

Understanding WWW-Authenticate Header Outline

- [WWW-Authenticate Header](#)
- [WWW-Authenticate Header - Syntax](#)
- [WWW-Authenticate Header – Directives](#)
- [Authentication Schemes](#)
- [Authentication Schemes - Basic](#)
- [Authentication Schemes – Digest – I](#)
- [Authentication Schemes – Digest – II](#)
- [Authentication Schemes – Digest – III](#)
- [Authentication Schemes – HTTP Origin-Bound Authentication \(HOBA\)](#)
- [WWW-Authenticate Header Usage Example with Basic Authentication](#)
- [WWW-Authenticate Header Usage Example with Digest Authentication – I](#)
- [WWW-Authenticate Header Usage Example with Digest Authentication – II](#)
- [WWW-Authenticate Header Usage Example with HOBA Authentication](#)

WWW-Authenticate Header

- The HTTP **WWW-Authenticate** response header defines the **HTTP authentication methods** ("challenges") that might be used to gain access to a specific resource.
- A **server** using **HTTP authentication** will respond with a **401 Unauthorized** response to a request for a **protected resource**.
 - This response must include
 - at least one **WWW-Authenticate** header and
 - at least one **challenge**,to indicate what **authentication schemes** can be used to access the resource (and any additional data that each scheme needs).
- After receiving the WWW-Authenticate header, a **client** will typically prompt the user for credentials, and then **re-request** the resource.
 - This **new request** uses the **Authorization** header to supply the credentials to the server, encoded appropriately for the selected "challenge" authentication method.
 - The client is expected to select the most secure of the challenges it understands.

WWW-Authenticate Header - Syntax

- At least one **challenge** must be specified.
- **Multiple** challenges may be specified, comma-separated, in a **single** header, or in **individual** headers.
- In the **single challenge**, the **scheme token**, **<auth-scheme>**, is mandatory.
- The presence of **realm**, **token68** and any other parameters depends on the definition of the selected scheme.

```
// Challenges specified in single header
WWW-Authenticate: challenge1, ..., challengeN

// Challenges specified in multiple headers
WWW-Authenticate: challenge1
...
WWW-Authenticate: challengeN
```

```
// Possible challenge formats (scheme dependent)
WWW-Authenticate: <auth-scheme>
WWW-Authenticate: <auth-scheme> realm=<realm>
WWW-Authenticate: <auth-scheme> token68
WWW-Authenticate: <auth-scheme> auth-param1=token1, ..., auth-paramN=auth-paramN-token
WWW-Authenticate: <auth-scheme> realm=<realm> token68
WWW-Authenticate: <auth-scheme> realm=<realm> token68 auth-param1=auth-param1-token , ..., auth-paramN=auth-paramN-token
WWW-Authenticate: <auth-scheme> realm=<realm> auth-param1=auth-param1-token, ..., auth-paramN=auth-paramN-token
WWW-Authenticate: <auth-scheme> token68 auth-param1=auth-param1-token, ..., auth-paramN=auth-paramN-token
```



WWW-Authenticate Header - Directives

- the **<auth-scheme>** token. —————> • represents **Authentication Scheme**.
 - A **string** describing a **protected area**.
 - A realm allows a **server** to partition up the areas it protects (if supported by a scheme that allows such partitioning).
- **realm=<realm>** —————> • Some clients show this value to the user to inform them about which particular credentials are required — ***though most browsers stopped doing so to counter phishing.***
 - The only reliably supported character set for this value is **us-ascii**.
 - If **no realm is specified**, clients often display a **formatted hostname** instead.
- **<token68>** —————> • A **token** that may be useful for some schemes.
 - The token allows the **66 unreserved URI characters** plus a few others.
 - According to the specification, it can hold a **base64**, **base64url**, **base32**, or **base16** (hex) encoding, with or without padding, but excluding whitespace.




Authentication Schemes

- Expressed with the **<auth-scheme>** token.
- More **common types** are
 - Basic,
 - Digest,
 - Negotiate and
 - AWS4-HMAC-SHA256.

Authentication Schemes - Basic

- <realm>  • Note that the realm is **mandatory** for [basic authentication](#).
- charset="UTF-8"  • Tells the [client](#) the [server's preferred encoding scheme](#) when submitting a username and password.
- The **only allowed value** is the case-insensitive string "**UTF-8**".
- This does not relate to the encoding of the realm string.

Authentication Schemes – Digest - I

- <realm> 
 - String indicating which **username/password** to use.
 - Minimally should include the **host name** but
 - might indicate the **users** or **group** that have access.
- domain 
 - A quoted, space-separated list of **URI prefixes** that define **all the locations** where the authentication information may be used.
 - If this key is not specified, then the authentication information may be used anywhere on the web root.
 - A **server-specified** quoted string that the **server** can use to control the **lifetime** in which credentials will be considered valid.
- nonce 
 - This must be uniquely generated **each time a 401 response is made**, and may be regenerated more often (for example, allowing a digest to be used only once).
 - The specification contains advice on possible algorithms for generating this value.
 - The nonce value is **opaque** to the **client**.

Authentication Schemes – Digest - II

- opaque

- A **server-specified** quoted string that should be returned unchanged in the [Authorization](#).
- This is **opaque** to the **client**.
- The **server** is recommended to include [Base64](#) or [hexadecimal data](#).

- algorithm

- Algorithm used to produce the digest.
- Valid **non-session** values are:
 - "**MD5**" (default if not specified), "[SHA-256](#)", "[SHA-512](#)".
- Valid **session** values are:
 - "[MD5-sess](#)", "[SHA-256-sess](#)", "[SHA-512-sess](#)".



- stale

- A case-insensitive **flag** indicating that the previous request from the **client** was rejected because the [nonce](#) used is too old (stale).
- If this is [true](#), the request can be re-tried using the **same** username/password encrypted using the **new nonce**.
- If it is any other value, then the username/password are invalid and must be re-requested from the user.

Authentication Schemes – Digest - III

- qop
 - - Quoted string indicating the **quality of protection** supported by the **server**.
 - This must be supplied, and unrecognized options must be ignored.
 - **"auth"**: Authentication
 - **"auth-int"**: Authentication with integrity protection.
- charset="UTF-8"
 - - Tells the **client** the **server's preferred encoding scheme** when submitting a username and password.
 - The **only allowed value** is the case-insensitive string **"UTF-8"**.
- userhash
 - - A **server** may specify **"true"** to indicate that it supports **username hashing**.
 - default is **"false"**.

Authentication Schemes – HTTP Origin-Bound Authentication (HOBA)

- <challenge> 
 - A set of pairs in the format of '<len>:<value>' concatenated together to be given to a **client**.
 - The **challenge** is made of up a
 - **nonce**,
 - **algorithm**,
 - **origin**,
 - **realm**,
 - **key identifier**, and
 - the **challenge**.
 - The **number of seconds** from the time the **HTTP response** is emitted for which responses to this **challenge** can be accepted.
- <max-age> 
- realm

WWW-Authenticate Header Usage Example with Basic Authentication

- A **server** that only supports **basic authentication** might have a **WWW-Authenticate** response header which looks like this:

→ `WWW-Authenticate: Basic realm="Access to the staging site", charset="UTF-8"`

- A **user-agent** receiving this header would
 - first prompt the user for their username and password and then
 - re-request the resource: this time including the **encoded credentials** in the **Authorization** header.

The **Authorization** header might look like this:

→ `Authorization: Basic YWxhZGRpbjpvGVuc2VzYW1l`

- For **Basic authentication**, the **credentials** are constructed
 - by first combining the username and the password with a colon (**aladdin:opensesame**), and then
 - by **encoding** the resulting string in **base64** (**YWxhZGRpbjpvGVuc2VzYW1l**).

WWW-Authenticate Header Usage Example with Digest Authentication - I

- The **client attempts to access** a document at URI <http://www.example.org/dir/index.html> that is **protected** via **digest authentication**.
 - The username for this document is "**Mufasa**" and the password is "**Circle of Life**" (note the single space between each of the words).
- The first time the **client requests** the document, no **Authorization** header field is sent.
- Here the **server responds** with an **HTTP 401 message** that includes a **challenge** for each digest algorithm it supports, in its order of preference, **SHA256** and then **MD5**.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
    realm="http-auth@example.org",
    qop="auth, auth-int",
    algorithm=SHA-256,
    nonce="7ypf/xlj9XXwfDPEoM4URrv/xwf94BcCAzFZH4GiTo0v",
    opaque="FQhe/qaU925kfnzjCev0ciny7QMkPqMAFRtzCUYo5tdS"
WWW-Authenticate: Digest
    realm="http-auth@example.org",
    qop="auth, auth-int",
    algorithm=MD5,
    nonce="7ypf/xlj9XXwfDPEoM4URrv/xwf94BcCAzFZH4GiTo0v",
    opaque="FQhe/qaU925kfnzjCev0ciny7QMkPqMAFRtzCUYo5tdS"
```

WWW-Authenticate Header Usage Example with Digest Authentication - II

- The **client prompts** the user for their username and password, and then **responds** with a new **request** that encodes the credentials in the **Authorization** header field.
- If the **client** chose the **MD5** digest, the **Authorization** header field might look like:
- If the **client** chose the **SHA-256** digest, the **Authorization** header field might look like:

```
Authorization: Digest username="Mufasa",  
    realm="http-auth@example.org",  
    uri="/dir/index.html",  
    algorithm=MD5,  
    nonce="7ypf/xlj9XXwfdPEoM4URrv/xwf94BcCAzFZH4GiTo0v",  
    nc=00000001,  
    cnonce="f2/wE4q74E6zIJEtWaHKaf5wv/H5QzzpXusqGemxURZJ",  
    qop=auth,  
    response="8ca523f5e9506fed4657c9700eebdbec",  
    opaque="FQhe/qaU925kfnzjCev0ciny7QMkPqMAFRtzCUYo5tdS"
```

```
Authorization: Digest username="Mufasa",  
    realm="http-auth@example.org",  
    uri="/dir/index.html",  
    algorithm=SHA-256,  
    nonce="7ypf/xlj9XXwfdPEoM4URrv/xwf94BcCAzFZH4GiTo0v",  
    nc=00000001,  
    cnonce="f2/wE4q74E6zIJEtWaHKaf5wv/H5QzzpXusqGemxURZJ",  
    qop=auth,  
    response="753927fa0e85d155564e2e272a28d1802ca10daf449  
        6794697cf8db5856cb6c1",  
    opaque="FQhe/qaU925kfnzjCev0ciny7QMkPqMAFRtzCUYo5tdS"
```

WWW-Authenticate Header Usage Example with HOBA Authentication

- A **server** that supports **HOBA authentication** might have a **WWW-Authenticate** response header which looks like this:

```
WWW-Authenticate: HOBA max-age="180",  
challenge="16:MTEyMzEyMzEyMw==1:028:https://www.example.com:80800:3:MTI48:NjgxNDdjOTctNDYxYi00MzEwLWJlOWItNGM3MDcyMzdYjUz"
```

- The **to-be-signed blob challenge** is made from these parts:

- www.example.com using port 8080,
- the **nonce** is '1123123123',

- the **algorithm** for signing is RSA-SHA256,
- the **key identifier** is 123, and finally
- the **challenge** is '68147c97-461b-4310-be9b-4c707237ab53'.

- A **client** would **receive** this header,

- **extract** the **challenge**,
- **sign it** with their **private key** that corresponds to **key identifier** 123 in our example using RSA-SHA256, and then
- send the result in the **Authorization** header as a dot-separated key **id, challenge, nonce**, and **signature**.

```
Authorization:  
123.16:MTEyMzEyMzEyMw==1:028:https://www.example.com:80800:3:MTI48:NjgxNDdjOTctNDYxYi00MzEwLWJlOWItNGM3MDcyMzdYjUz.1123123123.<signature-of-challenge>
```