DZone Refcardz

CONTENTS INCLUDE:

HTTP: The Hypertext Transfer Protocol

By Mick Knutson

ABOUT HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems. HTTP is the foundation of data communication for the World Wide Web.

Hypertext is a multi-linear set of objects, building a network by using logical links (the so-called hyperlinks) between the nodes (e.g. text or words). HTTP is the protocol to exchange or transfer hypertext.

```
RFC 2616 Hypertext Transfer Protocol:
http://www.w3.org/Protocols/rfc2616/rfc2616.html
```

Uniform resource identifier (URI) is a string of characters used to identify a name or a resource.

```
RFC 1630: Universal Resource Identifiers (URI):
http://tools.ietf.org/html/rfc1630
```

Uniform resource name (URN) is a uniform resource identifier (URI) that uses the urn scheme and does not imply availability of the identified resource. Both URN's (names) and URL's (locators) are URI's, and a particular URI may be a name and a locator at the same time.

BASIC SYNTAX:

urn: <Namespace Identifier> : <Namespace Specific String>

EXAMPLE:

```
urn:isbn:9781849683166
```

The URN for 'Java EE6 Cookbook for Securing, Tuning and Extending Enterprise applications.'

```
RFC 1737:
Uniform Resource Names (URN): http://tools.ietf.org/html/rfc1737
```

Uniform resource locator (URL) is a specific character string that constitutes a reference to an Internet resource.

BASIC SYNTAX:

scheme :// domain : port / path ? queryString # fragmentIdentifier

EXAMPLE:

```
http://baselogic.com:80/blog/?param1=value&param2=value#anchor
```

```
RFC 1808:
Relative Uniform Resource Locators (URL): http://tools.ietf.org/html/rfc1808
```

Request Methods

HTTP defines methods (sometimes referred to as "verbs") to indicate the desired action to be performed on the identified resource. What this resource represents, whether pre-existing data or data that is generated dynamically, depends on the implementation of the server. Often, the resource corresponds to a file or the output of an executable residing on the server.

The HTTP/1.0 specification: section 8 defined the GET, POST and HEAD methods and the HTTP/1.1 specification: section 9 added 5 new methods: OPTIONS, PUT, DELETE, TRACE and CONNECT. By being specified in these documents their semantics are known and can be depended upon. Any client can use any method that they want and the server can choose to support any method it wants. If a method is unknown to an intermediate it will be treated as an un-safe and non-idempotent method. There is no limit to the number of methods that can be defined and this allows for future methods to be specified without breaking existing infrastructure. For example WebDAV (RFC5789) defined 7 new methods and RFC5789 specified the PATCH method.

Method	Description
CONNECT	This specification reserves the method name CONNECT for use with a proxy that can dynamically switch to being a tunnel (e.g. SSL tunneling).
DELETE	The DELETE method requests that the origin server delete the resource identified by the Request-URI.
GET	The GET method means retrieves whatever information (in the form of an entity) is identified by the Request-URI.
HEAD	The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.
OPTIONS	The OPTIONS method represents a request for information about the communication options available on the request/response chain identified by the Request-URI.
POST	The POST method is 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.
PUT	The PUT method requests that the enclosed entity be stored under the supplied Request-URI.
TRACE	The TRACE method is used to invoke a remote, application-layer loop-back of the request message.

```
RFC 2616-sec9:
HTTP Method definitions: http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
```

Hot Tip

RFC 2616 (Hypertext Transfer Protocol HTTP/1.1) section 3.2.1: The HTTP protocol does not place any a priori limit on the length of a URI. Servers must be able to handle the URI of any resource they serve, and should be able to handle URIs of unbounded length if they provide GET-based forms that could generate such URIs. A server should return 414 (Request-URI Too Long) status if a URI is longer than the server can handle (see section 10.4.15).

URI Length Limits

Implementation	Limit
Firefox	Unlimited, although instability occurs with URLs reaching around 65,000 characters.
Safari	Unlimited.
Internet Explorer v6 - v7	Maximum length of a URL in Internet Explorer is 2,083 characters, with no more than 2,048 characters in the path portion of the URL.

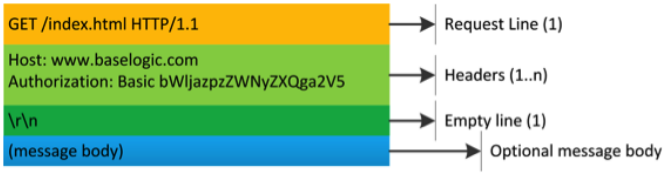
Internet Explorer v8+	Maximum length of a URL in Internet Explorer is 4,095 characters, with no more than 2,048 characters in the path portion of the URL. Maximum mailto: length is 500 to 512 characters long.
Sitemap Protocol	<loc> URL of the page. This URL must begin with the protocol (such as http) and end with a trailing slash, if your web server requires it. This value must be less than 2,048 characters.
GoogleBot crawler	Google will index URLs up to 2047 characters in length.
Google search results page(SERP)	Google will index URLs up to 2047 characters in length.
Google search results page(SERP)	Google index-able links that will work when clicked in the SERP's is ~1855 characters in length.

Hot Tip

RFC 3986: Uniform Resource Identifier (URI) section 3.2.2:
URI producers should use names that conform to the DNS syntax, even when use of DNS is not immediately apparent, and should limit these names to no more than 255 characters in length.

Request Message

The request message consists of the following:



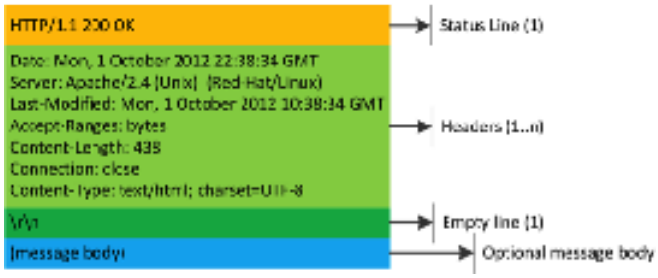
The request line and headers must all end with <CR><LF> (that is, a carriage return character followed by a line feed character (\r\n)). The empty line must consist of only <CR><LF> and no other whitespace. In the HTTP/1.1 protocol, all headers except Host are optional.

A request line containing only the path name is accepted by servers to maintain compatibility with HTTP clients before the HTTP/1.0 specification in RFC1945.

RFC 2616-sec5:
HTTP Request: http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html

Response Message

The response message consists of the following:



The Status-Line and headers must all end with <CR><LF> (a carriage return followed by a line feed). The empty line must consist of only <CR><LF> and no other whitespace.

RFC 2616-sec6:
HTTP Response: http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html

Status Code and Reason Phrase

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata and the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason- Phrase. The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. There are 5 values for the first digit:

Group	Description
Informational 1xx	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.
Successful 2xx	This class of status code indicates that the client's request was successfully received, understood, and accepted.
Redirection 3xx	This class of status code indicates that further action needs to be taken by the user agent in order to fulfill the request.
Client Error 4xx	The 4xx class of status code is intended for cases in which the client seems to have erred.
Server Error 5xx	Response status codes beginning with the digit "5" indicate cases in which the server is aware that it has erred or is incapable of performing the request.

Common Status Codes

Status Code	Description
200	RFC-2616 Section 10.2.1: OK
301	RFC-2616 Section 10.3.2: Moved Permanently
304	RFC-2616 Section 10.3.5: Not Modified
307	RFC-2616 Section 10.3.8: Temporary Redirect
400	RFC-2616 Section 10.4.1: Bad Request
401	RFC-2616 Section 10.4.2: Unauthorized
403	RFC-2616 Section 10.4.4: Forbidden
404	RFC-2616 Section 10.4.5: Not Found
405	RFC-2616 Section 10.4.6: Method Not Allowed
408	RFC-2616 Section 10.4.9: Request Time-out
414	RFC-2616 Section 10.4.15: Request-URI Too Large
500	RFC-2616 Section 10.5.1: Internal Server Error

RFC 2616-sec10 HTTP Status Code Definitions:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

Header Fields

HTTP header fields are components of the message header of requests and responses in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction.

The header fields are transmitted after the request or response line, the first line of a message. Header fields are colon-separated name-value pairs in clear-text string format, terminated by a carriage return (CR) and line feed (LF) character sequence. The end of the header fields is indicated by an empty field, resulting in the transmission of two consecutive CR-LF pairs. Long lines can be folded into multiple lines; continuation lines are indicated by presence of space (SP) or horizontal tab (HT) as first character on the next line. A few fields can also contain comments (i.e. in. User-Agent, Server, Via fields), which can be ignored by software.

Request Headers

Field	Description	Example
Accept	Content-Types that are acceptable.	Accept: text/plain
Accept-Charset	Character sets that are acceptable	Accept-Charset: utf-8
Accept-Encoding	Acceptable encodings. See HTTP compression.	Accept-Encoding: gzip, deflate
Accept-Language	Acceptable languages for response.	Accept-Language: en-US
Accept-Datetime	Acceptable version in time.	Accept-Datetime: Tue, 19 Jun 2012 10:10:10 GMT
Authorization	Authentication credentials for HTTP authentication.	Authorization: Basic bWljazpzZWNyZXQga2V5
Cache-Control	Used to specify directives that MUST be obeyed by all caching mechanisms along the request/response chain.	Cache-Control: no-cache
Connection	What type of connection the user-agent would prefer.	Connection: keep-alive
Cookie	an HTTP cookie previously sent by the server with Set-Cookie header.	Cookie: $Version=1; Skin=new;
Content-Length	The length of the request body in octets (8-bit bytes).	Content-Length: 348
Content-Type	The MIME type of the body of the request (used with POST and PUT requests).	Content-Type: application/x-www-form-urlencoded
Date	The date and time that the message was sent.	Date: Tue, 19 Jun 2012 10:10:10 GMT
Expect	Indicates that particular server behaviors are required by the client.	Expect: 100-continue

Field	Description	Example
From	The email address of the user making the request.	From: user@example.com
Host	The domain name of the server (for virtual hosting), and the TCP port number on which the server is listening. The port number may be omitted if the port is the standard port for the service requested. Mandatory since HTTP/1.1.	Host: baselogic.com:80 Host: baselogic.com
If-Match	Only perform the action if the client supplied entity matches the same entity on the server. This is mainly for methods like PUT to only update a resource if it has not been modified since the user last updated it.	If-Match: "bWljazpzZWNyZXQga2V5"
If-Modified-Since	Allows a 304 Not Modified to be returned if content is unchanged.	If-Modified-Since: Tue, 19 Jun 2012 10:10:10 GMT
If-None-Match	Allows a 304 Not Modified to be returned if content is unchanged.	If-None-Match: "bWljazpzZWNyZXQga2V5"
If-Range	If the entity is unchanged, send me the part(s) that I am missing; otherwise, send me the entire new entity.	If-Range: "bWljazpzZWNyZXQga2V5"
If-Unmodified-Since	Only send the response if the entity has not been modified since a specific time.	If-Unmodified-Since: Tue, 19 Jun 2012 10:10:10 GMT
Max-Forwards	Limit the number of times the message can be forwarded through proxies or gateways.	Max-Forwards: 10
Pragma	Implementation-specific headers that may have various effects anywhere along the request-response chain.	Pragma: no-cache
Proxy-Authorization	Authorization credentials for connecting to a proxy.	Proxy-Authorization: Basic bWljazpzZWNyZXQga2V5
Range	Request only part of an entity. Bytes are numbered from 0.	Range: bytes=500-999
Referer	This is the address of the previous web page from which a link to the currently requested page was followed. (The word "referrer" is misspelled in the RFC as well as in most implementations.)	Referer: http://baselogic.com/
TE	The transfer encodings the user agent is willing to accept: the same values as for the response header TE can be used, plus the "trailers" value (related to the "chunked" transfer method) to notify the server it expects to receive additional headers (the trailers) after the last, zero-sized, chunk.	TE: trailers, deflate
User-Agent	The user agent string of the user agent	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/12.0

Response headers

Field	Description	Example
Access-Control-Allow-Origin	Specifying which web sites can participate in cross-origin resource sharing	Access-Control-Allow-Origin: *
Accept-Ranges	What partial content range types this server supports	Accept-Ranges: bytes
Age	The age the object has been in a proxy cache in seconds	Age: 12
Allow	Valid actions for a specified resource. To be used for a 405 Method not allowed	Allow: GET, HEAD
Cache-Control	Tells all caching mechanisms from server to client whether they may cache this object. It is measured in seconds	Cache-Control: max-age=3600
Connection	Options that are desired for the connection	Connection: close
Content-Encoding	The type of encoding used on the data. See HTTP compression.	Content-Encoding: gzip
Content-Language	The language the content is in.	Content-Language: fr
Content-Length	The length of the response body in octets (8-bit bytes)	Content-Length: 348
Content-Location	An alternate location for the returned data	Content-Location: /index.htm
Content-Range	Where in a full body message this partial message belongs	Content-Range: bytes 21010-47021/47022
Content-Type	The MIME type of this content	Content-Type: text/html; charset=utf-8
Date	The date and time that the message was sent	Date: Tue, 19 Jun 2012 10:10:10 GMT
ETag	An identifier for a specific version of a resource, often a message digest	ETag: "bWljazpzZWNyZXQga2V5"
Expires	Gives the date/time after which the response is considered stale	Expires: Date: Tue, 19 Jun 2012 10:10:10 GMT
Last-Modified	The last modified date for the requested object, in RFC 2822 format	Last-Modified: Date: Tue, 19 Jun 2012 10:10:10 GMT
Link	Used to express a typed relationship with another resource, where the relation type is defined by RFC 5988	Link: </feed>; rel="alternate"
Location	Used in redirection, or when a new resource has been created.	Location: http://www.w3.org/pub/WWW/People.html
Pragma	Implementation-specific headers that may have various effects anywhere along the request-response chain.	Pragma: no-cache
Proxy-Authenticate	Request authentication to access the proxy.	Proxy-Authenticate: Basic
Refresh	Used in redirection, or when a new resource has been created. This refresh redirects after 5 seconds. This is a proprietary, non-standard header extension introduced by Netscape and supported by most web browsers.	Refresh: 5; url=http://baselogic.com/index.html
Retry-After	If an entity is temporarily unavailable, this instructs the client to try again after a specified period of time (seconds).	Retry-After: 120
Server	A name for the server	Server: Apache/2.4 (Unix)
Set-Cookie	Sets an HTTP Cookie	Set-Cookie: UserID=JaneSmith; Max-Age=3600; Version=1
Strict-Transfer-Security	A HSTS Policy informing the HTTP client how long to cache the HTTPS only policy and whether this applies to subdomains.	Strict-Transfer-Security: max-age=16070400; includeSubDomains
Transfer-Encoding	The form of encoding used to safely transfer the entity to the user. Currently defined methods are: chunked, compress, deflate, gzip, identity.	Transfer-Encoding: chunked
Via	Informs the client of proxies through which the response was sent.	Via: 1.0 mick, 1.1 baselogic.com (Apache/2.4)
Warning	A general warning about possible problems with the entity body.	A general warning about possible problems with the entity body.
WWW-Authenticate	Indicates the authentication scheme that should be used to access the requested entity.	WWW-Authenticate: Basic

RFC 2616-sec14:
Header Field Definitions: http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

MIME Types

An Internet media type is a two-part identifier for file formats on the Internet. The identifiers were originally defined in RFC 2046 for use in email sent through SMTP, but their use has expanded to other protocols such as HTTP. These types were called MIME types, and are sometimes referred to as Content-types, after the name of a header in several protocols whose value is such a type.

A media type is composed of two or more parts: A type, a subtype, and zero or more optional parameters. For example, subtypes of text have an optional charset parameter that can be included to indicate the character encoding (e.g. text/html; charset=UTF-8).

Common application MIME Types

Type	Description
application/atom+xml	Atom Feeds
application/ecmascript	ECMAScript/JavaScript; Defined in RFC 4329 (equivalent to application/javascript but with stricter processing rules)
application/EDI-X12	EDI X12 data; Defined in RFC 1767
application/EDIFACT	EDI EDIFACT data; Defined in RFC 1767
application/json	JavaScript Object Notation JSON; Defined in RFC 4627
application/javascript	ECMAScript/JavaScript; Defined in RFC 4329 (equivalent to application/ecmascript but with looser processing rules) It is not accepted in IE 8 or earlier - text/javascript is accepted but it is defined as obsolete in RFC 4329. The "type" attribute of the <script> tag in HTML5 is optional and in practice omitting the media type of JavaScript programs is the most interoperable solution since all browsers have always assumed the correct default even before HTML5.
application/octet-stream	Arbitrary binary data. Generally speaking this type identifies files that are not associated with a specific application. Contrary to past assumptions by software packages such as Apache this is not a type that should be applied to unknown files. In such a case, a server or application should not indicate a content type, as it may be incorrect, but rather, should omit the type in order to allow the recipient to guess the type.
application/ogg	Ogg, a multimedia bitstream container format; Defined in RFC 5334
application/pdf	Portable Document Format, PDF has been in use for document exchange on the Internet since 1993; Defined in RFC 3778
application/postscript	PostScript; Defined in RFC 2046
application/rdf+xml	Resource Description Framework; Defined by RFC 3870
application/rss+xml	RSS feeds
application/soap+xml	SOAP; Defined by RFC 3902
application/font-woff	Web Open Font Format; (candidate recommendation; use application/x-font-woff until standard is official)
application/xhtml+xml	XHTML; Defined by RFC 3236
application/xml-dtd	Document Type Definition (DTD) files; Defined by RFC 3023
application/xop+xml	XML-binary Optimized Packaging (XOP)
application/zip	ZIP archive files; Registered
application/gzip	Gzip, Defined in RFC 6713

Common multipart MIME Types

Type	Description
multipart/mixed	MIME Email; Defined in RFC 2045 and RFC 2046
multipart/alternative	MIME Email; Defined in RFC 2045 and RFC 2046
multipart/related	MIME Email; Defined in RFC 2387 and used by MHTML (HTML mail)
multipart/form-data	MIME Webform; Defined in RFC 2388
multipart/signed	Defined in RFC 1847
multipart/encrypted	Defined in RFC 1847

Common text MIME Types

Type	Description
text/cmd	commands; subtype resident in Gecko browsers like Firefox 3.5
text/css	Cascading Style Sheets; Defined in RFC 2318
text/csv	Comma-separated values; Defined in RFC 4180
text/html	HTML; Defined in RFC 2854
text/javascript (Obsolete)	JavaScript; Defined in and obsoleted by RFC 4329 in order to discourage its usage in favor of application/javascript. However, text/javascript is allowed in HTML 4 and 5 and, unlike application/javascript, has cross-browser support. The "type" attribute of the <script> tag in HTML5 is optional and there is no need to use it at all since all browsers have always assumed the correct default (even in HTML 4 where it was required by the specification).
text/plain	Textual data; Defined in RFC 2046 and RFC 3676
text/vcard	vCard (contact information); Defined in RFC 6350
text/xml	Extensible Markup Language; Defined in RFC 3023

Cookies

A cookie, also known as an HTTP cookie, web cookie, or browser cookie, is usually a small piece of data sent from a website and stored in a user's web browser while a user is browsing a website.

SERVER SETTING COOKIES IN RESPONSE:

```
Set-Cookie: name=value
Set-Cookie: name2=value2; Expires=Tue, 26 Jun 2012 19:19:47 GMT
```

BROWSER SENDING COOKIES TO SERVER IN REQUEST:

```
Cookie: name=value; name2=value2
```

Data URI's

Data URL scheme can be useful for embedding images into HTML/CSS/JS to save on HTTP requests instead of referencing remote files.

STANDARD <IMG> TAG:

```
<img width="99" height="99" alt="BASE Logic, Inc. logo" src="http://baselogic.
com/images/BLiNC_logo.png" />
```

DATA URI IMAGE IN <IMG> TAG:

```
<img width="99" height="99" alt="BASE Logic, Inc. logo" src="
fNwfjZ0frl3/zy7///
```

Avoiding caching

If a web server responds with Cache-Control: no-cache then a web browser or other caching system must not use the response to satisfy subsequent responses without first checking with the originating server. This header field is part of HTTP version 1.1, and is ignored by some caches and browsers. It may be simulated by setting the Expires HTTP version 1.0 header field value to a time earlier than the response time.

The request that a resource should not be cached is no guarantee that it will not be written to disk. In particular, the HTTP/1.1 definition draws a distinction between history stores and caches. If the user navigates back to a previous page a browser may still show you a page that has been stored on disk in the history store. This is correct behavior according to the specification. Many user agents show different behavior in loading pages from the history store or cache depending on whether the protocol is HTTP or HTTPS. The header field Cache-Control: no-store is intended to instruct a browser application to make a best effort not to write it to disk.
The Pragma: no-cache header field is an HTTP/1.0 header intended for use in requests. It is a means for the browser to tell the server and any intermediate caches that it wants a fresh version of the resource, not for the server to tell the browser not to cache the resource. Some user agents do pay attention to this header in responses, but the HTTP/1.1 RFC specifically warns against relying on this behavior.

RFC 2616-sec13: Caching in HTTP:
http://www.w3.org/Protocols/rfc2616/rfc2616-sec13.html

HTTP Conditional GET

A conditional GET is an HTTP GET request that returns an HTTP 304 response (versus HTTP 200). An HTTP 304 response indicates that the resource has not been modified since the previous GET request and the resource will not be returned to the requesting client as part of the response.

BASIC SYNTAX:

```
[Last-Modified / If-Modified-Since]
[ETag / If-None-Match]
```

CLIENT REQUEST:

```
[MickKnutson]$ curl --silent --head --header 'If-Modified-Since: Tue, 19 Jun
2012 10:10:10 GMT' http://baselogic.com/images/BLiNC_logo.png
```

SERVER RESPONSE HTTP 304:

```
HTTP/1.1 304 Not Modified
Date: Tue, 26 Jun 2012 19:13:29 GMT
Server: Apache/2.2.21 (Unix)
Connection: close
Expires: Wed, 27 Jun 2012 19:13:29 GMT
Cache-Control: max-age=86400
```

HTTP Authentication

Transfer Layer Security (TLS / SSL)

Transfer Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communication security over the Internet. TLS and SSL encrypt the segments of network connections at the Application Layer for the Transfer Layer, using asymmetric cryptography for key exchange, symmetric encryption for privacy, and message authentication codes for message integrity.

RFC 5246 Transfer Layer Security (TLS) Protocol v1.2: http://tools.ietf.org/html/rfc5246

Basic access Authentication (BASIC)

In the context of an HTTP transaction, basic access authentication is a method for a web browser or other client program to provide a user name and password when making a request. This is the most basic way of implementing authentication for a web application and is suitable when we are accessing the application both using browser and other software such as scripts and so on. In this mode, when accessed by a browser, the browser will use its standard dialog to collect the credentials. It is easy to implement, but the credentials will be transmitted as plain text and anyone can collect them if we do not have TLS/SSL or some network level encryption in place.

CLIENT REQUEST (NO AUTHENTICATION):

```
GET /secured/index.html HTTP/1.1
Host: localhost
```

SERVER RESPONSE:

```
HTTP/1.1 401 Authorization Required
Server: Apache
Date: Tue, 26 Jun 2012 19:19:47 GMT
WWW-Authenticate: Basic realm="Secure Area"
Content-Type: text/html
Content-Length: 311

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>Error</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
  </HEAD>
  <BODY><H1>401 Unauthorized.</H1></BODY>
</HTML>
```

Before transmitting the username and password entered by the user, the two are concatenated with a colon separating the two values; the resulting string is Base64 encoded. For example, given a username mick and password secret key, the string "mick:secret key" will be encoded with the Base64 algorithm resulting in bWljazpzZWNyZXQga2V5. The Base64-encoded string is transmitted in the HTTP header and decoded by the receiver, resulting in the decoded colon-separated username and password String. Encoding the username and password with the Base64 makes them unreadable visually, but they are easily decoded. Confidentiality is not the intent of the encoding step, rather, the intent is to encode non-HTTP-compatible characters that a username or password may contain, into those that are HTTP-compatible.

CLIENT REQUEST "MICK:SECRET KEY" (USER NAME "MICK", PASSWORD "SECRET KEY"):

```
GET /secured/index.html HTTP/1.1
Host: localhost
Authorization: Basic bWljazpzZWNyZXQga2V5
```

SERVER RESPONSE:

```
HTTP/1.1 200 OK
Server: Apache
Date: Tue, 26 Jun 2012 19:19:47 GMT
Content-Type: text/html
Content-Length: 10476
```

Digest access authentication (DIGEST)

This method is similar to the BASIC authentication method, but instead of the password a digest of the password is transmitted.
Digest communication starts with a client that requests a resource from a web server. If the resource is secured with Digest Authentication, the server will respond with the http status code 401, which means Unauthorized to access this resource.

CLIENT REQUEST (NO AUTHENTICATION):

```
GET /dir/index.html HTTP/1.0
Host: localhost
```

In the response from the initial request, the server indicates in the HTTP header with which mechanism the resource is secured.

SERVER RESPONSE:

```
HTTP/1.0 401 Unauthorized
Server: Apache
Date: Tue, 26 Jun 2012 19:19:47 GMT
WWW-Authenticate: Digest realm="baselogic.com",
         qop="auth",
         nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
         opaque="5ccc069c403ebaf9f0171e9517f40e41"
Content-Type: text/html
Content-Length: 311

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">
<HTML>
  <HEAD>
    <TITLE>Error</TITLE>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=ISO-8859-1">
  </HEAD>
  <BODY><H1>401 Unauthorized.</H1></BODY>
</HTML>
```

CLIENT REQUEST (USERNAME "MICK", PASSWORD "SECRET KEY"):

```
GET /secured/index.html HTTP/1.0
Host: localhost
Authorization: Digest username="mick",
                      realm="baselogic.com",
                      nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                      uri="/secured/index.html",
                      qop=auth,
                      nc=00000001,
                      cnonce="0a4f113b",
                      response="35f308904a9a9623498f358d1cb10afd"
```

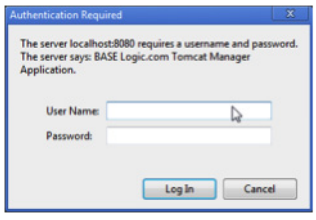
SERVER RESPONSE:

```
HTTP/1.0 200 OK
Server: Apache
Date: Tue, 26 Jun 2012 20:19:47 GMT
Content-Type: text/html
Content-Length: 7984
```

You should notice the term Digest in the response which indicates that the resource requested by the client is secured using Digest Authentication. The server also indicates the type of Digest Authentication algorithm used by the client with Quality Of Protection (QOP) and the nonce string, which is a Base64 encoded timestamp and private hash generated by the server.

```
String nonce = Base64.encode(new Timestamp() : "Private MD5 Hash")
```

The private hash is created by the server, and the Base64 encoding allows for decoding of the timestamp and private hash, even though the private MD5 hash is a one-way encryption.



An internet browser responds to this by presenting the user a dialog, in this dialog the user is able to enter a username and password for credentials. The dialog does not show the warning about transmitting the credentials in clear text as with a Basic Authentication secured site.

The response is generated by several digest properties sent from the client, with the addition to an HA1 and HA2 value concatenated together, then MD5 hash encrypted. The algorithm is the following:

```
Response = MD5( "HA1:\
                 nonce:\
                 nc:\
                 cnonce:\
                 qop:\
                 HA2" )
```

The HA1 hash is the username, realm, and password separated by colons.

```
HA1 = MD5( "mick:baselogic.com:secret key")
```

The MD5 hash for the HA1 is: 935f1e7b1582ffd6e05e7dc8e949ac6f

The HA2 hash is the initial HTTP GET request made to the server.

```
HA2 = MD5( "GET:/secured/index.html" )
```

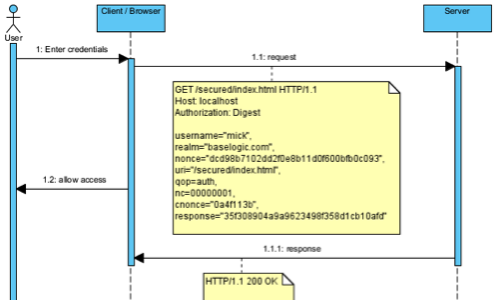
The MD5 hash for the HA2 is: cc21ab6caf04c32228f0250c9eb48705

The final response MD5 hash algorithm would look like this:

```
Response = MD5( "935f1e7b1582ffd6e05e7dc8e949ac6f:\
                 dcd98b7102dd2f0e8b11d0f600bfb0c093:\
                 00000001:0a4f113b:auth:\
                 cc21ab6caf04c32228f0250c9eb48705")
```

This would result in 35f308904a9a9623498f358d1cb10afd which is the value for the response to the client sent to the server for authentication.

Listing: Submitting DIGEST authentication credentials.



Form-Based Authentication (FORM)

In this mode, we can use our own form to collect the username and password. It is very flexible in terms of implementation and how we ask for the username and password, but requires extra work for implementing the forms.This method suffers from the same security risk as the BASIC method because the credentials are transmitted as plain text.

This is however, the most user friendly type of authentication as it allows site owners to control the look-and-feel over the user experience during site navigation.

```
<form method="post" action="/secured/login">
  <input type="text" name="username" required>
  <input type="password" name="password" required>
  <input type="submit" value="Login">
</form>
```

ADDITIONAL RESOURCES

BASE Logic, Inc: http://baselogic.com

Tcp: http://en.wikipedia.org/wiki/Transmission_Control_Protocol

RFC 793 TCP Connection states: http://tools.ietf.org/html/rfc793

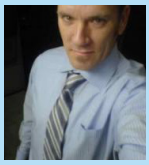
RFC 2397 The "data" URL scheme: http://tools.ietf.org/html/rfc2397

RFC 2459 Internet X.509 Public Key Infrastructure: http://tools.ietf.org/html/rfc2459

RFC 4918: HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV): http://tools.ietf.org/html/rfc4918

cURL: http://en.wikipedia.org/wiki/CURL

ABOUT THE AUTHOR



Mick Knutson, with nearly two decades of experience working in the IT industry in various roles as Enterprise technology consultant, Java Architect, project leader, Engineer, Designer and Developer, has gained a wide variety of experience in disciplines including Java EE, Web Services, Mobile Computing, and Enterprise Integration Solutions. Mr. Knutson has led training courses and book publishing engagements, authored technical white papers, and presented at seminars worldwide. As an active blogger and Tweeter, Mr. Knutson has also been inducted in the prestigious DZone.com "Most Valuable Blogger" (MVB) group, and can be followed at http://baselogic.com, http://dzone.com/users/mickknutson and http://twitter.com/mickknutson.

RECOMMENDED BOOK

Java EE6 Cookbook for securing, tuning, and extending enterprise applications. Java Platform, Enterprise Edition is a widely used platform for enterprise server programming in the Java programming language. This book covers exciting recipes on securing, tuning and extending enterprise applications using a Java EE 6 implementation. Java Platform, Enterprise Edition is a widely used platform for enterprise server programming in the Java programming language.

DZone

DZone communities deliver over 6 million pages each month to more than 3.3 million software developers, architects and decision makers. DZone offers something for everyone, including news, tutorials, cheat sheets, blogs, feature articles, source code and more. "'DZone is a developer's dream'," says PC Magazine.

DZone, Inc.
150 Preston Executive Dr.
Suite 201
Cary, NC 27513

888.678.0399
919.678.0300

Refcardz Feedback Welcome
refcardz@dzone.com

Sponsorship Opportunities
sales@dzone.com

ISBN-13: 978-1-936502-70-7
ISBN-10: 1-936502-70-4
50795
9 781936 502707
$7.95

Version 1.0