**HTTP Error Codes:**

100 - Continue

101 -Switching Protocols

102- Processing

200- OK

201 -Created

202- Accepted

203- Non-Authoritative Information

204- No Content

205- Reset Content

206- Partial Content

207- Multi-Status

208- Already Reported

226- IM Used

300- Multiple Choices

301- Moved Permanently

302- Found

303- See Other

304- Not Modified

305- Use Proxy

307- Temporary Redirect

308- Permanent Redirect

400- Bad Request

401- Unauthorized

402- Payment Required

403- Forbidden

404- Not Found

405- Method Not Allowed

406- Not Acceptable

407- Proxy Authentication Required

408- Request Timeout

409- Conflict

410- Gone

411- Length Required

412- Precondition Failed

413- Payload Too Large

414- Request-URI Too Long

415- Unsupported Media Type

416- Requested Range Not Satisfiable

417- Expectation Failed

418- I’m A Teapot

421- Misdirected Request

422- Unprocessable Entity

423- Locked

424- Failed Dependency

426- Upgrade Required

428- Precondition Required

429- Too Many Requests

431- Request Header Fields Too Large

444- Connection Closed Without Response

451- Unavailable For Legal Reasons

499- Client Closed Request

500- Internal Server Error

501- Not Implemented

502- Bad Gateway

503- Service Unavailable

504- Gateway Timeout

505- HTTP Version Not Supported

506- Variant Also Negotiates

507- Insufficient Storage

508- Loop Detected

510- Not Extended

511- Network Authentication Required

599- Network Connect Timeout Error

**What Is a 100 Status Code?**

The initial part of a request has been received and has not yet been rejected by the server. The server intends to send a final response after the request has been fully received and acted upon.

When the request contains an Expect header field that includes a 100-continue expectation, the 100 response indicates that the server wishes to receive the request payload body1. The client ought to continue sending the request and discard the 100 response.

If the request did not contain an Expect header field containing the 100-continue expectation, the client can simply discard this interim response.

**What Is a 101 Status Code?**

The server understands and is willing to comply with the client’s request, via the Upgrade header field 1, for a change in the application protocol being used on this connection.

The server MUST generate an Upgrade header field in the response that indicates which protocol(s) will be switched to immediately after the empty line that terminates the 101 response.

It is assumed that the server will only agree to switch protocols when it is advantageous to do so. For example, switching to a newer version of HTTP might be advantageous over older versions, and switching to a real-time, synchronous protocol might be advantageous when delivering resources that use such features.

**What Is a 102 Status Code?**

An interim response used to inform the client that the server has accepted the complete request but has not yet completed it.

This status code SHOULD only be sent when the server has a reasonable expectation that the request will take significant time to complete. As guidance, if a method is taking longer than 20 seconds (a reasonable, but arbitrary value) to process the server SHOULD return a 102 (Processing) response. The server MUST send a final response after the request has been completed.

Methods can potentially take a long period of time to process, especially methods that support the Depth header. In such cases the client may time-out the connection while waiting for a response. To prevent this the server may return a 102 Processing status code to indicate to the client that the server is still processing the method.

**What Is a 200 Status Code?**

The request has succeeded.

The payload sent in a 200 response depends on the request method. For the methods defined by this specification, the intended meaning of the payload can be summarized as:

GET a representation of the target resource

HEAD the same representation as GET, but without the representation data

POST a representation of the status of, or results obtained from, the action;

PUT DELETE a representation of the status of the action;

OPTIONS a representation of the communications options;

TRACE a representation of the request message as received by the end server.

Aside from responses to CONNECT, a 200 response always has a payload, though an origin server MAY generate a payload body of zero length. If no payload is desired, an origin server ought to send 204 No Content instead. For CONNECT, no payload is allowed because the successful result is a tunnel, which begins immediately after the 200 response header section.

A 200 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

What Is a 201 Status Code?

The request has been fulfilled and has resulted in one or more new resources being created.

The primary resource created by the request is identified by either a Location header field in the response or, if no Location field is received, by the effective request URI.

The 201 response payload typically describes and links to the resource(s) created.

**What Is a 202 Status Code?**

The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.

There is no facility in HTTP for re-sending a status code from an asynchronous operation.

The 202 response is intentionally noncommittal. Its purpose is to allow a server to accept a request for some other process (perhaps a batch-oriented process that is only run once per day) without requiring that the user agent’s connection to the server persist until the process is completed. The representation sent with this response ought to describe the request’s current status and point to (or embed) a status monitor that can provide the user with an estimate of when the request will be fulfilled.

**What Is a 203 Status Code?**

The request was successful but the enclosed payload has been modified from that of the origin server’s 200 OK response by a transforming proxy1.

This status code allows the proxy to notify recipients when a transformation has been applied, since that knowledge might impact later decisions regarding the content. For example, future cache validation requests for the content might only be applicable along the same request path (through the same proxies).

The 203 response is similar to the Warning code of 214 Transformation Applied2, which has the advantage of being applicable to responses with any status code.

A 203 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 204 Status Code?**

The server has successfully fulfilled the request and that there is no additional content to send in the response payload body.

Metadata in the response header fields refer to the target resource and its selected representation after the requested action was applied.

For example, if a 204 status code is received in response to a PUT request and the response contains an ETag header field, then the PUT was successful and the ETag field-value contains the entity-tag for the new representation of that target resource.

The 204 response allows a server to indicate that the action has been successfully applied to the target resource, while implying that the user agent does not need to traverse away from its current “document view” (if any). The server assumes that the user agent will provide some indication of the success to its user, in accord with its own interface, and apply any new or updated metadata in the response to its active representation.

For example, a 204 status code is commonly used with document editing interfaces corresponding to a “save” action, such that the document being saved remains available to the user for editing. It is also frequently used with interfaces that expect automated data transfers to be prevalent, such as within distributed version control systems.

A 204 response is terminated by the first empty line after the header fields because it cannot contain a message body.

A 204 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 205 Status Code?**

The server has fulfilled the request and desires that the user agent reset the “document view”, which caused the request to be sent, to its original state as received from the origin server.

This response is intended to support a common data entry use case where the user receives content that supports data entry (a form, notepad, canvas, etc.), enters or manipulates data in that space, causes the entered data to be submitted in a request, and then the data entry mechanism is reset for the next entry so that the user can easily initiate another input action.

Since the 205 status code implies that no additional content will be provided, a server MUST NOT generate a payload in a 205 response. In other words, a server MUST do one of the following for a 205 response: a) indicate a zero-length body for the response by including a Content-Length header field with a value of 0; b) indicate a zero-length payload for the response by including a Transfer-Encoding header field with a value of chunked and a message body consisting of a single chunk of zero-length; or, c) close the connection immediately after sending the blank line terminating the header section.

**What Is a 206 Status Code?**

The server is successfully fulfilling a range request for the target resource by transferring one or more parts of the selected representation that correspond to the satisfiable ranges found in the request’s Range header field1.

If a single part is being transferred, the server generating the 206 response MUST generate a Content-Range header field, describing what range of the selected representation is enclosed, and a payload consisting of the range. For example:

HTTP/1.1 206 Partial Content

Date: Wed, 15 Nov 1995 06:25:24 GMT

Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT

Content-Range: bytes 21010-47021/47022

Content-Length: 26012

Content-Type: image/gif

... 26012 bytes of partial image data ...

If multiple parts are being transferred, the server generating the 206 response MUST generate a “multipart/byteranges” payload2, and a Content-Type header field containing the multipart/byteranges media type and its required boundary parameter. To avoid confusion with single-part responses, a server MUST NOT generate a Content-Range header field in the HTTP header section of a multiple part response (this field will be sent in each part instead).

Within the header area of each body part in the multipart payload, the server MUST generate a Content-Range header field corresponding to the range being enclosed in that body part. If the selected representation would have had a Content-Type header field in a 200 OK response, the server SHOULD generate that same Content-Type field in the header area of each body part. For example:

HTTP/1.1 206 Partial Content

Date: Wed, 15 Nov 1995 06:25:24 GMT

Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT

Content-Length: 1741

Content-Type: multipart/byteranges; boundary=THIS\_STRING\_SEPARATES

--THIS\_STRING\_SEPARATES

Content-Type: application/pdf

Content-Range: bytes 500-999/8000

...the first range...

--THIS\_STRING\_SEPARATES

Content-Type: application/pdf

Content-Range: bytes 7000-7999/8000

...the second range

--THIS\_STRING\_SEPARATES--

When multiple ranges are requested, a server MAY coalesce any of the ranges that overlap, or that are separated by a gap that is smaller than the overhead of sending multiple parts, regardless of the order in which the corresponding byte-range-spec appeared in the received Range header field. Since the typical overhead between parts of a multipart/byteranges payload is around 80 bytes, depending on the selected representation’s media type and the chosen boundary parameter length, it can be less efficient to transfer many small disjoint parts than it is to transfer the entire selected representation.

A server MUST NOT generate a multipart response to a request for a single range, since a client that does not request multiple parts might not support multipart responses. However, a server MAY generate a multipart/byteranges payload with only a single body part if multiple ranges were requested and only one range was found to be satisfiable or only one range remained after coalescing. A client that cannot process a multipart/byteranges response MUST NOT generate a request that asks for multiple ranges.

When a multipart response payload is generated, the server SHOULD send the parts in the same order that the corresponding byte-range-spec appeared in the received Range header field, excluding those ranges that were deemed unsatisfiable or that were coalesced into other ranges. A client that receives a multipart response MUST inspect the Content-Range header field present in each body part in order to determine which range is contained in that body part; a client cannot rely on receiving the same ranges that it requested, nor the same order that it requested.

When a 206 response is generated, the server MUST generate the following header fields, in addition to those required above, if the field would have been sent in a 200 OK response to the same request: Date, Cache-Control, ETag, Expires, Content-Location, and Vary.

If a 206 is generated in response to a request with an If-Range header field, the sender SHOULD NOT generate other representation header fields beyond those required above, because the client is understood to already have a prior response containing those header fields. Otherwise, the sender MUST generate all of the representation header fields that would have been sent in a 200 OK response to the same request.

A 206 response is cacheable by default; i.e., unless otherwise indicated by explicit cache controls.

**What Is a 207 Status Code?**

A Multi-Status response conveys information about multiple resources in situations where multiple status codes might be appropriate.

The default Multi-Status response body is a text/xml or application/xml HTTP entity with a ‘multistatus’ root element. Further elements contain 200, 300, 400, and 500 series status codes generated during the method invocation. 100 series status codes SHOULD NOT be recorded in a ‘response’ XML element.

Although ‘207’ is used as the overall response status code, the recipient needs to consult the contents of the multistatus response body for further information about the success or failure of the method execution. The response MAY be used in success, partial success and also in failure situations.

The ‘multistatus’ root element holds zero or more ‘response’ elements in any order, each with information about an individual resource. Each ‘response’ element MUST have an ‘href’ element to identify the resource.

A Multi-Status response uses one out of two distinct formats for representing the status:

1. A ‘status’ element as child of the ‘response’ element indicates the status of the message execution for the identified resource as a whole1. Some method definitions provide information about specific status codes clients should be prepared to see in a response.

2. For PROPFIND and PROPPATCH, the format has been extended using the ‘propstat’ element instead of ‘status’, providing information about individual properties of a resource.

**What Is a 208 Status Code?**

Used inside a DAV: propstat response element to avoid enumerating the internal members of multiple bindings to the same collection repeatedly.

For each binding to a collection inside the request’s scope, only one will be reported with a 200 status, while subsequent DAV:response elements for all other bindings will use the 208 status, and no DAV:response elements for their descendants are included.

Note that the 208 status will only occur for “Depth: infinity” requests, and that it is of particular importance when the multiple collection bindings cause a bind loop1.

A client can request the DAV:resource-id property in a PROPFIND request to guarantee that they can accurately reconstruct the binding structure of a collection with multiple bindings to a single resource.

For backward compatibility with clients not aware of the 208 status code appearing in multistatus response bodies, it SHOULD NOT be used unless the client has signaled support for this specification using the “DAV” request headerd2. Instead, a 508 Loop Detected status should be returned when a binding loop is discovered. This allows the server to return the 508 as the top-level return status, if it discovers it before it started the response, or in the middle of a multistatus, if it discovers it in the middle of streaming out a multistatus response.

**What Is a 226 Status Code?**

The server has fulfilled a GET request for the resource, and the response is a representation of the result of one or more instance-manipulations applied to the current instance.

The actual current instance might not be available except by combining this response with other previous or future responses, as appropriate for the specific instance-manipulation(s). If so, the headers of the resulting instance are the result of combining the headers from the 226 response and the other instances, following the rules in section 13.5.3 of the HTTP/1.1 specification.

The request MUST have included an A-IM header field listing at least one instance-manipulation. The response MUST include an Etag header field giving the entity tag of the current instance.

A response received with a status code of 226 MAY be stored by a cache and used in reply to a subsequent request, subject to the HTTP expiration mechanism and any Cache-Control headers, and to the requirements in section 10.6.

A response received with a status code of 226 MAY be used by a cache, in conjunction with a cache entry for the base instance, to create a cache entry for the current instance.

**What Is a 300 Status Code?**

The target resource has more than one representation, each with its own more specific identifier, and information about the alternatives is being provided so that the user (or user agent) can select a preferred representation by redirecting its request to one or more of those identifiers.

In other words, the server desires that the user agent engage in reactive negotiation to select the most appropriate representation(s) for its needs1.

If the server has a preferred choice, the server SHOULD generate a Location header field containing a preferred choice’s URI reference. The user agent MAY use the Location field value for automatic redirection.

For request methods other than HEAD, the server SHOULD generate a payload in the 300 response containing a list of representation metadata and URI reference(s) from which the user or user agent can choose the one most preferred. The user agent MAY make a selection from that list automatically if it understands the provided media type. A specific format for automatic selection is not defined by this specification because HTTP tries to remain orthogonal to the definition of its payloads. In practice, the representation is provided in some easily parsed format believed to be acceptable to the user agent, as determined by shared design or content negotiation, or in some commonly accepted hypertext format.

A 300 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

Note: The original proposal for the 300 status code defined the URI header field as providing a list of alternative representations, such that it would be usable for 200, 300, and 406 responses and be transferred in responses to the HEAD method. However, lack of deployment and disagreement over syntax led to both URI and Alternates (a subsequent proposal) being dropped from this specification. It is possible to communicate the list using a set of Link header fields3, each with a relationship of “alternate”, though deployment is a chicken-and-egg problem.

**What Is a 301 Status Code?**

The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.

Clients with link-editing capabilities ought to automatically re-link references to the effective request URI to one or more of the new references sent by the server, where possible.

The server SHOULD generate a Location header field in the response containing a preferred URI reference for the new permanent URI. The user agent MAY use the Location field value for automatic redirection. The server’s response payload usually contains a short hypertext note with a hyperlink to the new URI(s).

Note: For historical reasons, a user agent MAY change the request method from POST to GET for the subsequent request. If this behavior is undesired, the 307 Temporary Redirect status code can be used instead.

A 301 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 302 Status Code?**

The target resource resides temporarily under a different URI. Since the redirection might be altered on occasion, the client ought to continue to use the effective request URI for future requests.

The server SHOULD generate a Location header field in the response containing a URI reference for the different URI. The user agent MAY use the Location field value for automatic redirection. The server’s response payload usually contains a short hypertext note with a hyperlink to the different URI(s).

Note: For historical reasons, a user agent MAY change the request method from POST to GET for the subsequent request. If this behavior is undesired, the 307 Temporary Redirect status code can be used instead.

**What Is a 303 Status Code?**

The server is redirecting the user agent to a different resource, as indicated by a URI in the Location header field, which is intended to provide an indirect response to the original request.

A user agent can perform a retrieval request targeting that URI (a GET or HEAD request if using HTTP), which might also be redirected, and present the eventual result as an answer to the original request. Note that the new URI in the Location header field is not considered equivalent to the effective request URI.

This status code is applicable to any HTTP method. It is primarily used to allow the output of a POST action to redirect the user agent to a selected resource, since doing so provides the information corresponding to the POST response in a form that can be separately identified, bookmarked, and cached, independent of the original request.

A 303 response to a GET request indicates that the origin server does not have a representation of the target resource that can be transferred by the server over HTTP. However, the Location field value refers to a resource that is descriptive of the target resource, such that making a retrieval request on that other resource might result in a representation that is useful to recipients without implying that it represents the original target resource. Note that answers to the questions of what can be represented, what representations are adequate, and what might be a useful description are outside the scope of HTTP.

Except for responses to a HEAD request, the representation of a 303 response ought to contain a short hypertext note with a hyperlink to the same URI reference provided in the Location header field.

**What Is a 304 Status Code?**

A conditional GET or HEAD request has been received and would have resulted in a 200 OK response if it were not for the fact that the condition evaluated to false.

In other words, there is no need for the server to transfer a representation of the target resource because the request indicates that the client, which made the request conditional, already has a valid representation; the server is therefore redirecting the client to make use of that stored representation as if it were the payload of a 200 OK response.

The server generating a 304 response MUST generate any of the following header fields that would have been sent in a 200 OK response to the same request: Cache-Control, Content-Location, Date, ETag, Expires, and Vary.

Since the goal of a 304 response is to minimize information transfer when the recipient already has one or more cached representations, a sender SHOULD NOT generate representation metadata other than the above listed fields unless said metadata exists for the purpose of guiding cache updates (e.g., Last-Modified might be useful if the response does not have an ETag field).

If the conditional request originated with an outbound client, such as a user agent with its own cache sending a conditional GET to a shared proxy, then the proxy SHOULD forward the 304 response to that client.

A 304 response cannot contain a message-body; it is always terminated by the first empty line after the header fields.

**What Is a 305 Status Code?**

Defined in a previous version of this specification and is now deprecated, due to security concerns regarding in-band configuration of a proxy.

What Is a 307 Status Code?

The target resource resides temporarily under a different URI and the user agent MUST NOT change the request method if it performs an automatic redirection to that URI.

Since the redirection can change over time, the client ought to continue using the original effective request URI for future requests.

The server SHOULD generate a Location header field in the response containing a URI reference for the different URI. The user agent MAY use the Location field value for automatic redirection. The server’s response payload usually contains a short hypertext note with a hyperlink to the different URI(s).

Note: This status code is similar to 302 Found, except that it does not allow changing the request method from POST to GET. This specification defines no equivalent counterpart for 301 Moved Permanently (RFC7238, however, proposes defining the status code 308 Permanent Redirect for this purpose).

**What Is a 308 Status Code?**

The target resource has been assigned a new permanent URI and any future references to this resource ought to use one of the enclosed URIs.

Clients with link editing capabilities ought to automatically re-link references to the effective request URI1 to one or more of the new references sent by the server, where possible.

The server SHOULD generate a Location header field2 in the response containing a preferred URI reference for the new permanent URI. The user agent MAY use the Location field value for automatic redirection. The server’s response payload usually contains a short hypertext note with a hyperlink to the new URI(s).

A 308 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls3.

Note: This status code is similar to 301 Moved Permanently, except that it does not allow changing the request method from POST to GET.

**What Is a 400 Status Code?**

The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

**What Is a 401 Status Code?**

The request has not been applied because it lacks valid authentication credentials for the target resource.

The server generating a 401 response MUST send a WWW-Authenticate header field1 containing at least one challenge applicable to the target resource.

If the request included authentication credentials, then the 401 response indicates that authorization has been refused for those credentials. The user agent MAY repeat the request with a new or replaced Authorization header field2. If the 401 response contains the same challenge as the prior response, and the user agent has already attempted authentication at least once, then the user agent SHOULD present the enclosed representation to the user, since it usually contains relevant diagnostic information.

**What Is a 402 Status Code?**

Reserved for future use.

**What Is a 403 Status Code?**

The server understood the request but refuses to authorize it.

A server that wishes to make public why the request has been forbidden can describe that reason in the response payload (if any).

If authentication credentials were provided in the request, the server considers them insufficient to grant access. The client SHOULD NOT automatically repeat the request with the same credentials. The client MAY repeat the request with new or different credentials. However, a request might be forbidden for reasons unrelated to the credentials.

An origin server that wishes to “hide” the current existence of a forbidden target resource MAY instead respond with a status code of 404 Not Found.

**What Is a 404 Status Code?**

The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

A 404 status code does not indicate whether this lack of representation is temporary or permanent; the 410 Gone status code is preferred over 404 if the origin server knows, presumably through some configurable means, that the condition is likely to be permanent.

A 404 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 405 Status Code?**

The method received in the request-line is known by the origin server but not supported by the target resource.

The origin server MUST generate an Allow header field in a 405 response containing a list of the target resource’s currently supported methods.

A 405 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 406 Status Code?**

The target resource does not have a current representation that would be acceptable to the user agent, according to the proactive negotiation header fields received in the request1, and the server is unwilling to supply a default representation.

The server SHOULD generate a payload containing a list of available representation characteristics and corresponding resource identifiers from which the user or user agent can choose the one most appropriate. A user agent MAY automatically select the most appropriate choice from that list. However, this specification does not define any standard for such automatic selection.

**What Is a 407 Status Code?**

Similar to 401 Unauthorized, but it indicates that the client needs to authenticate itself in order to use a proxy.

The proxy MUST send a Proxy-Authenticate header field1 containing a challenge applicable to that proxy for the target resource. The client MAY repeat the request with a new or replaced Proxy-Authorization header field.

**What Is a 408 Status Code?**

The server did not receive a complete request message within the time that it was prepared to wait.

A server SHOULD send the “close” connection option1 in the response, since 408 implies that the server has decided to close the connection rather than continue waiting. If the client has an outstanding request in transit, the client MAY repeat that request on a new connection.

**What Is a 409 Status Code?**

The request could not be completed due to a conflict with the current state of the target resource. This code is used in situations where the user might be able to resolve the conflict and resubmit the request.

The server SHOULD generate a payload that includes enough information for a user to recognize the source of the conflict.

Conflicts are most likely to occur in response to a PUT request. For example, if versioning were being used and the representation being PUT included changes to a resource that conflict with those made by an earlier (third-party) request, the origin server might use a 409 response to indicate that it can’t complete the request. In this case, the response representation would likely contain information useful for merging the differences based on the revision history.

**What Is a 410 Status Code?**

The target resource is no longer available at the origin server and that this condition is likely to be permanent.

If the origin server does not know, or has no facility to determine, whether or not the condition is permanent, the status code 404 Not Found ought to be used instead.

The 410 response is primarily intended to assist the task of web maintenance by notifying the recipient that the resource is intentionally unavailable and that the server owners desire that remote links to that resource be removed. Such an event is common for limited-time, promotional services and for resources belonging to individuals no longer associated with the origin server’s site. It is not necessary to mark all permanently unavailable resources as “gone” or to keep the mark for any length of time — that is left to the discretion of the server owner.

A 410 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 411 Status Code?**

The server refuses to accept the request without a defined Content-Length1.

The client MAY repeat the request if it adds a valid Content-Length header field containing the length of the message body in the request message.

**What Is a 412 Status Code?**

One or more conditions given in the request header fields evaluated to false when tested on the server.

This response code allows the client to place preconditions on the current resource state (its current representations and metadata) and, thus, prevent the request method from being applied if the target resource is in an unexpected state.

**What Is a 413 Status Code?**

The server is refusing to process a request because the request payload is larger than the server is willing or able to process.

The server MAY close the connection to prevent the client from continuing the request.

If the condition is temporary, the server SHOULD generate a Retry-After header field to indicate that it is temporary and after what time the client MAY try again.

**What Is a 414 Status Code?**

The server is refusing to service the request because the request-target1 is longer than the server is willing to interpret.

This rare condition is only likely to occur when a client has improperly converted a POST request to a GET request with long query information, when the client has descended into a “black hole” of redirection (e.g., a redirected URI prefix that points to a suffix of itself) or when the server is under attack by a client attempting to exploit potential security holes.

A 414 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 415 Status Code?**

The origin server is refusing to service the request because the payload is in a format not supported by this method on the target resource.

The format problem might be due to the request’s indicated Content-Type or Content-Encoding, or as a result of inspecting the data directly.

**What Is a 416 Status Code?**

None of the ranges in the request’s Range header field1 overlap the current extent of the selected resource or that the set of ranges requested has been rejected due to invalid ranges or an excessive request of small or overlapping ranges.

For byte ranges, failing to overlap the current extent means that the first-byte-pos of all of the byte-range-spec values were greater than the current length of the selected representation. When this status code is generated in response to a byte-range request, the sender SHOULD generate a Content-Range header field specifying the current length of the selected representation2.

For example:

HTTP/1.1 416 Range Not Satisfiable

Date: Fri, 20 Jan 2012 15:41:54 GMT

Content-Range: bytes \*/47022

Note: Because servers are free to ignore Range, many implementations will simply respond with the entire selected representation in a 200 OK response. That is partly because most clients are prepared to receive a 200 OK to complete the task (albeit less efficiently) and partly because clients might not stop making an invalid partial request until they have received a complete representation. Thus, clients cannot depend on receiving a 416 Range Not Satisfiable response even when it is most appropriate.

**What Is a 417 Status Code?**

The expectation given in the request’s Expect header field1 could not be met by at least one of the inbound servers.

**What Is a 418 Status Code?**

Any attempt to brew coffee with a teapot should result in the error code “418 I’m a teapot”. The resulting entity body MAY be short and stout.

This status code is a reference to the Hyper Text Coffee Pot Control Protocol, which was released in 1998 as an April Fools’ joke.

Although this status code started as a joke, websites can technically use it as a response. Some sites use 418s for requests they don’t want to handle, such as automated queries. Use of 418 is rare and is not usually considered best practice.

Reading through the Hyper Text Coffee Pot Control Protocol can be entertaining, however.

**What Is a 421 Status Code?**

The request was directed at a server that is not able to produce a response. This can be sent by a server that is not configured to produce responses for the combination of scheme and authority that are included in the request URI.

Clients receiving a 421 Misdirected Request response from a server MAY retry the request — whether the request method is idempotent or not — over a different connection. This is possible if a connection is reused1 or if an alternative service is selected ALT-SVC.

This status code MUST NOT be generated by proxies.

A 421 response is cacheable by default, i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 422 Status Code?**

The server understands the content type of the request entity (hence a 415 Unsupported Media Type status code is inappropriate), and the syntax of the request entity is correct (thus a 400 Bad Request status code is inappropriate) but was unable to process the contained instructions.

For example, this error condition may occur if an XML request body contains well-formed (i.e., syntactically correct), but semantically erroneous, XML instructions.

**What Is a 423 Status Code?**

The source or destination resource of a method is locked.

This response SHOULD contain an appropriate precondition or postcondition code, such as ‘lock-token-submitted’ or ‘no-conflicting-lock’.

**What Is a 424 Status Code?**

The method could not be performed on the resource because the requested action depended on another action and that action failed.

For example, if a command in a PROPPATCH method fails, then, at minimum, the rest of the commands will also fail with 424 Failed Dependency.

**What Is a 426 Status Code?**

The server refuses to perform the request using the current protocol but might be willing to do so after the client upgrades to a different protocol.

The server MUST send an Upgrade header field in a 426 response to indicate the required protocol(s).

**What Is a 428 Status Code?**

The origin server requires the request to be conditional.

Its typical use is to avoid the “lost update” problem, where a client GETs a resource’s state, modifies it, and PUTs it back to the server, when meanwhile a third party has modified the state on the server, leading to a conflict. By requiring requests to be conditional, the server can assure that clients are working with the correct copies.

Responses using this status code SHOULD explain how to resubmit the request successfully.

**What Is a 429 Status Code?**

The user has sent too many requests in a given amount of time (“rate limiting”).

The response representations SHOULD include details explaining the condition, and MAY include a Retry-After header indicating how long to wait before making a new request.

**What Is a 431 Status Code?**

The server is unwilling to process the request because its header fields are too large. The request MAY be resubmitted after reducing the size of the request header fields.

It can be used both when the set of request header fields in total is too large, and when a single header field is at fault. In the latter case, the response representation SHOULD specify which header field was too large.

**What Is a 444 Status Code?**

A non-standard status code used to instruct nginx to close the connection without sending a response to the client, most commonly used to deny malicious or malformed requests.

This status code is not seen by the client, it only appears in nginx log files.

**What Is a 451 Status Code?**

The server is denying access to the resource as a consequence of a legal demand.

The server in question might not be an origin server. This type of legal demand typically most directly affects the operations of ISPs and search engines.

Responses using this status code SHOULD include an explanation, in the response body, of the details of the legal demand: the party making it, the applicable legislation or regulation, and what classes of person and resource it applies to.

**What Is a 499 Status Code?**

A non-standard status code introduced by nginx for the case when a client closes the connection while nginx is processing the request.

**What Is a 500 Status Code?**

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

**What Is a 501 Status Code?**

The server does not support the functionality required to fulfill the request.

This is the appropriate response when the server does not recognize the request method and is not capable of supporting it for any resource.

A 501 response is cacheable by default; i.e., unless otherwise indicated by the method definition or explicit cache controls.

**What Is a 502 Status Code?**

The server, while acting as a gateway or proxy, received an invalid response from an inbound server it accessed while attempting to fulfill the request.

**What Is a 503 Status Code?**

The server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay.

The server MAY send a Retry-After header field1 to suggest an appropriate amount of time for the client to wait before retrying the request.

Note: The existence of the 503 status code does not imply that a server has to use it when becoming overloaded. Some servers might simply refuse the connection.

**What Is a 504 Status Code?**

The server, while acting as a gateway or proxy, did not receive a timely response from an upstream server it needed to access in order to complete the request.

**What Is a 505 Status Code**?

The server does not support, or refuses to support, the major version of HTTP that was used in the request message.

The server is indicating that it is unable or unwilling to complete the request using the same major version as the client. The server SHOULD generate a representation for the 505 response that describes why that version is not supported and what other protocols are supported by that server.

**What Is a 506 Status Code?**

The server has an internal configuration error: the chosen variant resource is configured to engage in transparent content negotiation itself, and is therefore not a proper end point in the negotiation process.

**What Is a 507 Status Code?**

The method could not be performed on the resource because the server is unable to store the representation needed to successfully complete the request.

This condition is considered to be temporary. If the request that received this status code was the result of a user action, the request MUST NOT be repeated until it is requested by a separate user action.

**What Is a 508 Status Code?**

The server terminated an operation because it encountered an infinite loop while processing a request with “Depth: infinity”. This status indicates that the entire operation failed.

**What Is a 510 Status Code?**

The policy for accessing the resource has not been met in the request. The server should send back all the information necessary for the client to issue an extended request.

It is outside the scope of this specification to specify how the extensions inform the client.

If the 510 response contains information about extensions that were not present in the initial request then the client MAY repeat the request if it has reason to believe it can fulfill the extension policy by modifying the request according to the information provided in the 510 response. Otherwise the client MAY present any entity included in the 510 response to the user, since that entity may include relevant diagnostic information.

**What Is a 511 Status Code?**

The client needs to authenticate to gain network access.

The response representation SHOULD contain a link to a resource that allows the user to submit credentials (e.g., with an HTML form).

Note that the 511 response SHOULD NOT contain a challenge or the login interface itself, because browsers would show the login interface as being associated with the originally requested URL, which may cause confusion.

The 511 status SHOULD NOT be generated by origin servers; it is intended for use by intercepting proxies that are interposed as a means of controlling access to the network.

Responses with the 511 status code MUST NOT be stored by a cache.

The 511 status code is designed to mitigate problems caused by “captive portals” to software (especially non-browser agents) that is expecting a response from the server that a request was made to, not the intervening network infrastructure. It is not intended to encourage deployment of captive portals — only to limit the damage caused by them.

A network operator wishing to require some authentication, acceptance of terms, or other user interaction before granting access usually does so by identifying clients who have not done so (“unknown clients”) using their Media Access Control (MAC) addresses.

Unknown clients then have all traffic blocked, except for that on TCP port 80, which is sent to an HTTP server (the “login server”) dedicated to “logging in” unknown clients, and of course traffic to the login server itself.

**What Is a 599 Status Code?**

This status code is not specified in any RFCs but is used by some HTTP proxies to signal a network connect timeout behind the proxy to a client in front of the proxy.