APPENDIX H

HTTP Headers and Status Codes

THIS APPENDIX DOCUMENTS HTTP status codes and headers.

Status Codes

The HTTP server reply status line contains three fields: HTTP version, status code, and description in the following format. Status is given with a three-digit server response code. Status codes are grouped as shown in Table H-1.

Table H-1. Status Codes

Code Range	**Meaning**
100–199	Informational
200–299	Success
300–399	Redirection
400–499	Client error
500–599	Server error

1XX: Informational

This class of status code consists only of the status line and optional headers, terminated by an empty line. HTTP/1.0 hasn't defined any 1XX status codes; these are reserved for future use.

100 Continue

The client should continue with its request. This is an interim response used to inform the client that the initial part of the request has been received and hasn't yet been rejected by the server. The client should send the rest of the request or ignore this response if the request has already completed. The server sends a final response when the request is fully completed.

101 Switching Protocols

The server understands the client's request for a change in the application protocol being used on this connection and is willing to comply with it.

2XX: Success

These status codes indicate that the client's request was successfully received, understood, and accepted.

200 OK

The request has succeeded. The server's response contains the requested data.

201 Created

The request has been carried out, and a new resource has been created. The URI(s) returned in the entity of the response can be used to reference the newly created resource.

202 Accepted

The request has been accepted but not yet fully processed. The request may or may not eventually be acted upon because it might be disallowed when the processing actually takes place.

203 Partial Information

The returned information in the entity header isn't the definitive set coming from the origin server. Instead, it comes from a local or a third-party copy.

204 No Content

The server has carried out the request but doesn't need to return an entity body. Browsers shouldn't update their document view upon receiving this response. This is useful code for an image-map handler to return when the user clicks the useless or blank areas of the image.

205 Reset Content

The browser should clear the form that caused the request to be sent. This response is intended to allow the user to input actions via a form, followed by the form being cleared (rather than being resent by the server) so the user can input further actions.

206 Partial Content

The server has carried out a partial `GET` request for the resource. This is used in response to a request specifying a `Range` header. The server must specify the range included in the response with the `Content-Range` header.

3XX: Redirection

These codes indicate that the user agent needs to take further actions for the request to be successfully carried out.

300 Multiple Choices

The requested URI corresponds to any one of a set of representations; for example, the URI could refer to a document that has been translated into many languages. Agent-driven negotiation information is provided to the user agent so that the preferred representation can be selected and the user agent's request redirected to that location.

301 Moved Permanently

The requested resource has been assigned a new permanent URI, and any future references to this resource should use one of the returned URIs in the `Location` header.

302 Found

The requested resource resides temporarily under a different URI. The `Location` header points to the new location. The client should use the new URI to resolve the request, but the old URI should be used for future requests because the redirection may not be permanent.

303 See Other

The response to the request can be found at a different URI that's specified in the `Location` header and should be retrieved using a `GET` method on that resource.

304 Not Modified

The client has performed a conditional `GET` request using `If-Modified-Since` header, but the document hasn't been modified. The entity body isn't sent, and the client should use its local copy.

305 Use Proxy

The requested resource must be accessed through a proxy whose URI is given in the `Location` field.

4XX: Client Request Incomplete

The 4XX class of status code is intended for cases where the client seems to have made an error.

400 Bad Request

The request couldn't be understood by the server because of badly formed syntax.

401 Unauthorized

The result code is given along with the `WWW-Authenticate` header to indicate that the request lacked proper authorization, and the client should supply proper authentication when requesting the same URI again.

402 Payment Required

This code is reserved for future use.

403 Forbidden

The server understood the request but is refusing to fulfill it. The request shouldn't be repeated.

404 Not Found

The server hasn't found anything matching the requested URI. If the server knows that this condition is permanent, then code 410 (`Gone`) should be used instead.

405 Method Not Allowed

The method specified in the request isn't allowed for the resource identified by the requested URI.

406 Not Acceptable

The resource identified by the request can only generate response entities that have content characteristics incompatible with the accept headers sent in the request.

407 Proxy Authentication Required

This code indicates that the client must first authenticate itself with the proxy, using the `Proxy-Authenticate` header.

408 Request Timeout

The client didn't produce a request within the time that the server was prepared to wait.

409 Conflict

The request couldn't be completed because of a conflict with the current state of the resource.

410 Gone

The requested resource is no longer available at the server, and no forwarding address is known.

411 Length Required

The server is refusing to accept the request without a defined `Content-Length` from the client.

412 Precondition Failed

The precondition given in one or more `If-` request headers evaluated to `false` when it was tested on the server.

413 Request Entity Too Large

The request entity is larger than the server is willing or able to process.

414 Request-URI Too Long

The requested URI is longer than the server is willing to interpret

415 Unsupported Media Type

The entity body of the request is in a format not supported.

5XX: Server Error

These response status codes indicate cases in which the server is aware that it has made an error or can't perform the request.

500 Internal Server Error

The server encountered an unexpected condition, which prevented it from fulfilling the request.

501 Not Implemented

The server doesn't support the functionality required to fulfill the request.

502 Bad Gateway

The server, while acting as a gateway or a proxy, received an invalid response from the upstream server it accessed while trying to carry out the request.

503 Service Unavailable

The server is unable to handle the request at the present time because of a temporary overloading or maintenance of the server.

504 Gateway Timeout

The server, while acting as a gateway or proxy, didn't receive a response from the upstream server within the time it was prepared to wait.

505 HTTP Version Not Supported

The server doesn't (or refuses to) support the HTTP protocol version that was used in the request message.

HTTP Headers

HTTP headers transfer information between the client and server. There are four categories of headers, as shown in Table H-2.

Table H-2. Header Categories

Category	**Description**
General	Information that isn't related to the client, server, or HTTP
Request	Preferred document formats and server parameters
Response	Information about the server
Entity	Information on the data that's being sent between the client and server

General and Entity headers are same for both client and servers. All headers follow the `Name: value` format. Header names are case-insensitive. In HTTP/1.1, the value of headers can extend over multiple lines by preceding each extra line with at least one space or tab. All headers are terminated by a carriage-return newline sequence (`\r\n`).

General Headers

These header fields have general applicability for both request and response messages but don't apply to the entity being transferred. These header fields apply only to the message being transmitted.

`Cache-Control: Directives`

Caching directives are specified in a comma-separated list. They fall into two categories—request-based and response-based. Table H-3 describes the allowed `request` directives, and Table H-4 describes the allowed `response` directives.

Table H-3. Request Directives

Directive	**Description**
`no-cache`	Don't cache the information.
`no-store`	Remove the information from volatile storage as soon as possible after forwarding it.
`max-age=seconds`	The client is willing to accept a response whose age is no greater than the specified time in seconds.
`max-stale[=seconds]`	If `max-stale` is assigned a value, then the client is willing to accept a response that has exceeded its expiration time by no more than the specified number of seconds. The client will accept a stale response of any age if no value is assigned.
`min-fresh=seconds`	Indicates that the client is willing to accept a response that'll still be fresh for the specified time in seconds.
`only-if-cached`	This directive is used if a client wants a cache to return only those responses that it currently has stored and not to reload or revalidate with the origin server.

Table H-4. Response Directives

Directive	**Description**
`no-transform`	Caches that convert data to different formats to save space or reduce traffic shouldn't do so if they see this directive.
`cache-extension`	Cache extension tokens are interpreted by individual applications and ignored by the applications that don't understand them.
`public`	Indicates that the response may be cached by any cache.
`private`	Indicates that all or part of the response message is intended for a single user and must not be cached by a shared cache.
`must-revalidate`	A cache must not use an entry after it becomes stale to respond to a subsequent request, without first revalidating it with the origin server.
`proxy-revalidate`	The `proxy-revalidate` directive has the same meaning as the `must-revalidate`directive, except for private client caches.
`max-age=seconds`	This directive may be used by an origin server to specify the expiry time of an entity.

`Connection: options`

The header allows the sender to specify options that are to be used for a particular connection and must not be communicated by proxies over further connections. HTTP/1.1 defines the `close` connection option to allow the sender to signal that the connection will be closed after the response has been completed.

Date: date-in-rfc1123-format

Represents the date and time at which the message was originated. The field value is sent in RFC 1123 date format. An example is as follows:

```
Date: Sat, 16 Oct 1999 19:24:31 GMT
```

Pragma: no-cache

When a request message contains the `Pragma: no-cache` directive, an application should forward the request to the origin server even if it has a cached copy of what's being requested.

Trailer: header-fields

This header indicates that the given set of header fields is present in the trailer of a message encoded with chunked transfer encoding.

Transfer-Encoding: encoding-type

Transfer encoding values indicate an encoding transformation that has been, can be, or may need to be applied to an entity body in ensure "safe transport" through the network.

Upgrade: protocol/version

This header allows the client to specify to the server what additional communication protocols it supports and would like to use. If the server finds it appropriate to switch protocols, it'll use this header within a 101 (Switching Protocols) response.

Via: protocol receiver-by-host [comment]

This header must be used by gateways and proxies to indicate the intermediate protocols and recipients between both the user agent and the server on requests and the origin server and the client on responses.

Warning: warn-code warn-agent warn-text

This header carries extra information about the status or transformation of a message that might not be present in the message.

Request Headers

These header fields allow the client to pass additional information about the request, and about the client itself, to the server.

Accept: type/subtype [; q=value]

This header specifies which media types are acceptable for the response. `Accept` headers can be used to indicate that the request is limited to a small set of specific types, as in the case of a request for an inline image. The `q=value` parameter ranges from `0` to `1` (with `1` being the default) and is used to indicate a relative preference for that type, for example:

```
Accept: text/plain; q=0.5, text/html; q=0.8
```

`Accept-Charset: charset [; q=value]`

This header is used to indicate which character sets are acceptable for the response. The `q=value` parameter represents the user's preference for that particular character set.

`Accept-Encoding: encoding-types [; q=value]`

This header restricts the content transfer encodings that are acceptable in the response. The `q=value` parameter allows the user to express a preference for a particular type of encoding.

`Accept-Language: language [; q=value]`

This header restricts the set of natural languages that are preferred as a response to the request. Each language may be given an associated preference with the `q=value` parameter.

`Authorization: credentials`

This provides the client's authorization to access the URI. When a requested URI requires authorization, the server responds with a `WWW-Authenticate` header describing the type of authorization required. The client then repeats the request with proper authorization information.

`Expect: 100-continue | expectation`

This header indicates that particular server behaviors are required by the client. A server that can't understand or comply with any of the expectation values in the `Expect` field of a request will respond with an appropriate error status.

`From: email`

This header contains an Internet e-mail address for the human controlling the requesting user agent.

`Host: host [: port]`

This header specifies the Internet host and port number of the resource being requested.

`If-Match:`

A client that has previously obtained one or more entities from the resource can include a list of their associated entity tags in this header field to verify that one of those entities is current.

If-Modified-Since: datein-rfc1123-format

This header specifies that the URI data should be sent only if it has been modified since the date given.

If-None-Match: entity-tags

This header is similar to the `If-Match` header but is used to verify that none of those entities previously obtained by the client is current.

If-Range: entity-tag | date

If a client has a partial copy of an entity in its cache, it can use this header to retrieve the rest of the entity if it's unmodified or the whole entity if it has changed.

If-Unmodified-Since: date-in-rfc1123-format

This specifies that the URI data should only be sent if it hasn't been modified since the given date.

Max-Forwards: number

This header limits the number of proxies and gateways that can forward the request.

Proxy-Authorization: credentials

The `Proxy-Authentication` request header allows the client to identify itself (or its user) to a proxy that requires authentication.

Range: bytes=n-m

Using this header with a conditional or unconditional `GET` allows the retrieval of one or more subranges of an entity, rather than the entire entity.

Referer: url

The `Referer` request header allows the client to specify the URI of the resource from which the requested URL was obtained.

TE: transfer-encoding [; q = val]

The `TE` request header indicates which extension transfer encodings the client is willing to accept in a response, over and above the encoding defined by `Transfer-Encoding`. If the keyword "trailers" is present, then the client is willing to accept trailing fields in the final chunk of a `chunked` transfer encoding.

User-Agent: product | comment

This header contains information about the user agent (a.k.a. the client) originating the request. This allows the server to automatically recognize user agents and tailor its responses to avoid particular user-agent limitations.

Response Headers

Response headers carry additional meta information about the response that can't be placed in the status line. These headers give information about the server and about further access to the resource identified by the requested URI.

Accept-Ranges: range-unit | none

This header allows the server to indicate its acceptance of range requests for a resource.

Age: seconds

This header contains the sender's estimate of the amount of time since the response was generated at the origin server.

Etag: entity-tag

This header provides the current value of the requested entity tag. See the `FileETag` directive for details.

Location: URI

This is used to redirect the recipient to a location other than the `Request-URI` to complete the request.

Proxy-Authenticate: scheme realm

This header indicates the authentication scheme and parameters applicable to the proxy for this `Request-URI`.

Retry-After: date | seconds

This is used by the server to indicate how long the service is expected to be unavailable to the requesting client.

Server string

The `Server` header contains information about the software that the origin server used to handle the request.

*Vary: * | headers*

This header specifies that the entity has multiple sources and may therefore vary according to a specified list of request headers. Multiple headers can be listed separated by commas. An asterisk means another factor other than the request headers may affect the response that is returned.

WWW-Authenticate: scheme realm

This header is used with the 401 response code to indicate to the client that the requested URI needs authentication. The value specifies the authorization scheme and the realm of authority required from the client.

Entity Headers

Entity headers, also called *content headers*, define meta information about the body of an HTTP request or response (the entity body) or about the resource identified by a request.

Allow: methods

This header informs the recipient of valid methods associated with the resource.

Content-Encoding: encoding

This header indicates what additional content codings have been applied to the entity body and hence what decoding must be carried out to obtain the media-type referenced by the `Content-Type` header.

Content-Language: languages

The `Content-Language` header describes the natural language(s) of the intended audience for the enclosed entity.

Content-Length: n

This header indicates the size of the entity body. Because of the dynamic nature of some requests, the content length is sometimes unknown, and this header is omitted (for example, with `chunked` transfer encoding).

The `Content-Language` header supplies the resource location for the entity enclosed in the message when that entity may be accessed from a different location to the requested resource's URI.

Content-MD5: digest

This header contains an MD5 digest of the entity body that's used to provide an end-to-end Message Integrity Check (MIC) of the entity body, as implemented by `mod_auth_digest`. See RFC 1864 for more details.

Content-Range: bytes n-m/length

The `Content-Range` header is sent with a partial entity body to specify where in the full entity body the partial body should come from.

Content-Type: type/subtype

This header describes the media type of the entity body sent to the recipient. In the case of the `HEAD` method, it describes the media type that would have been sent had the request been a `GET`.

Expires: RFC-1123-date

The `Expires` header gives the date and time after which the response is considered stale.

Last-Modified: RFC-1123-date

This header indicates the date and time at which the origin server believes the variant was last modified.