

HTTP

HTTP "One shot" Transaction

HTTP defines a request/response pair. It most often operate as a "one shot" transaction. The server fulfills the request and closes the connection. (it's a little different in HTTP/1.1.

HTTP Request

port 80

Request sent from client to server

Server sends back one response

Stateless

"Stateless" -- each request/reply pair (HTTP 1.0) uses its own connection. The dialog does not group logically related collections of requests. Keeps things simple, but causes some inefficiency + make it hard to create dialogs which keep state. We'll cover this in more detail in the CGI section. This is one of the areas addressed in the HTTP 1.1 revision.

GET Request

Request a resource

GET request HTTP\r\n\r\n

GET /foo.html HTTP/1.0\r\n\r\n

Actually use `\015\012` in your code in case "\n" has been mapped to the local end-of-line.

Complex URLs

`http://foo.com/a/b/bar.html?a=b`

scheme

http

host

foo.com

path

/a/b/

file

after the last /
bar.html

suffix

after the file, starting with # or ?
may contain almost anything

Request

The request is basically the directory path and file from the absolute form of the URL. `http://foo.com/dir/b.html` : request = `/dir/b.html`

Suffix

A "#foo" suffix is **not** sent as part of the request
The "#" part of the url is handled on the client side.
`http://foo.com/dir/b.html#bar` : request = `/dir/b.html`

? Suffix

A "?a=b&c=13" suffix is sent as the end part of the request -- the server (typically a CGI) does use it as part of the request.
`http://foo.com/dir/b.html?bar=http://www.yahoo.com/` request = `/dir/b.html?bar=http://www.yahoo.com/`
The suffixes may contain weird characters, so identify the "#" or "?" scanning from the left.

Get

telnet 80

Lab exercise: try telneting to port 80 of your favorite web server. Type the GET directive in by hand and watch what happens. On unix, the command looks like...

```
telnet www.stanford.edu 80
```

Get Options

Usually the GET includes optional information on subsequent lines...

```
GET request HTTP/1.0
User-agent: Mozilla/2.02      -- the type of client software
Accept: image/gif, image/jpeg, */*  -- return type preferences of the client
Referer: url      -- page where the client was just before making this request. The
server could use this to figure out who's pointing to it.
If-Modified-Since: date  -- use to only retrieve a document if it is newer than the
one the client has in cache.
```

Blank line

A final blank line after all the options marks the end of the request. So the end of the request reads "\015\012\015\012".

HEAD

Similar to GET, but retrieves the header but not the data. Use to test for the header of a page but without actually downloading all the HTML. Some servers don't implement HEAD.

The Empty Request

What about the URL `http://www.cnn.com`

The directory path and file are the empty string.

Could just send the empty string as the request in the HTTP dialog -- many servers interpret that as `"/`.

This used to be a real error, but it was so common that servers were special cased to just interpret it as `"/`.

In site sucker, you may special case change the empty string request to `"/`, but you'll need to be consistent to not mess up your cycle detection. This only applies if the request would be the empty string.

HTTP Response

The server responds with a header section which describes the document, followed by a blank line, followed by the document data.

HTTP/1.0 200 OK

The very first line of the response has the form VERSION CODE REASON -- e.g. HTTP/1.0 200 OK

VERSION = the version of HTTP which the server is speaking

CODE = a numeric code which compactly indicates how the request is going

REASON = a human-readable statement of the CODE

Other Fields

The rest of the HTTP header contains a variety of information, much of it optional, about the document to be returned. The two most important fields are...

Content-type: *MIME-type* -- type of the document returned

Content-Length: *length* -- the size in bytes of the document

HTTP Response Example

GET / http/1.0

HTTP 200 Document follows

Date: Wed, 23 Apr 1997 16:55:45 GMT

Server: NCSA/1.5.2

Last-modified: Sun, 18 Feb 1996 20:25:48 GMT

Content-type: text/html

Content-length: 2225
 <blank line>
 <the document data starts here>

Status Codes

200 = OK

3xx = Minor client error— document in another location

301 = Moved permanently. The Location: field of the header gives the correct URL.

302 = Moved temporarily -- the server may suggest the correct url using a location: in the header

304 = Not Modified. The request had If-Modified-Since: field, but the document has not been modified, so the client should use their cached copy.

400 = Real client error

bad request, document not found, not allowed

400 = syntax error

401 = Unauthorized

403 = Forbidden -- "permission denied"

404 = Not found

500 = Server error

service not implemented, service not available

503 = Service unavailable. The server is temporarily not able to provide the service. The header may contain a Retry-After: field to indicate when the client might give it another shot.

MIME Types

Multipurpose Internet Mail Extensions -- a standard which predates HTTP for identifying different types of data for inclusion in email messages.

content-type/sub-type [; aux-info]'

text/html

text/plain

text/plain ; charset = us-ascii

multipart/mixed ; boundary = SpecialBoundaryString

application/pdf

application/postscript

audio/basic -- .au audio

image/jpeg

video/quicktime

"X-.." = experimental

Since the MIME type is in HTTP response header, the server is taking responsibility for identifying the type of the data. Servers may be programmable in this respect. Most servers use the file extension (.jpeg, .html .htm) to guess the MIME type to claim when they send the bytes back.

Example 1 — HTML text

```

elaine45:~/193i> telnet www.stanford.edu 80
Trying 171.64.14.239...
Connected to www3.Stanford.EDU (171.64.14.239).
Escape character is '^]'.
GET /class/cs193i/test/basic.html HTTP/1.0

HTTP/1.1 200 OK
Date: Fri, 21 Apr 2000 18:03:20 GMT
Server: Stronghold/2.4.1 Apache/1.3.3 C2NetEU/2409 (Unix) mod_fastcgi/2.2.2
Last-Modified: Fri, 21 Apr 2000 09:18:26 GMT
ETag: "70741d1c-160-39001ce2"
Accept-Ranges: bytes
Content-Length: 352
Connection: close
Content-Type: text/html

<html>
<head>
<title>Basic</title>
</head>

<body>
<h1>Basic</h1>

A few straightforward links...
<ul>
<li><A HREF="http://www.yahoo.com/">yahoo</a>
<li><a href=http://www.theonion.com/>The Onion</a>
<li><A HREF=http://www.stanford.edu/class/cs193i/test/a.html>a absolute</A>
<li><a href=a.html>a relative</a>
</ul>

</body>
</html>

Connection closed by foreign host.

```

Example 2 — Error reported in both header and body

```
Sunburn:~ > telnet www-cs-faculty 80
GET /~nick/foobar/ http/1.0
```

```
HTTP/1.0 404 Not Found
Date: Mon, 29 Apr 1996 16:33:53 GMT
Server: NCSA/1.4.1
Content-type: text/html
```

```
<HEAD><TITLE>404 Not Found</TITLE></HEAD>
<BODY><H1>404 Not Found</H1>
The requested URL /~nick/foobar was not found on this server.<P>
</BODY>
```

Example 3 — Trailing / Omit the '/' URL forwarding -- is that trailing slash really necessary? Yes!

```
Sunburn:~ > telnet www-cs-faculty 80
GET /~nick HTTP/1.0
```

```
HTTP/1.1 301 Moved Permanently
Date: Mon, 26 Apr 1999 19:43:11 GMT
Server: Apache/1.3.0 (Unix)
Location: http://Sunburn.Stanford.EDU/~nick/
Connection: close
Content-Type: text/html
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
  The document has moved
<A HREF="http://Sunburn.Stanford.EDU/~nick/">here</A>.<P>
</BODY></HTML>
```

Location: url

The **location:** header field is used for some HTTP returns to give a new URL for the client to try. Most browsers ret-try the url automatically. That's why you can omit the trailing / in many cases, and the browser/server will still get you to the right place (with twice as many transactions).