# W3C httpd 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.

---

## In This Section...



---

## Important Note!

W3C httpd versions 2.15 and newer have **two** script interfaces:

- The original, very easy-to-use, interface, that was introduced in version 2.13.
- The official CGI, Common Gateway Interface, which enables scripts to be shared between different server implementations (NCSA server, Plexus, etc).

**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.

---

## 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

### 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/myscript
```

### 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.

---

# 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.)

# General Things to Remember in Scripts

Make sure that

- the script file has execute permission set, e.g.

```
chmod 755 your_script
```

- the shell to execute the script is specified in the first line of the script, e.g.:

```
#!/bin/sh
```

- All scripts are executed from the ServerRoot.

# 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).

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

For VMS the script result is generated a bit differently. Here are two examples:

```
$ write sys$output "Content-Type: text/html"
$ write sys$output ""
$ write sys$output "<HEAD>"
$ write sys$output "<TITLE>Script test</TITLE>"
$ write sys$output "</HEAD>"
$ write sys$output "<BODY>"
$ write sys$output "<H1>My First Virtual Document</H1>"
$ write sys$output "<b>This is a test.</b>"
$ write sys$output "</BODY>"
$ exit
```

and

```
$ write sys$output "Content-Type: text/html"
$ write sys$output ""
```

```
$ write sys$output "<TITLE>Script test</TITLE>"
$ write sys$output "</HEAD>"
$ write sys$output "<BODY>"
$ write sys$output "<H1>My First Virtual Document</H1>"
$ write sys$output "<b>This is a test.</b>"
$ write sys$output "<ISINDEX>"
$ write sys$output "<pre>"
$ show symbol p1
$ write sys$output ""
$ show user 'p1
$ write sys$output "</pre>"
$ write sys$output "</BODY>"
$ exit
```

---

## 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://www.w3.org/pub/WWW/
```

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/
```

Understand, that this is a **virtual path**, so after translations it might be, for example, `/Public/Web/`.

**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.

---

## 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>
```

```
Yep, seems to work.
</BODY>
```

---

# 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...
```

---

*httpd@w3.org, July 1995*