**HTTP FEATURES AND FUNCTIONALITY**

> CACHE CONTROL - used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain.

\* HTTP/1.0 caches might not implement Cache-Control and might only implement Pragma: no-cache

> CONTENT MEDIA TYPE (MIME) SPECIFICATION - HTTP uses Internet Media Types [17] in the Content-Type and Accept header fields in order to provide open and extensible data typing and type negotiation.

\*MIME - Multipurpose Internet Mail Extensions

> LANGUAGE AND CHARSET SPECIFICATION - refer to a method used with one or more tables to convert a sequence of octets into a sequence of characters

> CONTENT/ TRANSFER CODING

> Content Coding - used to allow a document to be compressed or otherwise usefully transformed without losing the identity of its underlying media type and without loss of information

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

- property of the message, not of the original entity.

> CONTENT NEGOTIATION - the process of selecting the best representation for a given response when there are multiple representations available.

- must agree on the client/server agreement first; Negotiation before sending

Two kinds of content negotiation: server-driven and agent-driven negotiation.

\*These two kinds of negotiation are orthogonal and thus may be used separately or in combination.

> CLIENT-SERVER PROTOCOL - HTTP/1.1 server MAY assume that a HTTP/1.1 client intends to maintain a persistent connection unless a Connection header including the connection-token "close" was sent in the request.

- HTTP/1.1 client MAY expect a connection to remain open, but would decide to keep it open based on whether the response from a server contains a Connection header with the connection-token close.

- Clients and servers SHOULD NOT assume that a persistent connection is maintained for HTTP versions less than 1.1 unless it is explicitly signaled.

- In order to remain persistent, all messages on the connection MUST have a self-defined message length

> PERSISTENT CONNECTION - provides a mechanism by which a client and a server can signal the close of a TCP connection.

STAGES:

Connection -> Request -> Disconnection

\*1.0 = nonpersistent

\*1.1 = connection can be used in multiple connections

> REQUEST PIPELINING - A client that supports persistent connections MAY "pipeline" its requests

- A server MUST send its responses to those requests in the same order that the requests were received.

- Clients SHOULD NOT pipeline requests using non-idempotent methods or non-idempotent sequences of methods

- To make everything/ anything fast

> AUTHENTICATION/ AUTHORIZATION - HTTP provides several OPTIONAL challenge-response authentication mechanisms which can be used by a server to challenge a client request and by a client to provide authentication information.

**HTTP RESOURCE ADDRESSING**

* URI (Uniform Resource Identifier) – it is the common point in referencing a network based resource. I has two types:
  + URN (Uniform Resource Name) – is in the form “urn:.” urn is mostly used when the resource is in the directory that you specified and that it is not moved somewhere else, then you specify the name of the resource that you intended.
  + URL – is in the form “http://.. or https://”is your typical web address that points to a specific location that you want to.
* HTTP URL COMPONENTS
  + - Scheme (http/https/FTP/File) is the first word you see in a url
    - Authority that has user information or host name of the machine which is resolved to an IP address using DNS(Descriptive Naming System)
    - Port number is by default set to 80 for HTTP and 40 for HTTPS
    - Path to Resource is dynamic or static. For example /info/profile.php?
    - Query Value (?) : key=value, (&) : where, (%) : if same

URL Writing

* + Fragment Identifier (#) is also known as bookmarks for example : <div id=”a”> then in the URL we type: index.html#a

**REQUEST MESSAGE**

> REQUEST LINE - begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF where the elements are separated by SP characters

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

> Method - indicates the method to be performed on the resource identified by the Request-URI.

- Case Sensitive

> Request-URI - a Uniform Resource Identifier and identifies the resource upon which to apply the request.

Request-URI = "\*" | absoluteURI | abs\_path | authority

\* four options for Request-URI are dependent on the nature of the request.

\* asterisk "\*" - the request does not apply to a particular resource, but to the server itself

- only allowed if method used does not necessarily apply to a resource

\* absoluteURI - required when the request is being made to a proxy.

\* authority (network connection of the URI) - CONNECT method can only be used by this form

> HTTP Protocol - a request/response protocol.

\* Client send a request in the form of a request method, URI, and protocol version, followed by a MIME-like message. The server responds with a status line, including the message's protocol version and a success or error code, followed by a MIME-like message

Situation 1: HTTP communication is initiated by a user agent and consists of a request to be applied to a resource on some origin server

request chain ------------------------> UA -------------------v------------------- O <----------------------- response chain

Situation 2: one or more intermediaries are present in the request/response chain

request chain --------------------------------------> UA -----v----- A -----v----- B -----v----- C -----v----- O <------------------------------------- response chain

Legend:

UA - User Agent

O - Origin Server

A, B, C - intermediaries

Three common forms of intermediary: proxy, gateway, and tunnel.

> Proxy - forwarding agent, receiving requests, rewriting messages, and forwarding the reformatted request toward the server identified by the URI.

> Gateway - receiving agent, acting as a layer above some other server(s) and, translating the requests to the underlying server's

protocol (if neccessay).

> Tunnel - acts as a relay point between two connections without changing the messages.

- tunnels are used when the communication needs to pass through an intermediary even when the intermediary cannot understand the contents of the messages.

> MESSAGE HEADER - includes GENERAL, REQUEST, RESPONSE, and ENTITY HEADERS

> General - can be used by clients and servers

> Request - can be used by clients

> Response - can be used by servers

> Entity - Entity-header fields define metainformation about the entity-body or about the resource identified by the request

\*One header is required in HTML 1.1

> EMPTY LINE - indicates the end of the header fields, and possibly a message-body.

> MESSAGE BODY (payload)- used to carry the entity-body associated with the request or response. The message-body differs from the entity-body only when a transfer-coding has been applied.

**HTTP REQUEST METHOD**

> Standard Method

> Get - retrieving whatever information (in the form of an entity) is identified by the Request-URI.

- returns responses as entity

- commonly used

- Response is cacheable if and only if it meets the requirements for HTTP Caching

> Head - identical to GET except that the server MUST NOT return a message-body in the response.

- Metainformation in the HTTP headers in response to a HEAD request SHOULD be identical to the information sent in response to a GET request.

- general purpose

- Often used to test the validity, accessibility and recent modification of hypertext links.

- may be cacheable.

> Post - Used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line.

- Its function is determined by the server and is usually dependent on the Request-URI.

- Responses are not cacheable (unless the response inclused Cache-Control or Expires header fields)

> Put - requests that the enclosed entity be stored under the supplied Request-URI.

\* If there is an existing resource, the enclosed entity should considered to be a modified version in the origin server.

- Responses are not cacheable

> Delete - requests that the origin server delete the resource identified by the Request-URI.

- May be overridden by human intervention on the origin server

> Option - represents a request for information about the communication options available on the request/response chain identified by the Request-URI

- asks the server what to do

- Responses are not cacheable

> Trace - used to invoke a remote, application-layer loop-back of the request message.

- must not include an entity.

- allows the client to see what is being received at the other end of the request chain

- diagnostic purposes

- responses must not be cached

> Connect - use with a proxy that can dynamically switch to being a tunnel

-establish tunnel -> blind relay

> Safe Method - the GET and HEAD methods SHOULD NOT have the significance of taking an action other than retrieval (method considered safe)

e.g GET, HEAD, OPTIONS, TRACE

- methods has no modification

> Idempotent Method - no concurrent operations are being executed on the same set of resources

e.g GET, HEAD, OPTIONS, TRACE, PUT, DELETE

-can be used several times

> Cacheable Method - if the cache is allowed to store a copy of the response message

**Message Headers**

The HTTP header fields provide information about the request, response, or about the object sent in the message body. HTTP header fields include General Header Fields, Request Header Fields, Response Header Field, and Entity Header Fields.

The format to be followed (taken from RFC 2616):

message-header = field-name ":" [ field-value ]

field-name = token

field-value = \*( field-content | LWS )

field-content = <the OCTETs making up the field-value

and consisting of either \*TEXT or combinations

of token, separators, and quoted-string>

General Header Fields – have general applicability for both request and response messages.

* Cache-control – used to specify directives that must be obeyed by all the caching system.
* Connection – allows the sender to specify options that are desired for a certain connection.
* Date – all data and time should be represented in Greenwhich Mean Time (GMT).
* Pragma – used to include implementation of specific directives that can apply to any recipient along the request and response chain.
* Trailer – general field value indicates that the set of header fields in present in the last part of a message.
* Transfer-encoding – indicates what type of transformation has been made to the message body in order to securely transfer it.
* Upgrade – allows the client to specify additional protocols that it supports and would like to be use.
* Via – must be used by gateways and proxies to indicate the intermediate protocols and recipients.
* Warning – used to carry additional information about the status of a message in which might not be reflected in the message.

Request Header Fields – have applicability for request messages.

* Accept – can be used to specify certain media types to consider whether if it is acceptable for the response.
* Accept-charset – to indicate the character sets that can be accepted for the response.
* Accept-encoding – similar to Accept except it restricts that content-codings that can be accepted in the response.
* Accept-language – similar to Accept except it restricts the set of natural languages that are preferred as a response to the request.
* Authorization – consists of credentials as a mean of authentication information of the user agent.
* Expect – used to indicate that a certain set of server behavior is required and expected by the client.
* From – contains an Internet email address for the user who is controlling the requesting user agent.
* Host – used to specify the Internet host and port number of the requested resource.
* If-match – to perform the requested method if the given value matches the given entity tags.
* If-modified-since – if the requested URL is still not modified since the time specified in this field, the entity will not be returned from the server and a 304 response will be returned.
* If-none-match – requests the server to do the requested method only if one of the given value matches the given entity tags.
* If-range – can be used with a conditional GET to request a portion of the entity that is missing, if it not been changed, and the entire entity has been changed.
* If-unmodified-since – if the requested resource has not been modified since the time specified, the server would perform the requested operation as if this header were not present.
* Max-forwards – provides a mechanism with the TRACE and OPTIONS methods in order to limit the number of proxies or gateways that can forward the request to the next server.
* Proxy-authorization – allows the client to identify itself to a proxy in which requires authentication.
* Range – specified the partial ranges of the content that has been requested from the document.
* Referrer – allows the client to specify the address of the resource from which URL is requested.
* TE – indicates what extension transfer-coding it is willing to accept in the response and if it is willing or not to accept the trailer fields in a chunked transfer-coding.
* User-agent – contains information about the user agent in which where the request is originating.

Response Header Fields – have the applicability for response messages.

* Accept-ranges – allows the server to indicate its range requests acceptance for a resource.
* Age – conveys the sender’s estimate of the time since the response was generated at the server.
* ETag (entity tag) – provides the current value of the entity tag for the requested variant.
* Location – used to redirect the recipient to another location other than the Request-URI.
* Proxy-authenticate – included as part of the 407 response.
* Retry-after – can be used with the 503 response in order to indicate how long the service is expected to be unavailable for the client.
* Server – contains information about the software used by the server in handling requests.
* Vary – specifies that the entity has multiple sources and therefore it can vary accordingly to the specified list of request headers.
* WWW-Authenticate – should consist of at least one challenge that would indicate the authentication schemes and parameters that are applicable to the Request-URI.

Entity Header Fields – define meta information about the entity-body or about the resource identified by the request.

* Allow – lists the methods that are supported by the resource indentified by the Request-URI.
* Content-encoding – a modifier to the media-type.
* Content-language – describe the natural languages of the audience for the enclosed entity, although multiple languages can be listed for multiple languages.
* Content-length – indicates the size of the entity-body.
* Content-location – may be used in order to give the resource location for the entity enclosed in the message when that entity is accessible from a location separate from the requested resource’s URI.
* Content-MD5 (message digest) – used to supply an MD5 digest in order to check the integrity of the message upon receipt.
* Content-range – to specify where in the full entity-body should the partial body be applied.
* Content-type – indicate the media type of the entity-body that has been sent to the recipient.
* Expires – give the date and time in which where the response would be considered stale.
* Last-modified – indicates the date and time at which the server believes the variant was last modified.

**HTTP STATUS CODES**

1. Informational 1xx

Indicates a provisional response.

* 1. 100 Continue

Response used to inform the client that the initial part of the request has been received and has not yet been rejected by the server.

* 1. 101 Switching Protocols

1. Successful 2xx

Indicates that the client’s request was successfully received, understood, and accepted.

* 1. 200 OK

A certain request has succeeded. The information returned with the response is dependent on the method used in the request like:

* GET, entity corresponding to the requested resource are sent in the response
* HEAD, entity-header fields corresponding to the requested resource are sent in the response without any message body.
* POST, entity describing or containing the result of the action.
* TRACE, entity containing the request message as received by the end server.
  1. 201 Created

Indicates that a request has been fulfilled and resulted in a new resource being created.

* 1. 202 Accepted

Indicates that a 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 takes place. The 202 response is intentionally non-committal. Its purpose is to allow a server to accept a request for some other process without requiring that the user agent’s connection to the server persist until the process is completed.

* 1. 203 Non-Authoritative Information

Indicates that the returned metainformation in the entity-header is not the definitive set as available from the origin server, but is gathered from a local or a third-party copy.

* 1. 204 No Content

Indicates that the server has fulfilled the request but does not need to return an entity-body, and might want to return updated information.

* 1. 205 Reset Content

Indicates that the server has fulfilled the request and the user agent should reset the document view which cause the request to be sent. This response is primarily intended to allow input for actions to take place via user input, followed by a clearing of the form in which the input is given so that the user can easily initiate another input action.

* 1. 206 Partial Content

Indicates that the server has fulfilled the partial GET request for the resource. This response must include header fields like Content-Range, date, Etag/Content-Location and Expires, Cache-control and vary.

1. Redirection 3xx

This class of status codes indicates tha further action needs to be taken by the user agent in order to fulfil the request.

* 1. 300 Multiple Choice

Indicates that the requested resource corresponds to any one of a set of representations, each with its own specific location, and agent-driven negotiation information is being provide so that the user agent can select a preferred representation and redirect its request to that location.

* 1. 301 Moved Permanently

Indicates that the requested resource has been assigned a new permanent URI and any future references to this resource should use one of the returned URIs. This response is cacheable unless indicated otherwise. If this response is received in response to a request other that GET or HEAD, the user agent must not automatically redirect the request unless it can be confirmed by the user, since this might change the conditions under which the request was issued.

* 1. 302 Found

Indicates that the requested resource resides temporarily under a different URI. Since the redirection might be altered in occasion, the client should continue to use the Request-URI for future requests.

* 1. 303 See Other

Indicates that the response to the request can be found under a different URI and should be retrieved using a GET method on that resource. This method exists primarily to allow the output of a POST activated script to redirect the user agent to as selected resource. This response must not be cached, but the response to the second (redirected) request might be cacheable.

* 1. 304 Not Modified

Indicates that the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server should respond with this status code. This response must not contain a message-body, and thus is always terminated by the first empty line after the header fields.

* 1. 305 Use Proxy

Indicates that the requested resource must be accessed through the proxy given by the Location field. The Location field gives the URI of the proxy. The recipient is expected to repeat this single request via the proxy. This response must only be generated by origin servers.

* 1. 306 (Unused)

This status code is no longer used, and the code was reserved.

* 1. 307 Temporary Redirect

Indicates that the requested resource resides temporarily under a different URI. Since he redirection may be altered on occasion, the client should continue to use the Request-URI for future requests.

1. Client Error 4xx

This class of status codes is intended for cases in which the client seems to have erred. These status codes are applicable to any request method.

* 1. 400 Bad Request

Indicates that the request could not be understood by the server due to malformed syntax. The client should not repeat the request without modifications.

* 1. 401 Unauthorized

Indicates that the request requires user authentication.

* 1. 402 Payment Required

This code is reserved for future use.

* 1. 403 Forbidden

Indicates that the server understood the request, but is refusing to fulfil it.

* 1. 404 Not Found

Indicates that the server has not found anything matching the Request-URI.

* 1. 405 Method Not Allowed

Indicates that the method specified in the Request-Line is not allowed or the resource identified by the Request-URI.

* 1. 406 Not Acceptable

Indicates that the resource identified by the request is only capable of generating response entities which have content characteristics not acceptable according to the accept headers sent in the request.

* 1. 407 Proxy Authentication Required

This code is similar to 401, but indicates that the client must first authenticate itself with the proxy. The proxy must return a Proxy-Authenticate header field containing a challenge applicable to the proxy for the requested resource.

* 1. 408 Request Timeout

Indicates that the client did not produce a request within the time that that the server was prepared to wait.

* 1. 409 Conflict

Indicates that request could not be completed due to a conflict with the current state of the resource.

* 1. 410 Gone

Indicates that the request resource is no longer available at the server and no forwarding address is known.

* 1. 411 Length Required

Indicates that the server refuse to accept the request without a defined Content-Length.

* 1. 412 Precondition Failed

Indicates that the precondition given in one or more of the request-header fields evaluated to false when it was tested on the server.

* 1. 413 Request Entity Too Large

Indicates that the server is refusing to process a request because the request entity is larger than that the server is willing or able to process.

* 1. 414 Request-URI Too Long

Indicates that the server is refusing to service that request because the Request-URI is longer that the server is willing to interpret.

* 1. 415 Unsupported Media Type

Indicates that the server is refusing to service the request because the entity of the request is in format not supported by the requested resource for the requested method.

* 1. 416 Requested Range Not Satisfiable

Indicates that a server should return a response with this status code I a request includes a Range request-header field and none of the range-specifier value in this field overlap the current extent of the selected resource, and the request did not include an If-Range request-header field.

* 1. 417 Expectation Failed

Indicates that the expectation given in an Expect request-header field could not be met by this server, or, if the server is a proxy, the server has unambiguous evidence that the request could not be met by the next-hop server.

1. Server Error 5xx

Class of status codes indicating cases on which the server is aware that it has erred or is incapable of performing the request.

* 1. 500 Internal Server Error

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

* 1. 501 Not Implemented

Indicates that the server does not support the functionality required to fulfil 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.

* 1. 502 Bad Gateway

Indicates that the server, while acting as a gateway or proxy, received an invalid response from the upstream server it accessed in attempting to fulfil the request.

* 1. 503 Service Unavailable

Indicates that the server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

* 1. 504 Gateway Timeout

Indicates that a server currently acting as a gateway or proxy did not receive a timely response from the upstream server specified by the URI or some other auxiliary server it need to access in attempting to complete the request.

* 1. 505 HTTP Version Not Supported

Indicates that the server does not support the HTTP protocol version that was used in the request message.