

Request and Response Messages

The *HTTP protocol* consist into a *request message*, sent from a *client* to a *web server*; and a *response message*, sent from the *server* to the *originating client*.

Request message general format:

HTTP Request Line
 HTTP Request Headers
 (empty line)
 HTTP Request Body

Response message general format:

HTTP Response Line
 HTTP Response Headers
 (empty line)
 HTTP Response Body

HTTP Request Line

Format: "METHOD PATH PROTOCOL"
 METHOD: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS, CONNECT, PATCH
 PATH: the path of the resource
 PROTOCOL: HTTP/1.1

Example: GET /images/logo.png
 HTTP/1.1

HTTP Request Headers

"NAME: VALUE"

NAME: [A-Za-z0-9] [A-Za-z0-9-]+
 VALUE: US-ASCII octets

HTTP Request Headers (cont)

Headers are extra information for the request. There are many standard headers and you can create your own.
 See Common HTTP Request Headers

Example set of headers:

Host: en.wikipedia.org
 User-Agent: Mozilla/5.0 Firefox/64.0
 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
 Accept-Language: en-GB,en-US;q=0.8,en;q=0.6,fr-FR;q=0.4,fr;q=0.2
 Accept-Encoding: gzip, deflate, br
 Connection: keep-alive
 Cookie: Auth=8QXA5fSQeZAEKZVG-6irJmWvQ8KtQKAaj

HTTP Request Body

If the HTTP method used is POST or PUT, the request may be followed by a body. It can be a file or a specific form of data.
 The Content-Type header must be filled with a MIME type to indicate the type of content.
 The body is separated from the header by two line feeds (\n).

HTTP Response Line

Format: PROTOCOL STATUS REASON
 PROTOCOL: HTTP/1.1
 STATUS: Any HTTP Status Code
 REASON: A reason message

HTTP Response Line (cont)

The reason message is usually the label associated to the status code. Some APIs may use this text field to specify an error message.

Examples:

HTTP/1.1 200 OK
 HTTP/1.1 404 Not Found

HTTP Response Headers

"NAME: VALUE"

NAME: [A-Za-z0-9] [A-Za-z0-9-]+
 VALUE: US-ASCII octets

Headers are extra information for the response. There are many standard headers and you can create your own.
 See Common HTTP Response Headers

HTTP Response Body

The is usually a response body after the response headers. It can be a file or a specific form of data.
 The Content-Type header must be filled with a MIME type to indicate the type of content.
 The body is separated from the headers by two line feeds (\n).

HTTP Request Methods

GET: used to retrieve a resource. Has no request body.
 POST: used to submit a new resource (path) or send data. Usually has a body.



HTTP Request Methods (cont)

HEAD: used to preview the result of a GET operation. Has no request body and no respond body.

PUT: used to submit an update to an existing resource

DELETE: used to delete the specified resource

TRACE: echoes the received request for tracing purposes

OPTIONS: verify the server supports a specified request (see [Preflight requests](#))

CONNECT: used by HTTPS

PATCH: allows partial modification of a resource

Common HTTP Request Headers

Accept	List of MIME types supported
Accept-Language	List of languages read by the user
Content-Length	Length in bytes of the request body
Content-Type	MIME type of the request body
Cookie	List of cookies stored by the client
Host	Host name of the website
User-Agent	Identification string for the web browser

There are [many more available](#).
You can create your own headers.

HTTP Status Codes

1xx	Informational
100	Continue
2xx	Successful
200	OK
201	Created
202	Accepted
204	No Content
3xx	Redirection
301	Moved permanently
302	Found
304	Not Modified
308	Permanent Redirect
4xx	Client Error
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
5xx	Server Error
500	Internal Server Error
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

There are many [other codes](#); these are the most used. You should not create your own codes.

Common HTTP Response Headers

Cache-Control: Indicates client caching conditions

Content-Length: Length of the response body in bytes

Content-Type: MIME type of the response body

Expires: Client is allowed to keep the resource in cache

Location: Redirection URL

Server: name of the server software

Set-Cookie: new cookies that should be stored client-side

There are [many more available](#).
You can create your own headers.

Content size & streaming

When a message body is exchanged, the receiver must be able to determine when the message is complete (or how many bytes should be received to consider the body complete).

The main operating way is to use the **Content-Length** header with the size (in bytes) of the body that is to come.

When a streaming method is desired, an alternative way is to use the **Transfer-Encoding: chunked** header and to follow the [Chunked transfer encoding](#) protocol.



Notes

All specifications in this document have been simplified from the official HTTP standard. Always refer to the RFCs if necessary.

[RFC-7230: Hypertext Transfer Protocol \(HTTP/1.1\): Message Syntax and Routing](#)

Protocol versions

HTTP/0.9 and HTTP/1.0

History: [RFC-1945](#) (actual)

HTTP/1.1

This is the most used HTTP version.

History: [RFC-2068](#) (obsolete), [RFC-2616](#) (obsolete), [RFC-7230](#) (actual)

HTTP/2

According to *W3Techs*, as of March 2019, 33.9% of the top 10 million websites supported HTTP/2.

History: [RFC-7540](#) (actual)

HTTP/3

Also called *HTTP-over-QUIC*, it is the upcoming major version of HTTP.

About MIME types

The `Accept` and `Content-Type` headers use *MIME types* to specify the type of message content.

There are basic MIME types for simple files and web formats: `text/plain`, `text/html`, `application/xml`, `application/json`, `application/octet-stream`, `text/css`, `text/javascript`...

About MIME types (cont)

There are MIME types for all known file formats: `image/jpeg`, `image/png`, `audio/mpeg`, `application/pdf`, `application/zip`, `font/woff`, `video/mp4`...

There are specific MIME types related to browsers and APIs: `multipart/mixed`, `multipart/form-data`, `multipart/byteranges`, `application/x-www-form-urlencoded`...

Sometimes, extra information are added to a type. Text format can have a charset specification: `text/plain; charset=UTF-8`

See: [RFC-2045](#), [RFC-2046](#), [RFC-2047](#), [RFC-4288](#), [RFC-4289](#), [RFC-2049](#); and [MDN: MIME type](#).

Examples:

```
Content-Type: text/plain; charset=utf-8
```

Proxy

HTTP Proxy servers act as an intermediary for client-to-server requests such as HTTP.

A forward proxy is a type of proxy server that receives and forwards requests in order to cache and facilitate access to a wide range of web servers.

A reverse proxy is a type of proxy server that receives and forwards requests in order to do load-balancing for a group of web servers.

See [Proxy servers](#)

