European
Microsoft
Windows NT
Academic
Centre

HTTP Server Manual
Version 0.99

Copyright Notice

European
Microsoft
Windows NT
Academic
Centre

HTTP Server Manual
Version 0.99

Chris Adie
10 October, 1995

European Microsoft Windows NT Academic Centre
Computing Services
James Clark Maxwell Building
Kings Buildings
Mayfield Road
Edinburgh
EH9 3JZ
United Kingdom

TABLE OF CONTENTS

1. INTRODUCTION

This manual describes the freeware EMWAC World-Wide Web HTTP Server for computers running the Windows NT operating system. You should read it if you plan to install, operate, manage or deinstall the HTTP Server software. This manual assumes you have a reasonable degree of competence in the use of Windows NT, and a reasonable knowledge of HTTP and the World-Wide Web.

Note that this manual is also available as hypertext, on-line at:

`http://emwac.ed.ac.uk/html/internet_toolchest/https/contents.htm`

The EMWAC HTTP Server for Windows NT implements the HTTP/1.0 protocol. It runs as a Windows NT "service", just like the FTP Server which comes with Windows NT. By analogy with the UNIX HTTP server daemon which is called `httpd`, the Windows NT HTTP server service is called `https`. The HTTP server service is configured using a Control Panel "applet".

This HTTP server originates from the European Microsoft Windows NT Academic Centre (EMWAC). Located at Edinburgh University Computing Service, EMWAC has been set up to support and act as a focus for Windows NT within academia. It is sponsored by Datalink Computers, Digital, Microsoft, Research Machines, Sequent and the University of Edinburgh.

1.1. Support

EMWAC does not provide any support for this product. However, there are various mailing lists which you may find helpful advice from. One such list is called WebServer-NT.

The WebServer-NT mailing list is intended as a forum where users of Windows NT can discuss World Wide Web server issues. Topics include (but are not limited to):

- ideas for novel ways of using WWW servers on Windows NT
- security issues on NT Web Servers (encryption, proxy servers, ACLs)
- integration of Web Servers with NT applications
- CGI scripts wanted/available for Windows NT
- features which people would like to see in future Web Server
- products for Windows NT
- the Freeware Web Server (HTTPS) for Windows NT
- configuration and document management issues for NT-based Web Servers

To subscribe to the list, send a message to:

`webserver-nt-request@mailserve.process.com`

and in the message body type:

```
SUBSCRIBE webserver-nt
```

To unsubscribe, send message to the same address, but in the message body type:

```
UNSUBSCRIBE webserver-nt
```

- To get help on the mailserver commands put `HELP` in your message body
- To receive a list of the available mailing list put `LISTS` in the body

The mailing list is not moderated. Thanks to Process Software for setting up the list.

1.2. Professional Version

Edinburgh University Computing Service has produced a "Professional" version of HTTPS, which is being marketed through third parties. The Professional version is fully supported and has enhanced functionality. The following URL contains further information about the Professional version:

`http://emwac.ed.ac.uk/html/internet_toolchest/https/prof.htm`

2. INSTALLATION

2.1. Requirements

To use the Windows NT HTTP Server, you need to have a computer with the following characteristics:

- Intel, MIPS, Power PC or Digital Alpha processor.
- Windows NT 3.1 final release, with TCP/IP software installed.
- At least 16Mb of memory.
- Network connection **-** typically Ethernet.

2.2. Installing

1. Log into your Windows NT system as a user with administrative privileges.
2. The HTTP Server is distributed in four versions, for the Intel, MIPS, Power PC and DEC Alpha architectures. Select the appropriate ZIP file for your processor.
3. Unzip the file. You should have the following files:

`HTTPS.EXE`	The HTTP Server itself.
`HTTPS.CPL`	The Control Panel applet.
`MSVCRT20.DLL`	The C run-time library.
`HTTPS.HLP`	The Control Panel applet help file.
`HTTPS.DOC`	This manual in Word for Windows format.
`HTTPS.PS`	This manual, in postscript ready for printing.
`HTTPS.WRI`	This manual in Windows Write format.
`EGSCRIPT.ZIP`	Sample CGI script programs.
`COPYRITE.TXT`	The copyright statement for the software.
`READ.ME`	Summary of new features, *etc*.

4. Decide which directory you are going to put `HTTPS.EXE` in, and move it there. A good choice is the `\WINNT\SYSTEM32` directory, which is where many other services live. Using the Security/Permissions menu option in the File Manager, verify that the SYSTEM user has read permission for the file.
5. Move `HTTPS.CPL`, `MSVCRT20.DLL` and `HTTPS.HLP` to the `\WINNT\SYSTEM32` directory. **WARNING:** be careful not to overwrite any later version of `MSVCRT20.DLL` which you may find in that directory. Start the Control Panel from the Program Manager to verify that the HTTP Server applet is represented as an icon in the Control Panel.

6. Determine which version of `https` you have. To do this, at the Windows NT Command Prompt, type:
   ```
   https -version
   ```
   and the version number will be displayed. This manual covers `https` Version 0.99. (If the program reports a later version number, you will find a corresponding later manual in the files you unpacked from the ZIP archive.) You should also check the IP address of your machine using the command:
   ```
   https -ipaddress
   ```
   This will display the name of your machine (*eg* `emwac.ed.ac.uk`) and its IP address(es) as reported by the Windows Sockets API. If this information is incorrect, you need to reconfigure the TCP/IP software on your machine. The HTTP Server will not work if this address (or list of addresses if your machine has more than one network interface) is wrong.

7. If you have installed a previous version of the HTTP Server, you must remove it by typing:
   ```
   https -remove
   ```
   See section 2.4 for further information. You can use either the old or the new version of `HTTPS.EXE` to perform this remove operation. IF YOU ARE REPLACING VERSION 0.7 OR EARLIER WITH VERSION 0.8 OR LATER, READ THE NOTE AT THE END OF THIS SECTION!

8. Install `https` into the table of Windows NT Services (and simultaneously register it with the Event Logger) by running the program from the Windows NT command line, specifying the `-install` flag. (NOTE - it is vital that you execute this command using the copy of `HTTPS.EXE` which you placed in the `\WINNT\SYSTEM32` directory, and not using some other copy which you plan subsequently to delete.) For instance:
   ```
   https -install
   ```
   The program will register itself with the Service Manager and with the Event Logger, and will report success or failure. In the case of failure, see the section on Installation Problems below.

9. To verify that the installation has succeeded, start the Windows NT Control Panel and double-click on the Services icon. The resulting dialog should list HTTP Server as one of the installed services. If so, see the Configuration section of this manual for further instructions.

NOTE - UPGRADING FROM VERSION 0.7 OR EARLIER TO VERSION 0.8 OR LATER. With version 0.8, the "short name" by which the HTTP Service is known to the operating system changed from "HTTP Server" to "HTTPS". This means that the Windows NT registry stores information about the service in a different place from earlier versions. If you are upgrading from an earlier version of HTTPS, this has two consequences:

- The information stored in the registry by the earlier version of HTTPS must be deleted. This can be done by running the earlier version of HTTPS from the command line with the `-remove` flag.

Alternatively, version 0.8 or later (when you run it with the `-remove` option) will detect whether information relating to version 0.7 or earlier is present in the registry, and if so it will delete it. Both methods have occasionally been observed to cause the EventLog Service to terminate with an access violation. This is harmless - just restart the EventLog Service from the Services dialog in the control panel.

- When you replace version 0.7 or earlier with 0.8 or later, any events recorded in the Event Log by the earlier version will be unintelligible. This is because the Event Viewer program cannot find the information in the registry which tells it where the HTTPS.EXE file is located.

2.3. Installation Problems

The system says that HTTPS is not a Windows NT program

This is probably because you are trying to run an executable for the wrong sort of processor. Check you have unpacked the correct ZIP file for your processor type.

The system says that the HTTP Service won't install because of a "duplicate service name"

You must remove a previous version of the HTTP Service using the `https -remove` command before installing with `https -install`.

HTTPS waits for a while, then terminates with a "usage" message

You must not run `https` from the command line or from the File Manager, except with a command-line argument. The `https` program is a Windows NT "Service", and must be started through the Services dialog in the Control Panel.

2.4. Deinstalling

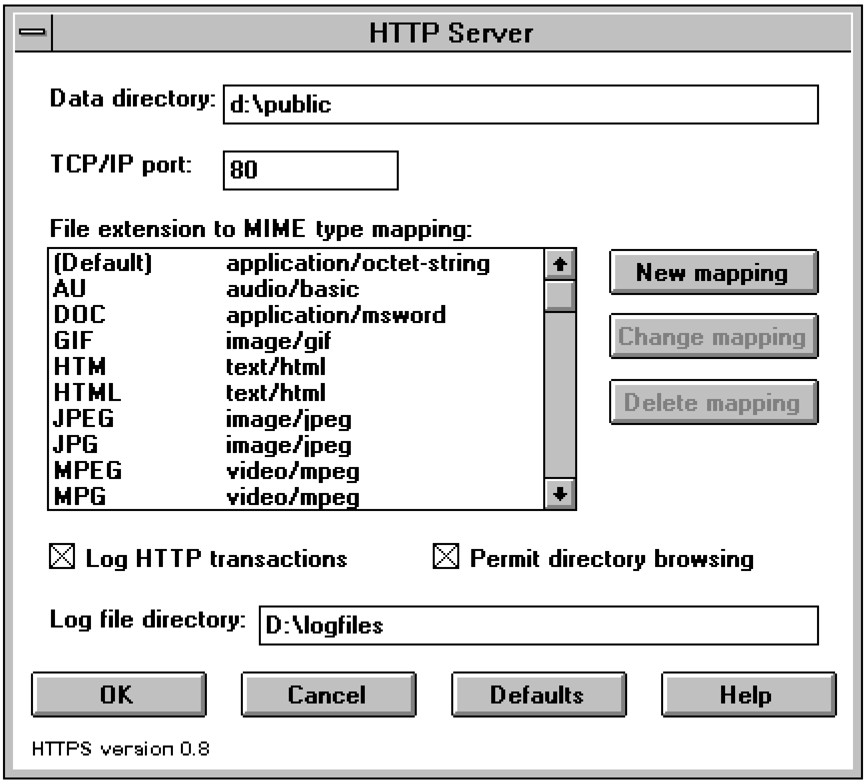
This section describes what to do if you want to remove the HTTP Server from your computer, or if you want to move the program to a new location.

1. If necessary, stop the HTTP Server using the **Stop** button in the Services dialog in the Control Panel.
2. At the Windows NT command line, run `https` with the `-remove` option:
   ```
   https -remove
   ```
   This will remove the HTTP Server from the Service Manager's list of services.
3. If you are deinstalling the HTTP Server, simply delete the `HTTPS.EXE` program and the `HTTPS.CPL` Control Panel applet.
4. If you want to move `HTTPS.EXE` to a new location, you must move the file, then type `https -install`. This informs the Service

Manager and Event Logger of the new location of the program. You will need to start the HTTP Server again from the Control Panel.

3. CONFIGURATION

The HTTP Server is configured using the HTTP Server applet in the Control Panel. The HTTP Server applet looks like this.



Note that the version number of the applet is displayed in the lower left-hand corner of the dialog. The version number reported by the command

```
https -version
```

must be the same as the version number of the applet. If there is no version number in the lower left-hand corner, you are using version 0.2 of the applet.

You can use this dialog to:

- Set the root of the directory tree containing the files you wish to make available on the World-Wide Web. Use the **Data directory:** field for this. Full details of how HTTP treats the files and directories in this directory tree are given in Section 5 of this manual. Default: `C:\HTTP`.
- Specify the TCP/IP port on which the HTTP Server listens for incoming HTTP connections. Use the **TCP/IP port:** field for this. The value must be a positive integer representing a legal and otherwise unused port. Default: 80.

- Specify the MIME type which corresponds to a given filename extension. This is covered in more detail below.
- Enable and disable the logging of HTTP transactions. If this box is checked, the HTTP Server will record each HTTP request it receives in a log file. See Section 4.3 of this manual for more information about logging. Logging is disabled by default.
- Specify the directory in which log files are stored. Use the **Log file directory:** field for this purpose. This is disabled unless the Log HTTP Transactions box is checked. The default is the Windows system directory (`\WINNT`).
- Permit the HTTP Data Directory tree to be browsed by HTTP clients. Further details on browsing are given in Section 5 of this manual. Browsing is disabled by default.
- Restore the default values of all the configuration settings. Click on the **Defaults** button to do this.

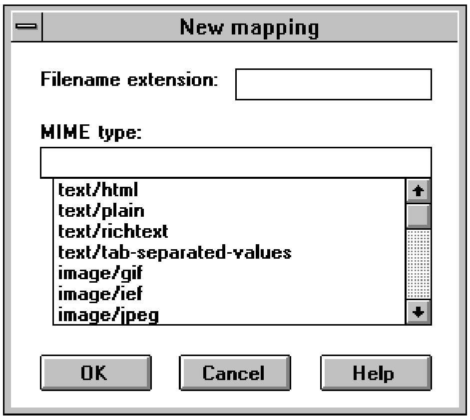
When you have finished making changes to the configuration, click on the **OK** button. The configuration will take effect the next time you start the HTTP Server. If the HTTP Server is already running, a dialog box to remind you to stop and restart it (using the Services dialog in the Control Panel) will be displayed.

3.1 File extension to MIME type mapping

The HTTP protocol represents the type of each file as a MIME type/subtype pair. The HTTP Server infers the MIME type of a file from the filename extension, using a mapping table. The mapping table is configurable, using the list in the Control Panel applet and the buttons to its right, labled **New Mapping**, **Change Mapping** and **Delete Mapping**. The default contents of the mapping table are as follows:

File extension	MIME type
HTM	text/html
HTML	text/html
TXT	text/plain
PS	application/postscript
RTF	application/rtf
PDF	application/pdf
ZIP	application/zip
DOC	application/msword
JPG	image/jpeg
JPEG	image/jpeg
GIF	image/gif
TIF	image/tiff
TIFF	image/tiff
XBM	image/x-xbitmap
WAV	audio/wav
AU	audio/basic
MPG	video/mpeg
MPEG	video/mpeg
Default	application/octet-string

To add a new mapping to the table, click the **New Mapping** button. The following dialog will be displayed:



Type the extension you wish to map into the Filename Extension field, and select the MIME type you wish to map it to from the list (or enter the MIME type in yourself if it's not in the list). Then select OK to add the new mapping to the main list. Note that you cannot create a new mapping for a filename extension already present in the mapping list - an extension may only occur once in the list.

To change an existing mapping, select it from the list in the main HTTP Server dialog, and click on the **Change Mapping** button. A dialog box, similar to the New Mapping box shown above, will appear. You can use this box to change the filename extension, or to select an new MIME type (or both).

To delete an existing mapping, select it from the list in the main HTTP Server dialog, and click on the **Delete Mapping** button.

3.2 Putting the Data Directory on a Fileserver

If the directory tree which you wish to make available to HTTP clients is located on a fileserver instead of on the local Windows NT machine, you will need to take special action.

Normally, directories on the fileserver are mapped to a drive letter on the local system. You might expect that simply using the mapped drive letter in the HTTP Server configuration dialog would have the desired effect, and indeed it does - until you log off the local machine.

Drive mappings are established only when someone logs on to the Windows NT machine. They are specific to a user, not to the machine. The HTTP Server is normally kept running, independently of whether someone is logged onto the machine or not. Often, the HTTP Server will be set to start up automatically when the operating system loads, when no-one is logged in and there are therefore no drive mappings in effect.

To overcome this, you can specify the HTTP Data Directory in the HTTP Configuration dialog using a UNC form of directory name - for instance:

`\\CLYDE\INFOSERVER`

Here, `CLYDE` is the name of the server, and `INFOSERVER` is the sharename of the directory which is to be served using HTTP.

4. OPERATION

4.1. Using the Services Dialog

You use the Services dialog in the Windows NT Control Panel for managing `https` operation.

After you install `https`, you can start it running by selecting it from the list of services in the dialog and clicking the **Start** button. If all goes well, a message box containing a rotating timer will appear while the service starts up, and will then disappear. The HTTP Server will then appear in the list of services with status "Started", and will be able to respond to HTTP clients.

If the service fails to start, it can be for one of several reasons. You may get a message box indicating the source of the problem. See the Troubleshooting section of this manual for further information.

You may want to arrange for `https` to start automatically when the system is started. You can do this using the **Startup** button in the Services dialog. You can also use this button to specify a different user ID for `https` to run under. See your Windows NT documentation for details. (NOTE - if you do run the HTTP Server under a different user ID, and you also set it to start automatically when the system boots, you may run into problems if the Netlogon service does not start before the HTTP service.)

Pausing the HTTP Server (using the **Pause** button in the Services dialog) causes the following behaviour:

- Any HTTP transactions currently underway will be unaffected. They will run to completion.
- Any new HTTP connections will be queued. When the service is **Resume**d, they will be accepted and processed.
- If more than five incoming connections are received while the service is paused, the extra connections will be rejected.

4.2. Error Logging

If an error in the operation of the server occurs, the error will be logged in the Application Event Log. This log may be viewed with the Event Viewer, which you will find in the Administrative Tools program group in the Program Manager. See your Windows NT documentation for details of how to use the Event Viewer.

Note that the Event Viewer uses the `HTTPS.EXE` program to interpret messages associated with events. Therefore, if you remove the `HTTPS.EXE` file, the HTTP Server events in your Application Event Log will be unintelligible.

The errors logged in the Application Event Log are usually associated with a `https` problem (*eg* a file I/O error, or a system call failure caused by lack of resources, or a problem with the configuration information).

Problems associated with the client (*eg* the client sends a URL which points to a file which does not exist) are recorded in the Application Event Log as Warning events.

When the HTTP Server is started or stopped, Information events are recorded in the Application Event Log.

See the Troubleshooting section of this manual for further details of the most common messages which may appear in the Event Log.

4.3 HTTP Transaction Logging

If you check the "Log HTTP Transactions" box in the HTTP Server configuration dialog, then for every HTTP request which the server receives, it will record a line of information in a log file. The log file is stored in the log file directory, which can also be configured in the dialog.

A new log file is created every day. The file name is of the form `HSyymmdd.LOG` - so that for instance the file corresponding to 4 July 1994 would be `HS940704.LOG`. For performance reasons, the current log file is kept open until the first HTTP transaction of the following day. When this transaction occurs, the preceeding day’s log file is closed, a new log file is opened, and the transaction is logged to it.

The information recorded is: the time and date of the request, the IP address or domain name of the server, the IP address of the client, the HTTP command, the URL requested, and the version of the HTTP protocol used (if there is no version, it means that HTTP 0.9 is in use).

5. THE HTTP DIRECTORY AND URLS

The HTTP Data Directory is the root of a directory tree within which you must locate files you want to make available to HTTP clients. Points in the file system above the Data Directory, or on other disks, are not accessible to HTTP clients. The Data Directory may be located on a disk which uses the FAT, HPFS or NTFS file systems.

URLs are relative to the Data Directory. Thus, if the HTTP client asks for a URL of the form:

`http://mymachine.mydomain.ac.uk/mydir/myfile.htm`

then the HTTP Server will send a file called `myfile.htm` in the `mydir` subdirectory of the Data Directory.

Files with the "hidden" or "system" attributes are ignored (*ie* treated as if they didn't exist).

The Data Directory tree must be accessible by the user ID under which `https` runs. Normally, this is the SYSTEM user ID.

By default, any file in the Data Directory tree for which the SYSTEM user has read permission can be retrieved by any HTTP client. If you want to prevent access to a particular file, use the Security/Permissions menu option in the File Manager to ensure that the SYSTEM user cannot access that file. Remember that the SYSTEM user is in the Administrators group (even though it does not appear in the User Manager list of members).

NOTE - the string `$HTTPS$` is reserved. You should not create any files or directories containing this string in their name, or they may not be visible to clients.

5.1 Browsing the HTTP Directory Tree

Suppose that an HTTP client asks for a URL of the form:

`http://mymachine.mydomain.ac.uk/mydir`

where `mydir` is a directory. The HTTP Server will do one of three things, depending on the contents of the directory and on how it is configured:

1. If a file called `DEFAULT.HTM` exists in the `mydir` directory, it will send that file to the client.
2. Otherwise, if Directory Browsing is enabled, the server will send a list of files and subdirectories within `mydir` to the client.
3. If Directory Browsing is not enabled, the server will send an error message to the client.

Thus, Directory Browsing enables a user to navigate through the Data Directory according to its hierarchical structure - rather like Gopher. You can enable Directory Browsing using the Control Panel configuration applet.

If you don't want a particular directory to be browsable, you can create a file called `NOBROWSE` in it. The file's contents are irrelevant - its presence simply causes step 2 above to be omitted.

Note that the top-level Data Directory itself may also have a `DEFAULT.HTM` or a `NOBROWSE` file.

6. Searching WAIS Indexes

The HTTP Server can search local WAIS databases. Before reading this section, you should read the WAIS Toolkit for Windows NT manual. (The WAIS Toolkit for Windows NT is available from the same place you obtained this HTTP Server software.)

When the server recieves an HTTP GET command for a URL which both:

- denotes an HTML file
- includes a search term

the HTTP Server passes the filename and the search term to the WAISLOOK program. The output from the WAISLOOK program is passed back to the client as an HTML document containing the result of the search. Note that for this to work, the name of the HTML file must be the same as the name of the WAIS database (apart from the extension).

To ensure that the HTTP Server can execute the WAISLOOK program, the `WAISLOOK.EXE` file should be located in the `\WINNT\SYSTEM32` directory.

There are a number of ways to exploit the search capabilities of the HTTP Server and the WAIS toolkit. The following example indicates one way of using these capabilities.

6.1. Example: A Simple Index of Text Files

This example illustrates how to set up a very common situation - you have a large number of text files (eg an archive of Usenet news messages) which you want to allow HTTP clients to search.

Prepare the directory structure

Let's assume that the HTTP data directory is `e:\web`, where drive `e:` is on an NTFS partition. Within that directory, you've created a subdirectory `News` to hold a number of newsgroup indexes. Under `News`, you've created a directory `alt.atheism` for that newsgroup. Now, create a directory `files` under `alt.atheism`, which is to hold the actual news messages, one to each file.

Here is the directory tree in graphical form:

```
                 e:\
                  |
                 web
                  |
       ┌─────┬────┴────┬─────┐
             |
            News
             |
   ┌─────────┴───┬───────────┬──────────┐
   |
alt.atheism
   |
   |
 files
   |
┌──────┬────┴──┬──────┐
|      |       |      |
103    104    105    106
```

The news messages in this example have filenames which do not include extensions. Normally, such messages are mapped by the mapping table to MIME type `application/octet-string`, which is not appropriate for these files. Therefore, you should configure the `(Default)` extension mapping in the HTTP Server configuration dialog so that these files are returned as `text/plain` by the HTTP Server.

Index the messages

Using a Windows NT command prompt, change to the HTTP data directory (`e:\web`). This is so that the filenames in the filename table which will be created by the indexing program are relative to the HTTP data directory.

Run the indexing command:

```
waisindex -d News\alt.atheism\index -t netnews News\alt.atheism\files\*
```

This will create files called `index.src`, `index.inv`, etc, in the `e:\web\News\alt.atheism` directory.

Test the index

To verify that the index functions correctly, use the WAISLOOK program to examine it:

```
waislook -d News\alt.atheism\index religion
```

This should return a list of files and headlines which contain the word "religion".

Create the HTML file

Now, you need to create an HTML file which invites the user to search the index. In the `e:\web\News\alt.atheism` directory, create a file `index.htm` which looks something like this:

```
<title>Search the alt.atheism news archive.</title>
<body>
<p>
```

```
Please enter the term(s) to search for.
<isindex>
</body>
```

Note that the HTML file must have the same name and path as the database, but of course should have an `HTM` or `HTML` extension to identify it as an HTML file according to the extension mapping table.

If the HTTP client user retrieves this file, then (because of the `<isindex>` tag) she will be able to enter a search term. When she types Return (or otherwise initiates a request), the client will request a URL such as:

```
http://myhost.mydomain.ac.uk/News/alt.atheism/index.htm?search+words
```

The HTTP Server will invoke the WAISLOOK program to search the index, and will return a HTML file containing the list of matching messages.

Updating the index

When a new message (say `107`) arrives in the `e:\web\News\alt.atheism\files` directory, you should run the WAISINDEX command to update the index. Change to the `e:\web` directory, and issue the command:

```
waisindex -d News\alt.atheism\index -t netnews -a News\alt.atheism\files\107
```

This will add the new file to the directory. Normally, of course, you would arrange for this command to be executed automatically in a batch file.

7. Scripts and Forms

The HTTP Server conforms to the Common Gateway Interface (CGI) 1.1 standard (see `http://hoohoo.ncsa.uiuc.edu/cgi/`). This means that you can write your own programs (known as CGI scripts) which can be invoked by WWW clients, and which run on the Windows NT machine which is running `https`. CGI scripts could for instance be used to provide a gateway between WWW and a database package, or to process user input read from an HTML form.

7.1 Script Execution

A script is an executable Windows NT program with a `.EXE` extension. It must be located within the HTTP Data Directory tree. It must be a Windows NT "console" application - one which could in principle be used from the Windows NT command line. A script cannot be a GUI program, or another Windows NT Service. Due to a bug in Windows NT 3.1, a script cannot be a DOS program either. [1]

A script may also be a Perl program with a `.PL` extension.

Full details of how to write CGI scripts are given in the CGI specification referred to above. Briefly, the script accesses information about how it was invoked through Environment variables, reads any information supplied by the client in a POST request via `stdin`, and sends output to the client through `stdout`. The file `EGSCRIPT.ZIP` contains two example scripts (executable and C source) and a corresponding makefile. The makefile assumes you have the Windows NT Software Developer's Kit (SDK) or Microsoft's Visual C++ 2.0 or greater installed.

A `.EXE` CGI script will be executed in two circumstances:

- When the HTTP Server recieves a POST request for a URL which corresponds to an executable file.
- When the HTTP Server recieves a GET request for a URL which both:
  - corresponds to an executable file
  - contains a query string

If the script is specified in a GET request *without* a query string in the URL, the script will be sent to the client as an `application/octet-string` file (or whatever the `.EXE` extension corresponds to in the mapping table). If you do not wish to allow users to download your script like this, you should use the File Manager to assign execute-only permission to the script, so that it can't be read by the SYSTEM user.

---

[1] See Microsoft's Windows NT KnowlegeBase article "BUG: Redirecting Output to an MS-DOS Application", PSS ID Number Q105303. Note that the workaround suggested in that article is not appropriate for HTTPS.

7.2 Using Forms with HTTPS

HTML lets you create "forms", which allow the user to submit information (such as a complex database query) to a HTTP server. A description of how to create forms may be found at:

`http://hoohoo.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html`

When a client program submits a form, it may do so using either the GET or the POST method. The query string (in the former case) or the body of the request message (in the latter case) will contain the data entered into the form by the user. See the scripts in `EGSCRIPT.ZIP` for an example of how to decode the data from a form.

Here is an example form defined in HTML.

```
<head><title>A test form</title></head>
<body>
A test form for checking out the POST method in HTTPS.
<p>
<form
action="http://host.domain.edu/scripts/egscript.exe"
method="POST">
Field 1 (text entry field) <input type="text"
name="field1"> <p>
Field 2 (checkbox) <input type="checkbox"
name="field2"> <p>
Field 3 (radio buttons)<br>
<input type="radio" name=field3 value="Male" checked>
Male<BR>
<input type="radio" name=field3 value="Female"> Female
<BR>
<input type="radio" name=field3 value="Neuter">
Neuter<BR>
<p>
Field 4 (select)
<select name="field4">
<option> First
<option> Second
<option> Third
<option> Fourth
</select>
<p>
Field 5 (textarea) <textarea name=field5 rows=4
cols=40>
</textarea>
<p>
Field 6 (submit) <input type="submit" value="Go">
</form>
</body>
```

The `action` attribute of the form is the URL of the script, and (in this example) the `method` attribute is `POST`.

If the `action` attribute of the example form is changed to:

`http://host.domain.edu/scripts/egscript.exe/foo/bar` then the script program `scripts/egscript.exe` will still be executed. The `/foo/bar` part will be passed to the script in the `PATH_INFO` environment variable. Note that it is not a good idea to create directories with names ending in `.EXE` - the server will get confused.

7.3 Command-Line Parameters

If a script is invoked with a URL which includes a query string, then the query string will be passed to the script in the command line. The query string is decoded before being passed to the script - any "`+`" characters are replaced by spaces, and any `%xx` sequences (where `xx` are two hex digits) are replaced by the corresponding ASCII character. (If required, the undecoded query string is available to the script in the `QUERY_STRING` environment variable.)

NOTE - if the undecoded query string contains an "=" character, the CGI specification states that the query string shall not be passed to the script in the command line. If you want to pass a "`=`" in the command line, encode it in the URL as `%3D`.

Here is an example URL:

```
http://host.domain.edu/scripts/myscript.exe/foo/bar?arg1+arg2
```

This would result in the following command being executed:

```
scripts\myscript.exe arg1 arg2
```

7.4 CGI Scripts in Perl

If the URL in a GET or POST request (regardless of whether there is a query string) corresponds to a file with an extension of `.PL`, then the server takes special action. It attempts to execute the Perl interpreter `PERL.EXE`, passing it the script file name on the command line.

In order for this to work, a Perl interpreter must be installed on your system, and the directory containing the Perl executable must be located on the system PATH. Note that it is not sufficent that the directory be in a user's path - it must be in the PATH System Environment Variable - use the System icon in the Control Panel to ensure this. Furthermore, note that in Windows NT 3.51, in order for a change to the System Environment Variables to take effect *for services*, you must shut down and restart Windows NT.

Here is an example of a very simple PERL program which could be executed in this way:

```
#!./perl
print "200 OK\n";
print "Content-type: text/plain\n";
print "\n";
print "hello world\n";
```

7.5 Supported Environment Variables

The CGI standard specifies certain environment variables which are used for conveying information to a CGI script. The following subset of those environment variables are supported by HTTPS:

```
CONTENT_LENGTH
CONTENT_TYPE
GATEWAY_INTERFACE
HTTP_ACCEPT
PATH_INFO
QUERY_STRING
REMOTE_ADDR
REQUEST_METHOD
SCRIPT_NAME
SERVER_NAME
SERVER_PROTOCOL
SERVER_PORT
SERVER_SOFTWARE
```

Other HTTP headers received from the client are available in environment variables of the form `HTTP_*`. For instance, the `User-Agent:` header value is available in `HTTP_USER_AGENT`. Note that "`-`" in the header names is replaced by "_" in the corresponding environment variable names.

8. Clickable Images

The HTTP Server supports the use of "clickable images" - in other words, it can return different documents depending on where in an image the user clicks their mouse.

Here is a typical example of a "clickable image" document in HTML.

```
<html>
<head><title>WWW Servers in UK</title><head>
<body>
<h1>WWW Servers in the UK</h1>
<hr>
To have a server added to the map send email to:
<a href="http://www.ed.ac.uk/webperson.html">
<address>webperson@ed.ac.uk</address>
</a>
<p>
<hr>
<a href="/ukmap.map">
<img src="/Images/ukmap.gif" ismap>
</a>
<hr>
</body>
```

Note the use of the `ISMAP` attribute in the `IMG` tag. This tells the client to append the mouse coordinates to the URL when it sends it to the server.

The URL in the enclosing `<a>` anchor element must refer to a "map" file on the HTTP Server, with the extension `.MAP`. This file contains information about how to map the coordinates of the mouse click to another URL. The mapped-to URL is the one actually returned to the client.

8.1 Map File Format

A map file is a text file consisting of definitions, comments and blank lines. Comment lines start with a `#`. Definition lines have one of the following four forms:

`default` *`URL`*

`circle` *`x y r URL`*

`rectangle` *`x0 y0 x1 y1 URL`*

`polygon` *`x0 y0 x1 y1 x2 y2 ... URL`*

The keywords may be abbreviated to `def`, `circ`, `rect` and `poly` respectively.

The `default` keyword defines the URL to be used if the mouse click falls outside any other shape defined in the file. There must always be a default statement in the map file.

The `circle` statement defines a circle with centre *(x,y)* and radius *r*, and the URL to use if the mouse click lies within the circle.

The `rectangle` statement defines a rectangle with top left at *(x0,y0)* and bottom right at *(x1,y1)*, and the URL to use if the mouse click lies within the rectangle.

The `polygon` statement defines a polygon with vertices at *(x0,y0)*, *(x1,y1)*, *(x2,y2)*, ... (up to 100 vertices), and the URL to use if the mouse click lies within the polygon.

The coordinates are measured from the top left-hand corner of the image. The coordinates may be separated within the statement by any combination of blank spaces, TABs, commas and round brackets, as you can see in the example below. (This means that you can use image configuration files in exactly the same format as for the CERN `httpd` server.)

The *`URL`* at the end of each line can either point at a local file (in which case it must start with a forward slash), or it can refer to a document on another server (in which case it must be a full URL). In the latter case, the HTTP Server will send a "302 Found" redirection message, containing the URL in question, to the client. Clients which are capable of understanding the redirection message will automatically fetch the document. Other clients will display an explanatory message containing a hyperlink to the document.

Here is an example map file. Suppose that your image is 1000 by 1000 pixels in size.

```
# Circle at centre of image
circle (500,500) 100 /local/file.htm

# Rectangle at lower right
rectangle 550 550 850 850 http://some.other.host.uk/some/file.html

# Triangle at lower left
polygon 10,700  200,700  10,900 /another/local/file.htm

# Use this URL if mouse is outside any of above shapes
default /error.htm
```

Note that there is a slight overlap between the circle and the rectangle. If the mouse click occurs in the overlap, the first matching shape in the file will determine the URL which will be returned - in this case the circle.

Hints for using MAP files

- If the URL which you want to use in a MAP file contains a # (indicating a location within a file), you should quote the full URL (starting `http://`) in the MAP file so that the server issues a redirection message to the client.
- If you want the URL in a MAP file to point to a document which is not in the same directory as the MAP file itself, you should either quote the full URL as above, or else use the HTML construct `<base href="....">` in the pointed-to document. This ensures that the client handles relative URLs within the document correctly.

- In general, trouble with clickable image maps can usually be overcome by quoting the full URL (starting `http://`) in the MAP file.

9. TROUBLESHOOTING

This chapter lists some of the problems which you may have in running the HTTP Server, and describes how to overcome them.

9.1 Errors Starting the HTTP Server

When starting the HTTP Server, you may see one of the following error messages.

**Could not start the HTTP Server service on \\yourmachine.
Error 0002: The system cannot find the file specified.**

The Service Manager could not locate `HTTPS.EXE`. This probably means it has been moved, or has not been installed correctly. Remove and reinstall `https` - see section 2.4 for details.

**Could not start the HTTP Server service on \\yourmachine.
Error 0005: Access is denied.**

`HTTPS.EXE` is inaccessible to the SYSTEM user. By default, the Service Manager starts the HTTP Server process running under a user ID of SYSTEM. The executable file for the service must be readable by this user.

**Could not start the HTTP Server service on \\yourmachine.
Error 2140: An internal Windows NT error occurred.**

This usually means a problem with the configuration of the HTTP Server. Further information detailing the precise problem will be recorded in the Application Event Log - see the subsection on Error Logging elsewhere in this manual for details on how to view this information. A description of some of the most common errors is given below. Note - you may get this error, or a more alarming error telling you that the HTTPS service is not correctly written, if you installed HTTPS under Windows NT 3.1, and later upgrade to Windows NT 3.5. To fix this problem, simply remove and reinstall HTTPS.

9.2 Errors Recorded in the Event Log

This section records some of the error messages which may appear in the Application Event Log. Many of these are self-explanatory.

Windows Sockets library function "bind" failed. The address or port is already in use.

One of the following situations may give rise to this error:

The TCP/IP port you have specified in the HTTP Server configuration dialog is conflicting with another application. Choose a different port number.

The IP address which the HTTP Server is using is incorrect. Start the Network Control Panel applet, and configure the TCP/IP software to use the correct IP address.

Windows Sockets library function "accept" failed. The call was cancelled.

This indicates that the HTTP Server terminated abnormally for some reason. Restart the service.

APPENDIX I - REGISTRY ENTRIES

This appendix lists entries added *explicitly* to the Windows NT Registry by the HTTP Server. A number of other entries are added to the Registry implicitly by the Service Control Manager. The information in this appendix is not guaranteed to remain unchanged between releases of the HTTP Server (for instance there were significant changes between versions 0.7 and 0.8). The information is intended for advanced users of Windows NT who understand the function and structure of the Registry.

The Service Control Manager creates the following entry in the `HKEY_LOCAL_MACHINE` database:

`SYSTEM\CurrentControlSet\Services\HTTPS`

Under this entry, the HTTP Server itself creates a `Parameters` key containing the following configuration entries.

Entry name	**Entry type**	**Description**
`Directory`	`REG_SZ`	HTTP Data Directory name
`PortNo`	`REG_DWORD`	TCP/IP port number
`DefaultMIMEType`	`REG_SZ`	MIME type used for file extensions not listed explicitly
`BrowsingEnabled`	`REG_DWORD`	Non-zero if browsing is enabled
`LogDirectory`	`REG_SZ`	Directory where log files are stored
`LoggingEnabled`	`REG_DWORD`	Non-zero if logging is enabled
`ExtensionMapping`	See below	

The `ExtensionMapping` entry holds the mapping table. It contains a key name for each file extension, and the value of the key (type `REG_SZ`) is the corresponding MIME type.

APPENDIX II - FORMAL COMMAND SYNTAX

The formal description of the `https` command options is as follows.

Syntax

```
https [-remove | -install] [-version] [-ipaddress]
```

Description

The `https` command installs or removes the HTTP Server.

Options

`-install`	Add the HTTP Server to the list of installed services. Make sure you are executing a copy of `HTTPS.EXE` which you aren't planning to delete later.
`-remove`	Remove the HTTP Server from the list of installed services. This will also delete the HTTP Server configuration information from the Registry.
`-version`	Report the version number of the HTTP Server.
`-ipaddress`	Report the IP addresses on which the HTTP Server is listening.