FREE eBook

LEARNING http-headers

Free unaffiliated eBook created from
Stack Overflow contributors.

#http-headers

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: http-headers

It is an unofficial and free http-headers ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official http-headers.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with http-headers

Remarks

HTTP Headers are an important part of HTTP communication. Each HTTP request and HTTP response usually contain multiple headers. Intermediaries such as proxies often interpret some of the headers and pass on or filter out others.

Examples

HTTP Request

A simple HTTP Request for the resource `/index.html`. The host `www.example.com` is specified in the HTTP `Host` header.

```
GET /index.html HTTP/1.1
Host: www.example.com
```

HTTP Response

A possible response to the request above. The response contains the HTTP headers `Date`, `Content-Type`, `Content-Encoding` and `Content-Length`.

```
HTTP/1.1 200 OK
Date: Wed, 21 Jun 2017 10:58:03 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 150

<response body>
```

The response body is sent after the headers, separated by a blank line.

Read Getting started with http-headers online: https://riptutorial.com/http-headers/topic/10573/getting-started-with-http-headers

Chapter 2: Accept: (Request)

Introduction

What `Content-Type` does the client accept.

Syntax

- Accept: MIMEType/MIMESubtype;QualityFactor
- Accept: MIMEType/MIMESubtype;QualityFactor, MIMEType/MIMESubtype;QualityFactor, ...

Parameters

Parameter	Description
MIMEType	The first half of the mime type. This can also be a `*/*` for all types
MIMESubtype	The second half of the mime type or a * for all sub types (ie `image/*`)
QualityFactor	The quality factor in the format `;q=0.8` (optional)

Remarks

The content types are MIME types (ie `text/html`) separated by comma with an optional quality factor (using a `;q=`) that is used the clients preference for using this type. The quality factor has a value from 0 to 1 with the higher the number the more preference for that type.

If the server can't find an acceptable type to reply with then it should send a 406 (not acceptable) response.

Examples

HTML only type

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept Content-Types of text/html

Match all text types

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept Content-Types of any of the text/* types of MIME types. For example text/html, text/plain, text/css.

text/html and application/xml with a preference text/html

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html;q=1.0,application/xml;q=0.9
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept Content-Types of text/html and application/xml but it prefers text/html

Preference for one type over another

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept all types of Content-Types but prefers text/html and application/xml

Read Accept: (Request) online: https://riptutorial.com/http-headers/topic/10614/accept---request-

Chapter 3: Accept-Charset: (Request)

Introduction

`Accept-Charset` tells the server what character sets the client accepts.

Syntax

- Accept-Charset: type;QualityFactor
- Accept-Charset: type;QualityFactor, type;QualityFactor, type;QualityFactor, ...

Parameters

Parameter	Description
type	A character set name. This can also be a `*` for all character sets
QualityFactor	The quality factor in the format `;q=0.8` (optional)

Remarks

`Accept-Charset` takes a number of character sets and includes an optional preference for which one the server should use. The charset is one from the list of available charsets at IANA "Character Sets" registry. For example `UTF-8`.

The charset is separated by commas with an optional quality factor (using a `;q=`) that is used the clients preference for using this type. The quality factor has a value from 0 to 1 with the higher the number the more preference for that type.

If this header is not included then the client will accept any charset.

The server uses `Content-Type` to inform the client what character set it is using.

If the server can't find an acceptable charset to reply with then it should send a 406 (not acceptable) response or ignore this header and not doing any content negotiation.

Examples

Only accept UTF-8

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Charset: UTF-8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept only UTF-8 char sets.

Only accept UTF-8 and iso-8859-1

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Charset: UTF-8, iso-8859-1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept only UTF-8 and iso-8859-1 char sets.

Only accept UTF-8, iso-8859-1 with a preference

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Charset: UTF-8, iso-8859-1;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept UTF-8 and iso-8859-1 char sets but prefers UTF-8 (which has a quality factor of 1.0).

Accept any charset but have preference for some types

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Charset: UTF-8, iso-8859-1;q=0.8, *;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept any charset but prefers UTF-8 and then iso-8859-1 if UTF-8 is not available.

Read Accept-Charset: (Request) online: https://riptutorial.com/http-headers/topic/10613/accept-charset---request-

Chapter 4: Accept-Encoding: (Request)

Introduction

`Accept-Encoding` tells the server what encoding the client accepts. Encoding is mostly used for compression.

Syntax

- Accept-Encoding: Encoding;QualityFactor
- Accept-Encoding: Encoding;QualityFactor, type;QualityFactor, type;QualityFactor, ...

Parameters

Parameter	Description
Encoding	The type of encoding to use. This can also be a `*` to say the client has no preference to what encoding to use
QualityFactor	The quality factor in the format `;q=0.8`. If this is set to 0 then it means "not acceptable". (optional)

Remarks

`Accept-Encoding` takes a number of encoding and includes an optional preference for which one the server should use. The encoding is one from the list of available encodings at IANA registry. For example `gzip`.

The encoding is separated by commas with an optional quality factor (using a `;q=`) that is used the clients preference for using this encoding. The quality factor has a value from 0 to 1 with the higher the number the more preference for that encoding.

If this header is not included then the client does not state any preference for the encoding. It does not mean that the client supports all encodings.

A value of `identity` is always acceptable unless you reject it with `identity;q=0`.

The server uses `Content-Encoding` to inform the client what encoding it is using.

If the server can't find an acceptable charset to reply with then it should send a 406 (not acceptable) response or ignore this header and not doing any content negotiation.

Examples

Request gzip

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept gzip and identity encoding.

Request gzip and deflate

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: compress, gzip
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept only gzip, compress, and identity encodings.

Request compres but prefer gzip

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip;q=1.0, compress;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept gzip, compress, and identity encoding but prefers gzip (which has a quality factor of 1.0).

No preference for the type of encoding

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: *
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client has not preference for the type of encoding.

Read Accept-Encoding: (Request) online: https://riptutorial.com/http-headers/topic/10615/accept-encoding---request-

Chapter 5: Accept-Language: (Request)

Introduction

`Accept-Language` tells the server what language (such as English) does the client accept.

Syntax

- Accept-Language: Language;QualityFactor
- Accept-Language: Language;QualityFactor, Language;QualityFactor, ...
- Accept-Language: *

Parameters

Parameter	Description
Language	What language is acceptable.
QualityFactor	The quality factor in the format `;q=0.8` (optional)
*	Match any language

Remarks

`Accept-Language` takes a number of languages and includes an optional preference for which one the server should use. The language is one from the list of available at IANA Language Subtag Registry page. For example `en` is English, and `en-US` is USA English.

The language is separated by commas with an optional quality factor (using a `;q=`) that is used the clients preference for using this language. The quality factor has a value from 0 to 1 with the higher the number the more preference for that language.

If this header is not included then the client will accept any language.

The server uses `Content-Language` to inform the client what langauge it is using.

Examples

English only

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
```

```
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html
Accept-Language: en
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will only accept Content-Language of English.

US English or basic english

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept English but prefers US English.

US English or basic english

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html
Accept-Language: da, en-gb;q=0.8, en;q=0.7
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client prefers Danish, but will also accept British English, or if that's not available basic English.

Match any language

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html
Accept-Language: *
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

The client will accept any language.

Read Accept-Language: (Request) online: https://riptutorial.com/http-headers/topic/10616/accept-

language---request-

Chapter 6: Accept-Ranges: (Response)

Introduction

`Accept-Ranges` tells the client that this server will supports ranges for this resource (file).

Syntax

- Accept-Ranges: RangeType
- Accept-Ranges: none

Parameters

Parameter	Description
RangeType	That type of ranges are supported. This is currently only `bytes` or `none`.
none	The server does not support ranges on this resource

Remarks

`Accept-Ranges` is part of the ranges system. The ranges system lets the client request only part of a file instead of having to download the whole file.

For example if a client only needs the last 100 bytes of a 10M file it can request the server only send data from offset 10485660 to 10485760.

`Accept-Ranges` is sent from the server to tell the client if it supports ranges. This only applies to this particular resource (file), other files may accept different range types.

Only two values are currently defined, `bytes` and `none`. The values `bytes` means that you can request byte ranges (offset and end will be in bytes). A value of 'none' means the server does not support ranges.

Clients are free to request byte range requests without checking if the server supports ranges.

The client uses `Range` to request a range from the server and the server replies with a status of 206 (Partial Content) if it is sending the range of bytes or 200 (ok) if it is going to send the whole file.

Examples

Server supports ranges

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Response:

```
HTTP/1.1 200 OK
Date: Sat, 01 Jan 2000 01:00:00 GMT
Server: Apache/2.4.10 (Win32) OpenSSL/1.0.1h PHP/5.4.31
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
Content-Type: text/html
Accept-Ranges: bytes
Content-Length: 500
```

Server doesn't support ranges

Request:"http://example.com"

```
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:53.0) Gecko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

Response:

```
HTTP/1.1 200 OK
Date: Sat, 01 Jan 2000 01:00:00 GMT
Server: Apache/2.4.10 (Win32) OpenSSL/1.0.1h PHP/5.4.31
Keep-Alive: timeout=5, max=97
Connection: Keep-Alive
Content-Type: text/html
Accept-Ranges: none
Content-Length: 500
```

Read Accept-Ranges: (Response) online: https://riptutorial.com/http-headers/topic/10659/accept-ranges---response-

Chapter 7: X-Request-ID

Introduction

The `X-Request-ID` header can be used to trace individual requests to a web service (such as a REST API) from the client to the server and its backends.

Syntax

- X-Request-ID: < value >

Remarks

A Client can send an HTTP header `X-Request-ID: some-value`. The server should use the provided value and provide it in any requests that it makes to backend services for the purpose of serving the initial the request. When sending the response, the server will return the same header back to the client. For the purpose of tracing, the server will include the value into its logs, to enable correlating requests and responses with the corresponding logs.

Examples

nginx

Reverse proxies can detect if a client provides a X-Request-ID header, and pass it on to the backend server. If no such header is provided, it can provide a random value.

```
map $http_x_request_id $reqid {
    default   $http_x_request_id;
    ""        $request_id;
}
```

The code above stores the Request ID in the variable `$reqid` from where it can be subsequently used in logs.

```
log_format trace '$remote_addr - $remote_user [$time_local] "$request" '
                 '$status $body_bytes_sent "$http_referer" "$http_user_agent" '
                 '"$http_x_forwarded_for" $reqid';
```

It should also be passed on to the backend services

```
    location @proxy_to_app {
        proxy_set_header X-Request-ID $reqid;
        proxy_pass   http://backend;
        access_log /var/log/nginx/access_trace.log trace;
    }
```

Heroku

Heroku will always pass on a `X-Request-ID` header send by the client, or generate its own.

See documentation at HTTP Request IDs.

Django

When using Django as a web service framework, the package `django-log-request-id` can be used to parse and log request IDs.

Settings

```
MIDDLEWARE_CLASSES = (
    'log_request_id.middleware.RequestIDMiddleware',
    # ... other middleware goes here
)

LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'filters': {
        'request_id': {
            '()': 'log_request_id.filters.RequestIDFilter'
        }
    },
    'formatters': {
        'standard': {
            'format': '%(levelname)-8s [%(asctime)s] [%(request_id)s] %(name)s: %(message)s'
        },
    },
    'handlers': {
        'console': {
            'level': 'DEBUG',
            'class': 'logging.StreamHandler',
            'filters': ['request_id'],
            'formatter': 'standard',
        },
    },
    'loggers': {
        'myapp': {
            'handlers': ['console'],
            'level': 'DEBUG',
            'propagate': False,
        },
    }
}
```

Request ID (Request / Response)

The same `X-Request-ID` header can be sent by a client in a request, or by a server in a response.

```
X-Request-ID: f9ed4675f1c53513c61a3b3b4e25b4c0
```

The value does not carry any inherent meaning, but is just a token to identify correlating requests

and responses.

Read X-Request-ID online: https://riptutorial.com/http-headers/topic/10581/x-request-id

Credits

S. No	Chapters	Contributors
1	Getting started with http-headers	Community, Stefan Kögl
2	Accept: (Request)	Paul Hutchinson
3	Accept-Charset: (Request)	Paul Hutchinson
4	Accept-Encoding: (Request)	Paul Hutchinson
5	Accept-Language: (Request)	Paul Hutchinson
6	Accept-Ranges: (Response)	Paul Hutchinson
7	X-Request-ID	Stefan Kögl