HTTP Messages, Message Formats, Methods and Status Codes

As we saw in the operational overview of the Hypertext Transfer Protocol, HTTP is entirely oriented around the sending of client requests and server responses. These take the form of *HTTP messages* sent between clients and servers. As with all protocols, HTTP uses a special format that dictates the structure of both client request messages and server response messages; understanding how these messages work is a big part of comprehending HTTP as a whole.

In this section I describe the messages used by HTTP and the specific commands and responses issued by clients and servers. I begin with a look at the generic HTTP message format and the major components of every HTTP message. I then discuss the specific formats used for both requests and responses. I explain the different types of HTTP methods (commands) used in client requests, and the HTTP status codes used in server replies.

Quick navigation to **subsections** *and* regular topics *in this section*

HTTP Generic Message Format

As we've already seen, all of the communication between devices using the Hypertext Transfer Protocol takes place via *HTTP messages*, of which there are only two types: *requests* and *responses*. Clients usually send requests and receive responses, while servers receive requests and send responses. Intermediate devices such as gateways or proxies may send and receive both types of message.

All HTTP messages are created to fit a message structure that the standard calls the *generic message format*. Like most of the other TCP/IP messaging protocols such as SMTP and NNTP, HTTP messages do not use a binary message format; rather, they are text-based. HTTP messages are based loosely on the electronic mail RFC 822 and 2822 message standards, as well as theMultipurpose Internet Mail Extensions

(MIME) standard. I say “loosely” because HTTP messages are similar in construction to e-mail messages but do not strictly follow all of the e-mail or MIME format requirements. For one thing, not all of the RFC 822 and MIME headers are used; there are other differences as well, which we will soon examine.

The HTTP generic message format is as follows:

<start-line>
<message-headers>
<empty-line>
[<message-body>]
[<message-trailers>]

You can see that this is pretty much the same as the format used for e-mail messages and for Usenet newsgroup messages too: headers, an empty line and then a message body. All text lines are terminated with the standard “CRLF” control character sequence; the empty line contains just those two characters and nothing else. The headers are always sent as regular text; the body, however, may be either text or 8-bit binary information, depending on the nature of the data to be sent. (This is another way that HTTP does not adhere strictly to the RFC 822 standard; see the discussion of entities and media types for a full discussion.)

The *start line* is a special text line that conveys the nature of the message. In a request, this line indicates the nature of the request, in the form of a *method*, as well as specifying a URI to indicate the resource that is the object of the request. Responses use the start line to indicate status information in reply to a request. More details on the use of the start line can be found in the next two topics that detail request messages and response messages respectively.

HTTP Generic Message Format

As we’ve already seen, all of the communication between devices using the Hypertext Transfer Protocol takes place via *HTTP messages*, of which there are only two types: *requests* and *responses*. Clients usually send requests and receive responses, while servers receive requests and send responses. Intermediate devices such as gateways or proxies may send and receive both types of message.

All HTTP messages are created to fit a message structure that the standard calls the *generic message format*. Like most of the other TCP/IP messaging protocols such as SMTP and NNTP, HTTP messages do not use a binary message format; rather, they are text-based. HTTP messages are based loosely on the electronic mail RFC 822 and 2822 message standards, as well as theMultipurpose Internet Mail Extensions (MIME) standard. I say "loosely" because HTTP messages are similar in construction to e-mail messages but do not strictly follow all of the e-mail or MIME format requirements. For one thing, not all of the RFC 822 and MIME headers are used; there are other differences as well, which we will soon examine.

The HTTP generic message format is as follows:

<start-line>
<message-headers>
<empty-line>
[<message-body>]
[<message-trailers>]

You can see that this is pretty much the same as the format used for e-mail messages and for Usenet newsgroup messages too: headers, an empty line and then a message body. All text lines are terminated with the standard "CRLF" control character sequence; the empty line contains just those two characters and nothing else. The headers are always sent as regular text; the body, however, may be either text or 8-bit binary information, depending on the nature of the data to be sent. (This is another way that HTTP does not adhere strictly to the RFC 822 standard; see the discussion of entities and media types for a full discussion.)

The *start line* is a special text line that conveys the nature of the message. In a request, this line indicates the nature of the request, in the form of a *method*, as well as specifying a URI to indicate the resource that is the object of the request. Responses use the start line to indicate status information in reply to a request. More details on the use of the start line can be found in the next two topics that detail request messages and response messages respectively.

HTTP Request Message Format

The client initiates an HTTP session by opening a TCP connection to the HTTP server with which it wishes to communicate. It then sends *request messages* to the server, each of which specifies a particular type of action that the user of the HTTP client would like the server to take. Requests can be generated either by specific user action (such as clicking a hyperlink in a Web browser) or indirectly as a result of a prior action (such as a reference to an inline image in an HTML document leading to a request for that image.)

HTTP requests use a message format that is based on the generic message format described in the preceding topic, but specific to the needs of requests. The structure of this format is as follows (see Figure 317):

<request-line>
<general-headers>
<request-headers>
<entity-headers>
<empty-line>
[<message-body>]
[<message-trailers>]

Figure 317: HTTP Request Message Format

This diagram shows the structural elements of an HTTP request and an example of the sorts of headers a request might contain. Like most HTTP requests, this one carries no entity, so there are no entity headers and the message body is empty. See Figure 318 for the HTTP response format.

HTTP Response Message Format

Up and down; east and west; black and white; yin and yang. Well, you get the idea. Each request message sent by an HTTP client to a server prompts the server to send back a *response message*. Actually, in certain cases the server may in fact send two responses, a preliminary response followed by the real one. Usually though, one request yields one response, which indicates the results of the server's processing of the request, and often also carries an entity (file or resource) in the message body.

Like requests, responses use their own specific message format that is based on the HTTP generic message format. The format, shown in Figure 318, is:

<status-line>
<general-headers>
<response-headers>
<entity-headers>
<empty-line>
[<message-body>]
[<message-trailers>]

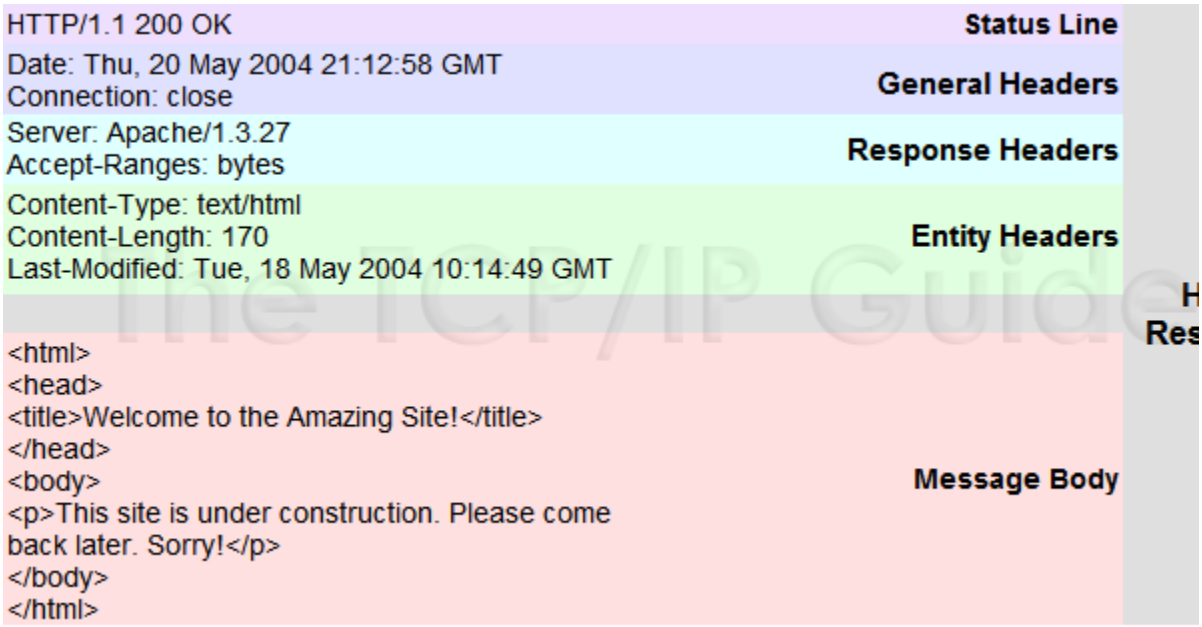

Figure 318: HTTP Response Message Format

This figure illustrates the construction of an HTTP response, and includes an example of both message headers and body. The status code “200” indicates that

this is a successful response to a request; it contains a brief text HTML entity in the message body. See Figure 317 for the HTTP request format.