timed out connection in 99.99% of the cases (0.01% comes from the fact that Plexus sends a timeout error message concatenated to the document and if so should happen that this produces exactly the correct content-length then there is nothing that can be done about it; in practice this never happens).

- Made `HEAD` work always, even on proxy with other protocols (FTP, Gopher...).

- PASV (Passive mode) in FTP now supported. It is no longer necessary to allow incoming connections above 1024 on the firewall host just to make FTP work. If PASV fails `httpd` will retry PORT.
- Welcome messages from FTP servers get shown on top of the directory listings.
- Fixed bug with old FTP files fixed getting wrong date in the listing.
- Gopher listings now have icons.
- Proxy now reports unknown host errors appropriately.
- Fixed encoding-decoding problems with directory listings.
- Added `ScriptTimeOut` - scripts that do not finish in this amount of time will be killed by `httpd`. Default value is 5 minutes.
- A /ũsername URL with an invalid username no longer causes an infinite redirection loop.
- The two files missing in FTP listings are no longer missing (they weren't in 2.18beta, either).
- Fixed a possible error condition that might cause the server to stop responding, or even die.
- Server now resets its UserId and GroupId even when in gc-only mode (this solves problems with `.cache_info` files sometimes being unwritable to actual caching processes).
- CacheAccessLog is now opened during startup while running as root to avoid opening problems. There is no longer logging to individual files according to remote hosts - all cache accesses are logged to this single file.
- `CacheOnly` directive for specifying a set of URLs that should be cached (for cases when there are only a few sites that should be cached).
- Added `DELETE-Script` directive for specifying the CGI script to handle `DELETE` method.
- `NoProxy` directive to allow the proxy to do direct access to some servers instead of connecting to another proxy server (contains a list of domain names). This works exactly like the `no_proxy` environment variable on clients. (Thanks to Rainer Klute for the patch!) This is only necessary when running multiple proxy servers that connect to each other.
- Fixed a bug that sometimes caused time directives to be parsed incorrectly (e.g. `CacheDefaultExpiry`).
- Multilanguage addition to allow server to understand e.g. that British English is also English, and that the US citizens do understand it (thanks to Toshihiro Takada for the patch!).
- Removed:
  - `GcReqInterval` and `GcTimeInterval` - not very good criteria to start doing garbage collection (`GcDailyGc` is better, giving the actual time to lauch gc)
  - cache access logging to individual logfiles according to remote host (wasted resources - a separate program is better for collecting this information from a single log file).
  - `-a` and `-R` options (never used).
  - `BodyTimeOut` replaced by `ScriptTimeOut`
  - `includes` from Makefiles (not supported by all the `makes`).
  - `#elif` preprocessor directive removed (wasn't supported by all the HP preprocessors)