HTTP - HEADER FIELDS

HTTP header fields provide required information about the request or response, or about the object sent in the message body. There are four types of HTTP message headers:

- **General-header:** These header fields have general applicability for both request and response messages.
- **Client Request-header:** These header fields have applicability only for request messages.
- **Server Response-header:** These header fields have applicability only for response messages.
- **Entity-header:** These header fields define meta information about the entity-body or, if no body is present, about the resource identified by the request.

General Headers

Cache-Control

The Cache-Control general-header field is used to specify directives that MUST be obeyed by all the caching system. The syntax is as follows:

```
Cache-Control : cache-request-directive|cache-response-directive
```

An HTTP client or server can use the **Cache-control** general header to specify parameters for the cache or to request certain kinds of documents from the cache. The caching directives are specified in a comma-separated list. For example:

```
Cache-control: no-cache
```

The following table lists the important cache request directives that can be used by the client in its HTTP request:

S.N.	Cache Request Directive and Description
1	**no-cache** A cache must not use the response to satisfy a subsequent request without successful revalidation with the origin server.
2	**no-store** The cache should not store anything about the client request or server response.
3	**max-age = seconds** Indicates that the client is willing to accept a response whose age is not greater than the specified time in seconds.
4	**max-stale [= seconds]** Indicates that the client is willing to accept a response that has exceeded its expiration time. If seconds are given, it must not be expired by more than that time.
5	**min-fresh = seconds**

	Indicates that the client is willing to accept a response whose freshness lifetime is not less than its current age plus the specified time in seconds.
6	**no-transform** Does not convert the entity-body.
7	**only-if-cached** Does not retrieve new data. The cache can send a document only if it is in the cache, and should not contact the origin-server to see if a newer copy exists.

The following important cache response directives that can be used by the server in its HTTP response:

S.N.	Cache Request Directive and Description
1	**public** Indicates that the response may be cached by any cache.
2	**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.
3	**no-cache** A cache must not use the response to satisfy a subsequent request without successful re-validation with the origin server.
4	**no-store** The cache should not store anything about the client request or server response.
5	**no-transform** Does not convert the entity-body.
6	**must-revalidate** The cache must verify the status of the stale documents before using it and expired ones should not be used.
7	**proxy-revalidate** The proxy-revalidate directive has the same meaning as the must- revalidate directive, except that it does not apply to non-shared user agent caches.
8	**max-age = seconds** Indicates that the client is willing to accept a response whose age is not greater than the specified time in seconds.
9	**s-maxage = seconds**

The maximum age specified by this directive overrides the maximum age specified by either the max-age directive or the Expires header. The s-maxage directive is always ignored by a private cache.

Connection

The Connection general-header field allows the sender to specify options that are desired for that particular connection and must not be communicated by proxies over further connections. Following is the simple syntax for using connection header:

```
Connection : "Connection"
```

HTTP/1.1 defines the "close" connection option for the sender to signal that the connection will be closed after completion of the response. For example:

```
Connection: close
```

By default, HTTP 1.1 uses persistent connections, where the connection does not automatically close after a transaction. HTTP 1.0, on the other hand, does not have persistent connections by default. If a 1.0 client wishes to use persistent connections, it uses the **keep-alive** parameter as follows:

```
Connection: keep-alive
```

Date

All HTTP date/time stamps MUST be represented in Greenwich Mean Time GMT, without exception. HTTP applications are allowed to use any of the following three representations of date/time stamps:

```
Sun, 06 Nov 1994 08:49:37 GMT  ; RFC 822, updated by RFC 1123
Sunday, 06-Nov-94 08:49:37 GMT ; RFC 850, obsoleted by RFC 1036
Sun Nov  6 08:49:37 1994       ; ANSI C's asctime() format
```

Here the first format is the most preferred one.

Pragma

The Pragma general-header field is used to include implementation specific directives that might apply to any recipient along the request/response chain. For example:

```
Pragma: no-cache
```

The only directive defined in HTTP/1.0 is the no-cache directive and is maintained in HTTP 1.1 for backward compatibility. No new Pragma directives will be defined in the future.

Trailer

The Trailer general field value indicates that the given set of header fields is present in the trailer of a message encoded with chunked transfer-coding. Following is the syntax of Trailer header field:

```
Trailer : field-name
```

Message header fields listed in the Trailer header field must not include the following header fields:

- Transfer-Encoding
- Content-Length

- Trailer

Transfer-Encoding

The *Transfer-Encoding* general-header field indicates what type of transformation has been applied to the message body in order to safely transfer it between the sender and the recipient. This is not the same as content-encoding because transfer-encodings are a property of the message, not of the entity-body. The syntax of Transfer-Encoding header field is as follows:

```
Transfer-Encoding: chunked
```

All transfer-coding values are case-insensitive.

Upgrade

The *Upgrade* general-header allows the client to specify what additional communication protocols it supports and would like to use if the server finds it appropriate to switch protocols. For example:

```
Upgrade: HTTP/2.0, SHTTP/1.3, IRC/6.9, RTA/x11
```

The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol.

Via

The *Via* general-header must be used by gateways and proxies to indicate the intermediate protocols and recipients. For example, a request message could be sent from an HTTP/1.0 user agent to an internal proxy code-named "fred", which uses HTTP/1.1 to forward the request to a public proxy at nowhere.com, which completes the request by forwarding it to the origin server at www.ics.uci.edu. The request received by www.ics.uci.edu would then have the following Via header field:

```
Via: 1.0 fred, 1.1 nowhere.com (Apache/1.1)
```

The Upgrade header field is intended to provide a simple mechanism for transition from HTTP/1.1 to some other, incompatible protocol.

Warning

The *Warning* general-header is used to carry additional information about the status or transformation of a message which might not be reflected in the message. A response may carry more than one Warning header.

```
Warning : warn-code SP warn-agent SP warn-text SP warn-date
```

Client Request Headers

Accept

The *Accept* request-header field can be used to specify certain media types which are acceptable for the response. The general syntax is as follows:

```
Accept: type/subtype [q=qvalue]
```

Multiple media types can be listed separated by commas and the optional qvalue represents an acceptable quality level for accept types on a scale of 0 to 1. Following is an example:

```
Accept: text/plain; q=0.5, text/html, text/x-dvi; q=0.8, text/x-c
```

This would be interpreted as **text/html** and **text/x-c** and are the preferred media types, but if

they do not exist, then send the **text/x-dvi** entity, and if that does not exist, send the **text/plain** entity.

Accept-Charset

The *Accept-Charset* request-header field can be used to indicate what character sets are acceptable for the response. Following is the general syntax:

```
Accept-Charset: character_set [q=qvalue]
```

Multiple character sets can be listed separated by commas and the optional qvalue represents an acceptable quality level for nonpreferred character sets on a scale of 0 to 1. Following is an example:

```
Accept-Charset: iso-8859-5, unicode-1-1; q=0.8
```

The special value "*", if present in the **Accept-Charset** field, matches every character set and if no **Accept-Charset** header is present, the default is that any character set is acceptable.

Accept-Encoding

The *Accept-Encoding* request-header field is similar to Accept, but restricts the content-codings that are acceptable in the response. The general syntax is:

```
Accept-Encoding: encoding types
```

Examples are as follows:

```
Accept-Encoding: compress, gzip
Accept-Encoding:
Accept-Encoding: *
Accept-Encoding: compress;q=0.5, gzip;q=1.0
Accept-Encoding: gzip;q=1.0, identity; q=0.5, *;q=0
```

Accept-Language

The *Accept-Language* request-header field is similar to Accept, but restricts the set of natural languages that are preferred as a response to the request. The general syntax is:

```
Accept-Language: language [q=qvalue]
```

Multiple languages can be listed separated by commas and the optional qvalue represents an acceptable quality level for non preferred languages on a scale of 0 to 1. Following is an example:

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

Authorization

The *Authorization* request-header field value consists of credentials containing the authentication information of the user agent for the realm of the resource being requested. The general syntax is:

```
Authorization : credentials
```

The HTTP/1.0 specification defines the BASIC authorization scheme, where the authorization parameter is the string of **username:password** encoded in base 64. Following is an example:

```
Authorization: BASIC Z3Vlc3Q6Z3Vlc3QxMjM=
```

The value decodes into is **guest:guest123** where **guest** is user ID and **guest123** is the password.

Cookie

The *Cookie* request-header field value contains a name/value pair of information stored for that URL. Following is the general syntax:

```
Cookie: name=value
```

Multiple cookies can be specified separated by semicolons as follows:

```
Cookie: name1=value1;name2=value2;name3=value3
```

Expect

The *Expect* request-header field is used to indicate that a particular set of server behaviors is required by the client. The general syntax is:

```
Expect : 100-continue | expectation-extension
```

If a server receives a request containing an Expect field that includes an expectation-extension that it does not support, it must respond with a 417 *ExpectationFailed* status.

From

The *From* request-header field contains an Internet e-mail address for the human user who controls the requesting user agent. Following is a simple example:

```
From: webmaster@w3.org
```

This header field may be used for logging purposes and as a means for identifying the source of invalid or unwanted requests.

Host

The *Host* request-header field is used to specify the Internet host and the port number of the resource being requested. The general syntax is:

```
Host : "Host" ":" host [ ":" port ] ;
```

A **host** without any trailing port information implies the default port, which is 80. For example, a request on the origin server for *http://www.w3.org/pub/WWW/* would be:

```
GET /pub/WWW/ HTTP/1.1
Host: www.w3.org
```

If-Match

The *If-Match* request-header field is used with a method to make it conditional. This header requests the server to perform the requested method only if the given value in this tag matches the given entity tags represented by **ETag**. The general syntax is:

```
If-Match : entity-tag
```

An asterisk ∗ matches any entity, and the transaction continues only if the entity exists. Following are possible examples:

```
If-Match: "xyzzy"
If-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"
If-Match: *
```

If none of the entity tags match, or if "*" is given and no current entity exists, the server must not

perform the requested method, and must return a 412 *PreconditionFailed* response.

If-Modified-Since

The *If-Modified-Since* request-header field is used with a method to make it conditional. If the requested URL has not been modified since the time specified in this field, an entity will not be returned from the server; instead, a 304 *notmodified* response will be returned without any message-body. The general syntax of if-modified-since is:

```
If-Modified-Since : HTTP-date
```

An example of the field is:

```
If-Modified-Since: Sat, 29 Oct 1994 19:43:31 GMT
```

If none of the entity tags match, or if "*" is given and no current entity exists, the server must not perform the requested method, and must return a 412 *PreconditionFailed* response.

If-None-Match

The *If-None-Match* request-header field is used with a method to make it conditional. This header requests the server to perform the requested method only if one of the given value in this tag matches the given entity tags represented by **ETag**. The general syntax is:

```
If-None-Match : entity-tag
```

An asterisk ∗ matches any entity, and the transaction continues only if the entity does not exist. Following are the possible examples:

```
If-None-Match: "xyzzy"
If-None-Match: "xyzzy", "r2d2xxxx", "c3piozzzz"
If-None-Match: *
```

If-Range

The *If-Range* request-header field can be used with a conditional GET to request only the portion of the entity that is missing, if it has not been changed, and the entire entity if it has been changed. The general syntax is as follows:

```
If-Range : entity-tag | HTTP-date
```

Either an entity tag or a date can be used to identify the partial entity already received. For example:

```
If-Range: Sat, 29 Oct 1994 19:43:31 GMT
```

Here if the document has not been modified since the given date, the server returns the byte range given by the Range header, otherwise it returns all of the new document.

If-Unmodified-Since

The *If-Unmodified-Since* request-header field is used with a method to make it conditional. The general syntax is:

```
If-Unmodified-Since : HTTP-date
```

If the requested resource has not been modified since the time specified in this field, the server should perform the requested operation as if the If-Unmodified-Since header were not present. For example:

```
If-Unmodified-Since: Sat, 29 Oct 1994 19:43:31 GMT
```

If the request results in anything other than a 2xx or 412 status, the *If-Unmodified-Since* header should be ignored.

Max-Forwards

The *Max-Forwards* request-header field provides a mechanism with the TRACE and OPTIONS methods to limit the number of proxies or gateways that can forward the request to the next inbound server. Here is the general syntax:

```
Max-Forwards : n
```

The Max-Forwards value is a decimal integer indicating the remaining number of times this request message may be forwarded. This is useful for debugging with the TRACE method, avoiding infinite loops. For example:

```
Max-Forwards : 5
```

The Max-Forwards header field may be ignored for all other methods defined in the HTTP specification.

Proxy-Authorization

The *Proxy-Authorization* request-header field allows the client to identify itself *oritsuser* to a proxy which requires authentication. Here is the general syntax:

```
Proxy-Authorization : credentials
```

The Proxy-Authorization field value consists of credentials containing the authentication information of the user agent for the proxy and/or realm of the resource being requested.

Range

The *Range* request-header field specifies the partial range*s* of the content requested from the document. The general syntax is:

```
Range: bytes-unit=first-byte-pos "-" [last-byte-pos]
```

The first-byte-pos value in a byte-range-spec gives the byte-offset of the first byte in a range. The last-byte-pos value gives the byte-offset of the last byte in the range; that is, the byte positions specified are inclusive. You can specify a byte-unit as bytes. Byte offsets start at zero. Some simple examples are as follows:

```
- The first 500 bytes
Range: bytes=0-499

- The second 500 bytes
Range: bytes=500-999

- The final 500 bytes
Range: bytes=-500

- The first and last bytes only
Range: bytes=0-0,-1
```

Multiple ranges can be listed, separated by commas. If the first digit in the comma-separated byte range*s* is missing, the range is assumed to count from the end of the document. If the second digit is missing, the range is byte n to the end of the document.

Referer

The *Referer* request-header field allows the client to specify the address *URI* of the resource from

which the URL has been requested. The general syntax is as follows:

```
Referer : absoluteURI | relativeURI
```

Following is a simple example:

```
Referer: http://www.tutorialspoint.org/http/index.htm
```

If the field value is a relative URI, it should be interpreted relative to the *Request-URI*.

TE

The *TE* request-header field indicates what extension *transfer-coding* it is willing to accept in the response and whether or not it is willing to accept trailer fields in a chunked *transfer-coding*. Following is the general syntax:

```
TE   : t-codings
```

The presence of the keyword "trailers" indicates that the client is willing to accept trailer fields in a chunked transfer-coding and it is specified either of the ways:

```
TE: deflate
TE:
TE: trailers, deflate;q=0.5
```

If the TE field-value is empty or if no TE field is present, then only transfer-coding is *chunked*. A message with no transfer-coding is always acceptable.

User-Agent

The *User-Agent* request-header field contains information about the user agent originating the request. Following is the general syntax:

```
User-Agent : product | comment
```

Example:

```
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
```

Server Response Headers

Accept-Ranges

The *Accept-Ranges* response-header field allows the server to indicate its acceptance of range requests for a resource. The general syntax is:

```
Accept-Ranges  : range-unit | none
```

For example a server that accepts byte-range requests may send:

```
Accept-Ranges: bytes
```

Servers that do not accept any kind of range request for a resource may send:

```
Accept-Ranges: none
```

This will advise the client not to attempt a range request.

Age

The *Age* response-header field conveys the sender's estimate of the amount of time since the

response *oritsrevalidation* was generated at the origin server. The general syntax is:

```
Age : delta-seconds
```

Age values are non-negative decimal integers, representing time in seconds. Following is a simple example:

```
Age: 1030
```

An HTTP/1.1 server that includes a cache must include an Age header field in every response generated from its own cache.

ETag

The *ETag* response-header field provides the current value of the entity tag for the requested variant. The general syntax is:

```
ETag :  entity-tag
```

Here are some simple examples:

```
ETag: "xyzzy"
ETag: W/"xyzzy"
ETag: ""
```

Location

The *Location* response-header field is used to redirect the recipient to a location other than the Request-URI for completion. The general syntax is:

```
Location : absoluteURI
```

Following is a simple example:

```
Location: http://www.tutorialspoint.org/http/index.htm
```

The Content-Location header field differs from Location in that the Content-Location identifies the original location of the entity enclosed in the request.

Proxy-Authenticate

The *Proxy-Authenticate* response-header field must be included as a part of a 407 *ProxyAuthenticationRequired* response. The general syntax is:

```
Proxy-Authenticate  : challenge
```

Retry-After

The *Retry-After* response-header field can be used with a 503 *ServiceUnavailable* response to indicate how long the service is expected to be unavailable to the requesting client. The general syntax is:

```
Retry-After : HTTP-date | delta-seconds
```

Examples:

```
Retry-After: Fri, 31 Dec 1999 23:59:59 GMT
Retry-After: 120
```

In the latter example, the delay is 2 minutes.

Server

The *Server* response-header field contains information about the software used by the origin server to handle the request. The general syntax is:

```
Server : product | comment
```

Following is a simple example:

```
Server: Apache/2.2.14 (Win32)
```

If the response is being forwarded through a proxy, the proxy application must not modify the Server response-header.

Set-Cookie

The *Set-Cookie* response-header field contains a name/value pair of information to retain for this URL. The general syntax is:

```
Set-Cookie: NAME=VALUE; OPTIONS
```

Set-Cookie response header comprises the token Set-Cookie, followed by a comma-separated list of one or more cookies. Here are the possible values you can specify as options:

S.N.	Options and Description
1	**Comment=comment** This option can be used to specify any comment associated with the cookie.
2	**Domain=domain** The Domain attribute specifies the domain for which the cookie is valid.
3	**Expires=Date-time** The date the cookie will expire. If it is blank, the cookie will expire when the visitor quits the browser.
4	**Path=path** The Path attribute specifies the subset of URLs to which this cookie applies.
5	**Secure** It instructs the user agent to return the cookie only under a secure connection.

Following is an example of a simple cookie header generated by the server:

```
Set-Cookie: name1=value1,name2=value2; Expires=Wed, 09 Jun 2021 10:18:14 GMT
```

Vary

The *Vary* response-header field specifies that the entity has multiple sources and may therefore vary according to the specified list of request header*s*. Following is the general syntax:

```
Vary : field-name
```

You can specify multiple headers separated by commas and a value of asterisk "*" signals that unspecified parameters are not limited to the request-headers. Following is a simple example:

```
Vary: Accept-Language, Accept-Encoding
```

Here field names are case-insensitive.

WWW-Authenticate

The *WWW-Authenticate* response-header field must be included in 401 *Unauthorized* response messages. The field value consists of at least one challenge that indicates the authentication scheme*s* and parameters applicable to the Request-URI. The general syntax is:

```
WWW-Authenticate : challenge
```

WWW- Authenticate field value might contain more than one challenge, or if more than one WWW-Authenticate header field is provided, the contents of a challenge itself can contain a comma-separated list of authentication parameters. Following is a simple example:

```
WWW-Authenticate: BASIC realm="Admin"
```

Entity Headers

Allow

The *Allow* entity-header field lists the set of methods supported by the resource identified by the Request-URI. The general syntax is:

```
Allow : Method
```

You can specify multiple methods separated by commas. Following is a simple example:

```
Allow: GET, HEAD, PUT
```

This field cannot prevent a client from trying other methods.

Content-Encoding

The *Content-Encoding* entity-header field is used as a modifier to the media-type. The general syntax is:

```
Content-Encoding : content-coding
```

The content-coding is a characteristic of the entity identified by the Request-URI. Following is a simple example:

```
Content-Encoding: gzip
```

If the content-coding of an entity in a request message is not acceptable to the origin server, the server should respond with a status code of 415 *UnsupportedMediaType*.

Content-Language

The *Content-Language* entity-header field describes the natural language*s* of the intended audience for the enclosed entity. Following is the general syntax:

```
Content-Language : language-tag
```

Multiple languages may be listed for content that is intended for multiple audiences. Following is a

simple example:

```
Content-Language: mi, en
```

The primary purpose of Content-Language is to allow a user to identify and differentiate entities according to the user's own preferred language.

Content-Length

The *Content-Length* entity-header field indicates the size of the entity-body, in decimal number of OCTETs, sent to the recipient or, in the case of the HEAD method, the size of the entity-body that would have been sent, had the request been a GET. The general syntax is:

```
Content-Length : DIGITS
```

Following is a simple example:

```
Content-Length: 3495
```

Any Content-Length greater than or equal to zero is a valid value.

Content-Location

The *Content-Location* entity-header field may be used to supply 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. The general syntax is:

```
Content-Location:  absoluteURI | relativeURI
```

Following is a simple example:

```
Content-Location: http://www.tutorialspoint.org/http/index.htm
```

The value of Content-Location also defines the base URI for the entity.

Content-MD5

The *Content-MD5* entity-header field may be used to supply an MD5 digest of the entity for checking the integrity of the message upon receipt. The general syntax is:

```
Content-MD5  : md5-digest using base64 of 128 bit MD5 digest as per RFC 1864
```

Following is a simple example:

```
Content-MD5  : 8c2d46911f3f5a326455f0ed7a8ed3b3
```

The MD5 digest is computed based on the content of the entity-body, including any content-coding that has been applied, but not including any transfer-encoding applied to the message-body.

Content-Range

The *Content-Range* entity-header field is sent with a partial entity-body to specify where in the full entity-body the partial body should be applied. The general syntax is:

```
Content-Range : bytes-unit SP first-byte-pos "-" last-byte-pos
```

Examples of byte-content-range-spec values, assuming that the entity contains a total of 1234 bytes:

```
- The first 500 bytes:
```

```
Content-Range : bytes 0-499/1234

- The second 500 bytes:
Content-Range : bytes 500-999/1234

- All except for the first 500 bytes:
Content-Range : bytes 500-1233/1234

- The last 500 bytes:
Content-Range : bytes 734-1233/1234
```

When an HTTP message includes the content of a single range, this content is transmitted with a Content-Range header, and a Content-Length header showing the number of bytes actually transferred. 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
```

Content-Type

The *Content-Type* entity-header field indicates the media type of the entity-body sent to the recipient or, in the case of the HEAD method, the media type that would have been sent, had the request been a GET. The general syntax is:

```
Content-Type : media-type
```

Following is an example:

```
Content-Type: text/html; charset=ISO-8859-4
```

Expires

The *Expires* entity-header field gives the date/time after which the response is considered stale. The general syntax is:

```
Expires : HTTP-date
```

Following is an example:

```
Expires: Thu, 01 Dec 1994 16:00:00 GMT
```

Last-Modified

The *Last-Modified* entity-header field indicates the date and time at which the origin server believes the variant was last modified. The general syntax is:

```
Last-Modified: HTTP-date
```

Following is an example:

```
Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
```