HTTPAUTH Working Group
Internet-Draft
Intended status: Experimental
Expires: January 2, 2014

Y. Oiwa
H. Watanabe
H. Takagi
RISEC, AIST
B. Kihara
T. Hayashi
Lepidum
Y. Ioku
Yahoo! Japan
July 1, 2013

# HTTP Authentication Extensions for Interactive Clients
# draft-ietf-httpauth-extension-00

## Abstract

This document specifies a few extensions of HTTP authentication framework for interactive clients. Recently, fundamental features of HTTP-level authentication is not enough for complex requirements of various Web-based applications. This makes these applications to implement their own authentication frameworks using HTML Forms and other means, which becomes one of the hurdles against introducing secure authentication mechanisms handled jointly by servers and user-agent clients. The extended framework fills gaps between Web application requirements and HTTP authentication provisions to solve the above problems, while maintaining compatibility against existing Web and non-Web uses of HTTP authentications.

## Status of this Memo

## Copyright Notice



## Table of Contents

# 1. Introduction

The document proposes several extensions to the current HTTP authentication framework, to provide enough functionality comparable with current widely-used form-based Web authentication. A majority of the recent Web-sites on the Internet use custom application-layer authentication implementations using Web forms. The reasons for these may vary, but many people believe that the current HTTP Basic (and Digest, too) authentication method does not have enough functionality (including a good-feeling user interfaces) to support most of realistic Web-based applications. However, the method is very weak against phishing and other attacks, because the whole behavior of the authentication is controlled from the server-side applications. This makes it really hard to implement any cryptographically strong authentication mechanisms into Web systems. To overcome this problem, we need to "modernize" the HTTP authentication framework so that better client-controlled secure methods can be used with Web applications. The extensions proposed in this document include:

- non-mandatory, optional authentication on HTTP (Section 3),
- log out from both server and client side (Section 4), and
- finer control for redirection depending on authentication status (Section 4).

[I-D note: These extensions are initially proposed as a part of [I-D.ietf-httpauth-mutual]. However, since these functionalities might possibly be useful in combination even with other authentication schemes, the extensions were separated from the original document as this independent draft.]

## 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The terms "encouraged" and "advised" are used for suggestions that do not constitute "SHOULD"-level requirements. People MAY freely choose not to include the suggested items regarding [RFC2119], but complying with those suggestions would be a best practice; it will improve the security, interoperability, and/or operational performance.

This document distinguishes the terms "client" and "user" in the following way: A "client" is an entity understanding and talking HTTP and the specified authentication protocol, usually computer software; a "user" is a (usually natural) person who wants to access data resources using "a client".

# 2. Definitions

## 2.1. Terms for describing authentication protocol flow

HTTP Authentication defined in [I-D.ietf-httpbis-p7-auth] may involve with several pairs of HTTP requests/responses. Throughout this document, the following terms are used to categorize those messages: for requests,

- A non-authenticating request is a request not attempting any authentication: a request without any Authorization header.
- An authenticating request is the opposite: a request with an Authorization header.

For responses,

1) A non-authenticated response:

   is a response which does not involve with any HTTP authentication. It may not contain any WWW-Authenticate or Authentication-Info header.

   Servers send this response when the requested resource is not protected by HTTP authentication mechanisms. In context of this specification, not-authentication-related negative responses (e.g. 403 and 404) are also considered as non-authenticated responses.

   (See note on successfully-authenticated responses below for some ambiguous cases.)

2) An authentication-initializing response:

   is a response which requires or allows clients to start authentication attempts. Servers send this response when the requested resource is protected by HTTP authentication mechanism, and the request meets one of the following cases:

   - The request is non-authenticating request, or
   - The request contained an authentication trial directed to the protection space (realm) other than the server's expected one.

   The server will specify the protection space for authentication in this response.

   Upon reception, the client's behavior is further divided to two possible cases.

   - If the client may have no prior knowledge on authentication credentials (e.g. a user-name and a password) related to the requested protection space, the protocol flow terminates and the client will ask the user to provide authentication credentials,
   - On the other hand, if client already have an enough credentials for authentication to the requested protection space, the client will automatically send an authenticating request. Such cases often occur when the client did not know beforehand that the current request-URL requires an authentication.

3) A successfully-authenticated response:

   is a response for an authenticating request meaning that the authentication attempt was granted. (Note: if the authentication scheme used does not use an Authentication-Info header, it may be indistinguishable from a non-authenticated response.)

4) An intermediate authenticating response:

   is a response for an authenticating request which requires some more reaction by the client software without involving users. Such a response is required when an authentication scheme requires two or more round-trip messages to perform authentication, or when an authentication scheme uses some speculative short-cut method (such as uses of cached shared secrets) and it failed.

5) A negatively-authenticated response:

   is a response for an authenticating request which means that the authentication attempt was declined and can not continue without another authentication credential. Clients typically erase memory of the currently-using credentials and ask the user for other ones.

   Usually the format of these responses are as same as the one for authentication-initializing responses. Client can distinguish it by comparing the protection spaces contained in the request and in the response.

Figure 1 shows a state diagram of generic HTTP authentication with the above message categorization. Note that many authentication schemes uses only a subset of the transitions described on the diagram. Labels in the figure show the abbreviated names of response types.

---

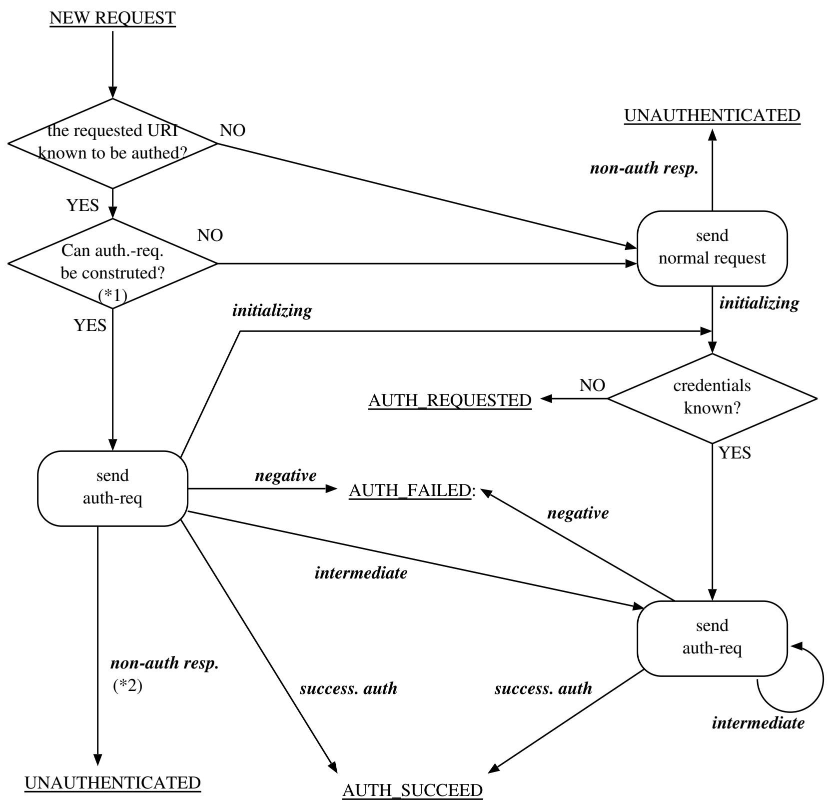


**Figure 1: Generic state diagram for HTTP authentication**

---

Note: (*1) For example, "Digest" scheme requires server-provided nonces to construct client-side challenges.
(*2) In "Basic" and some others, this cannot be distinguished from a successfully-authenticated response.

## 2.2. Syntax Notation

This specification uses an extended BNF syntax defined in [I-D.ietf-httpbis-p1-messaging]. The following syntax definitions are quoted from [I-D.ietf-httpbis-p1-messaging] and [I-D.ietf-httpbis-p7-auth]: auth-scheme, quoted-string, auth-param, SP, header-field, and challenge. It also uses the convention of using header names for specifying syntax of header values.

Additionally, this specification uses the following syntax elements following syntax definitions as a refinement for token and the righthand-side of auth-param in [I-D.ietf-httpbis-p7-auth]. (Note: these definitions are consistent with those in [I-D.ietf-httpauth-mutual].)

---

```
bare-token        = 1*(%x30-39 / %x41-5A / %x61-7A / "-" / "_")
extension-token   = "-" bare-token 1*("." bare-token)
extensive-token   = bare-token / extension-token
integer           = "0" / (%x31-39 *%x30-39)      ; no leading zeros
```

**Figure 2: the BNF syntax for common notations**

Extensive-tokens are used in this protocol where the set of acceptable tokens may include private extensions. Any private extensions of this protocol MUST use the extension-tokens with format "-<token>.<domain-name>", where <domain-name> is a validly registered (sub-)domain name on the Internet owned by the party who defines the extensions.

# 3. Optional Authentication

The Optional-WWW-Authenticate header enables a non-mandatory authentication, which is not possible under the current HTTP authentication mechanism. In several Web applications, users can access the same contents as both a guest user and an authenticated user. In most Web applications, it is implemented using HTTP cookies [RFC6265] and custom form-based authentications. The new authentication method using this message will provide a replacement for these authentication systems.

Servers MAY send HTTP successful responses (response code 200, 206 and others) containing the Optional-WWW-Authenticate header as a replacement of a 401 response when it is an authentication-initializing response. The Optional-WWW-Authenticate header MUST NOT be contained in 401 responses.

```
HTTP/1.1 200 OK
Optional-WWW-Authenticate: Basic realm="xxxx"
```

---

```
Optional-WWW-Authenticate = challenge
```

**Figure 3: BNF syntax for Optional-WWW-Authenticate header**

---

The challenge contained in the Optional-WWW-Authenticate header are the same as those for a 401 responses corresponding for a same request. For authentication-related matters, an optional authentication request will have the same meaning as a 401 message with a corresponding WWW-Authenticate header (as an authentication-initializing response). (The behavior for other matters, such as caching, MAY be different between the optional authentication and 401 messages.)

A response with an Optional-WWW-Authenticate header SHOULD be returned from the server only when the request is either non-authenticated or authenticating to a wrong (not the server’s expected) protection space. If a response is either an intermediate or a negative response to a client’s authentication attempt, the server MUST respond with a 401 status response with a WWW-Authenticate header instead. Failure to comply this rule will make client not able to distinguish authentication successes and failures.

The server is NOT RECOMMENDED to include an Optional-WWW-Authenticate header in a positive response when a client’s authentication attempt succeeds.

Whenever an authentication scheme support for servers to send some parameter which gives a hint of URL space for the corresponding protection space for the same realm (e.g. "path" or "domain"), servers requesting non-mandatory authentication SHOULD send such parameter with the response. Clients supporting non-mandatory authentication MUST recognize the parameter, and MUST send a request with an appropriate authentication credential in an Authorization header for any URI inside the specified paths.

Support of this header is OPTIONAL; Clients MAY also choose any set of authentication schemes for which optional authentication is supported (in other words, its support MAY be scheme-dependent). However, some authentication schemes MAY require mandatory/recommended support for this header, so that server-side applications MAY assume that clients supporting such schemes are likely to support the extension as well.

# 4. Authentication-Control header

```
Authentication-Control = auth-scheme 1*SP 1#auth-param
```

**Figure 4: the BNF syntax for the Authentication-Control header**

The Authentication-Control header provides a more precise control of the client behavior for Web applications using an HTTP authentication protocol. This header is supposed to be generated in the application layer, as opposed to WWW-Authenticate headers which will be generated usually by the Web servers.

Support of this header is OPTIONAL, and clients MAY choose any subset of these parameters to be supported. The set of supported parameters MAY also be authentication scheme-dependent. However, some authentication schemes MAY require mandatory/recommended support for some or all of the features provided in this header.

The "auth-scheme" specified in this header and other authentication-related headers within the same message MUST be the same. If there are no authentication currently performed, and the auth-scheme contained in this header is not recognizable for the client, the whole header SHOULD be ignored.

The header contain one or more parameters, each of which is a name-value pair. The name of each parameter MUST be an extensive-token. Clients MUST ignore any unknown parameters contained in this header.

The type of parameter value depends on the parameter name as defined in the following subsections. Regardless of the type, however, the recipients SHOULD accept both quoted and unquoted representations of values as defined in HTTP. If it is defined as a string, it is encouraged to be sent in a quoted-string form. If it defined as a token (or similar) or an integer, the value SHOULD follow the corresponding ABNF syntax after possible unquoting of the quoted-string value (as defined in HTTP), and is encouraged to be sent in a unquoted form.

Server-side application SHOULD always be reminded that any parameters contained in this header MAY be ignored by clients. Also, even when a client accepts this header, users may always be able to circumvent semantics of this header. Therefore, if this header is used for security purposes, its use MUST be limited for providing some non-fundamental additional security measures valuable for end-users (such as client-side log-out for protecting against console takeover). Server-side application MUST NOT rely on the use of this header for protecting server-side resources.

## 4.1. Auth-style parameter

Authentication-Control: Digest auth-style=modal

The parameter "auth-style" specifies the server's preferences over user interface behavior for user authentication. This parameter can be included in any kind of responses, however, it is only meaningful for either authentication-initializing or negatively-authenticated responses. The value of this parameter MUST be one of the bare-tokens "modal" or "non-modal". When the Optional-WWW-Authenticate header is used, the value of this parameter MUST be disregarded and the value "non-modal" is implied.

The value "modal" means that the server thinks the content of the response (body and other content-related headers) is valuable only for users refusing authentication request. The clients are expected to ask the user a password before processing the content. This behavior is common for most of the current implementations of Basic and Digest authentication schemes.

The value "non-modal" means that the server thinks the content of the response (body and other content-related headers) is valuable for users before processing an authentication request. The clients are expected to first process the content and then provide users opportunities to perform authentication.

The default behavior for the clients is implementation-dependent, and clients MAY choose different defaults for different authentication schemes. The proposed default behavior is "modal" for all authentication schemes, but specifications for authentication schemes MAY propose a different default.

The above two different methods of authentication may introduce a observable difference of semantics when the response contains state-changing side effects; for example, it may change whether Cookie headers [RFC6265] in 401 responses are processed or not. However, the server applications SHOULD NOT depend on both existence and non-existence of such side effects.

## 4.2. Location-when-unauthenticated parameter

Authentication-Control: Mutual
location-when-unauthenticated="http://www.example.com/login.html"

The parameter "location-when-unauthenticated" specifies a location where any unauthenticated clients should be redirected to. This header may be used, for example, when there is a central login page for the entire Web application. The value of this parameter MUST be a string that contains an absolute URL location. If a given URL is not absolute, the clients MAY consider it a relative URL from the current location.

This parameter MAY be used with a 401 response for authentication-initializing response. It can also be contained, although NOT RECOMMENDED, in a positive response with an Optional-WWW-Authenticate header. The clients MUST ignore this parameter, when a response is either successfully-authenticated or intermediately-authenticated. The clients SHOULD ignore this parameter when a response is a negatively-authenticated one (the case is unlikely to happen, though).

When a client receives an authentication-initiating response with this parameter, if the client has to ask users for authentication credentials, the client will treat the entire response as if it were a 303 "See Other" response with a Location header that contains the value of this parameter (i.e., client will be redirected to the specified location with a GET request). Unlike a normal 303 response, if the client can process authentication without the user's interaction, this parameter MUST be ignored.

### 4.3. No-auth parameter

Authentication-Control: Basic no-auth=true

The parameter "no-auth" is a variant of the location-when-unauthenticated parameter; it specifies that new authentication attempt is not to be performed on this location for better user experience, without specifying the redirection on the HTTP level. This header may be used, for example, when there is a central login page for the entire Web application, and when a (Web content's level) explicit interaction of users is desired before authentications. The value of this parameter MUST be a token "true". If the value is incorrect, client MAY ignore this parameter.

This parameter MAY be used with authentication-initiating responses. It can also be contained, although NOT RECOMMENDED, in a positive response with an Optional-WWW-Authenticate header. The clients MUST ignore this parameter, when a response is either successfully-authenticated or intermediately-authenticated. The clients SHOULD ignore this parameter when a response is a negatively-authenticated one (the case is unlikely to happen, though).

When a client receives an authentication-initiating response with this parameter, if the client has to ask users for authentication credentials, the client will ignore the WWW-Authenticate header contained in the response and treat the whole response as a normal negative 4xx-class response instead of giving user an opportunity to start authentication. If the client can process authentication without the user's interaction, this parameter MUST ignored.

This parameter SHOULD NOT be used along with the location-when-unauthenticated parameter. If both were supplied, clients MAY choose which one is to be honored.

This parameter SHOULD NOT be used as any security measures to prevent authentication attempts, as it is easily circumvented by users. This parameter SHOULD be used solely for improving user experience of web applications.

### 4.4. Location-when-logout parameter

Authentication-Control: Digest location-when-logout="http://www.example.com/byebye.html"

The parameter "location-when-logout" specifies a location where the client is to be redirected when the user explicitly request a logout. The value of this parameter MUST be a string that contains an absolute URL location. If a given URL is not absolute, the clients MAY consider it a relative URL from the current location.

This parameter MAY be used with successfully-authenticated responses. If this parameter is contained in other kinds of responses, the clients MUST ignore this parameter.

When the user requests to terminate an authentication period, and if the client currently displays a page supplied by a response with this parameter, the client will be redirected to the specified location by a new GET request (as if it received a 303 response). The log-out operation (e.g. erasing memories of user name, authentication credential and all related one-time credentials such as nonce or keys) SHOULD occur before processing a redirection.

When the user requests to terminate an authentication period, if the client supports this parameter but the server response does not contain this parameter, the client's RECOMMENDED behavior is as follows: if the request corresponding to the current content was idempotent (e.g. GET), reload the page without the authentication credential. If the request was non-idempotent (e.g. POST), keep the current content as-is and simply forget the authentication status. The client SHOULD NOT replay a non-idempotent request without the user's explicit approval.

Web applications are encouraged to send this parameter with an appropriate value for any responses (except those with redirection (3XX) statuses) for non-GET requests.

### 4.5. Logout-timeout

Authentication-Control: Basic logout-timeout=300

The parameter "logout-timeout", when contained in a successfully-authenticated response, means that any authentication credentials and states related to the current protection space are to be discarded if a time specified in this header (in seconds) has been passed from the time received. The value MUST be an integer. As a special case, the value 0 means that the client is requested to immediately log-out from the current authentication space and revert to an unauthenticated status. This does not, however, mean that the long-term memories for the passwords (such as the password reminders and auto fill-ins) should be removed. If a new timeout value is received for the same authentication space, it cancels the previous timeout and sets a new timeout.

## 5. Usage examples (informative)

This section shows some examples for applying this extension to typical Web-sites which are using Forms and cookies for managing authentication and authorization. The content of this section is not normative and for illustrative purposes only.

We assume that all features described in the previous sections are implemented in clients (Web browsers). We also assume that browsers will have a user interface which allows users to deactivate (log-out from) current authentication sessions. If this assumption is not hold, texts below provides another approach with de-authentication pages used instead of such a UI.

Without explicit notices, all settings described below are to be applied with Authentication-Control headers, and these can be sent to clients regardless of authentication statuses (these will be silently ignored whenever not effective).

### 5.1. Example 1: a portal site

This subsection provides an example application for a site whose structure is somewhat similar to conventional portal sites. In particular, most of web pages are available for guest (unauthenticated) users, and if authentication is performed, contents of these pages are customized for each user. We assume the site has the following kinds of pages currently:

- Content pages.
- Pages/mechanism for performing authentication:
  - There is one page which asks a user name and a password using a HTML POST form.
  - After the authentication attempt, the user will be redirected to either the page which is previously displayed before the authentication, or some specific page.
- A de-authentication (log-out) page.

### 5.1.1. Case 1: a simple application

When such a site does not need a specific actions upon log-in and log-out, the following simple settings can be used.

- Set up an optional authentication to all pages available to guests. Set up an Authentication-Control header with "auth-style=non-modal" setting.
- If there are pages only available to authenticated users, Set up a mandatory authentication with "auth-style=non-modal" setting.
- No specific pages for authentication is needed. It will be performed automatically, directed by the above setting.
- A de-authentication page is also not needed. If the site will have one, put "logout-timeout=0" there.
- For all pages for POST requests, it is advisable to have "location-when-logout=<some page>".

### 5.1.2. Case 2: specific action required on log-out

If the site needs a specific actions upon log-out, the following settings can be used.

- All shown in the Case 1 are to be applied.
- For all pages, set up the Authentication-Control header "location-when-logout=<de-authentication page>".
- In de-authentication pages, no specific set-up is needed. If there is any direct links to it, put "logout-timeout=0".

### 5.1.3. Case 3: specific page displayed before log-in

If the site needs to display a specific page before log-in actions (some announces, user notices, or even advertisements), the following settings can be applied.

- Set up an optional authentication to all pages available to guest. Set up an Authentication-Control header with "no-auth=true". Put a link to a specific log-in page in contents.
- If there are pages only available to authenticated users, Set up a mandatory authentication with "location-when-unauthenticated=<the log-in page>".
- For the specific log-in page, Set up a mandatory authentication.
- For all pages for POST requests, it is advisable to have "location-when-logout=<some page>", too.
- De-authentication pages are not needed. If the site will have one, put "logout-timeout=0".

## 5.2. Example 2: authenticated user-only sites

If almost all pages in the target site requires authentication (e.g., an Internet banking site), or there are no needs to support both unauthenticated and authenticated users on the same resource, the setting will become somewhat simple. The following are an example to realize such a site:

- Set up a mandatory authentication to all pages available to authenticated. Set up an Authentication-Control header with "auth-style=non-modal" setting.
- Set up a handler for the 401-status which requests users to authenticate.
- For all pages for POST requests, it is advisable to have "location-when-logout=<some page>", too.
- De-authentication pages are not needed. If the site will have one, put "logout-timeout=0" there.

## 5.3. When to use Cookies

In the current Web sites using Form-based authentications, Cookies [RFC6265] are used for managing both authorization and application sessions. Using the extensions in this document, the former features will be provided by using (extended) HTTP authentication/authorization mechanisms. In some cases, there will be some ambiguous situations whether some functions are authorization management or session management. The following hints will be helpful for deciding which features to be used.

- If there is a need to serve multiple sessions for a single user using multiple browsers concurrently, use a Cookie for distinguishing between sessions for the same user. (C.f. if there is a need to distinguish sessions in the same browser, HTML5 Web Storage [W3C.CR-webstorage-20111208] features may be used instead of Cookies.)
- If a web site is currently deploying a session time-out feature, consider who benefits from the feature. In most cases, the main requirement for such feature is to protect users from their consoles and browsers hijacked (i.e. benefits are on the users' side). In such cases, the time-out features provided in this extension may be used. On the other hand, the requirements is to protect server's privilege (e.g. when some regulations require to limit the time difference between user's two-factor authentication and financial transaction commitment; the requirement is strictly on the servers' side), that should be managed on the server side using Cookies or other session management mechanisms.

## 5.4. Parallel deployment with Form/Cookie authentications

In some transition periods, sites may need to support both HTTP-layer and Form-based authentications. The following example shows one way to achieve that.

- If Cookies are used even for HTTP-authenticated users, each session determined by Cookies should identify which authentication are used for the session.
- First, set up any of the above settings for enabling HTTP-layer authentication.
- For unauthenticated users, put the following things to the Web pages, unless the client supports this extension and HTTP-level authentication.
  - For non-mandatory authenticated pages, put a link to Form-based authenticated pages.
  - For mandatory authenticated pages, either put a link to Form-based authenticated pages, or put a HTML-level redirection (using META element) to such pages.
- In Form-based authenticated pages, if users are not authenticated, it may have a diversion for HTTP-level authentication by "location-when-unauthenticated" setting.
- Users are identified for authorizations and content customizations by the following logic.
  - First, check the result of the HTTP-level authentication. If there is a Cookie session tied to a specific user, both ones should match.
  - If the user is not authenticated on the HTTP-level, use the conventional Form-based method to determine the user.
  - If there is a Cookie tied to an HTTP authentication, but there is no corresponding HTTP authentication result, that session will be discarded (because it means that authentication is deactivated by the corresponding user).

# 6. Methods to extend this protocol

If a private extension to this protocol is implemented, it MUST use the extension-param to avoid conflicts with this protocol and other future official extensions.

Extension-tokens MAY be freely used for any non-standard, private, and/or experimental uses. The extension-tokens MUST be with format "-<bare-token>.<domain-name>", where <domain-name> is a validly registered (sub-)domain name on the Internet owned by the party who defines the extensions. Unknown parameter names are to be ignored regardless of whether it is extension-tokens or bare-tokens.

# 7. IANA Considerations

Tokens used for the authentication control parameters may be either extension-tokens or bare-tokens as outlined in Section 2.2. When bare-tokens are used in this protocol, these MUST be allocated by IANA. Any tokens used for non-private, non-experimental parameters are RECOMMENDED to be registered to IANA, regardless of the kind of tokens used.

To acquire registered tokens, a specification for the use of such tokens MUST be available as a publicly-accessible documents, as outlined as "Specification Required" level in [RFC5226].

Note: More formal declarations will be added in the future drafts to meet the RFC 5226 requirements.

# 8. Security Considerations

The purpose of the log-out timeout feature in the Authentication-control header is to protect users of clients from impersonation caused by an attacker having access to the same console. Server application implementors SHOULD be aware that the directive may always be ignored by either malicious clients or clients not supporting this extension. If the purpose of introducing a timeout for an authentication period is to protect server-side resources, such features MUST be implemented by other means such as HTTP Cookies [RFC6265].

All parameters in Authentication-Control header SHOULD NOT be used for any security-enforcement purposes. Server-side applications MUST be implemented always considering that the header may be either ignored by clients or even bypassed by users.

# 9. References

## 9.1. Normative References


[I-D.ietf-httpbis-p1-messaging] Fielding, R. and J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing,” draft-ietf-httpbis-p1-messaging-22 (work in progress), February 2013 (TXT).

[I-D.ietf-httpbis-p7-auth] Fielding, R. and J. Reschke, “Hypertext Transfer Protocol (HTTP/1.1): Authentication,” draft-ietf-httpbis-p7-auth-22 (work in progress), February 2013 (TXT).

| | |
|---|---|
| [RFC2119] | Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML). |
| [RFC5226] | Narten, T. and H. Alvestrand, “Guidelines for Writing an IANA Considerations Section in RFCs,” BCP 26, RFC 5226, May 2008 (TXT). |


### 9.2. Informative References


| | |
|---|---|
| [I-D.ietf-httpauth-mutual] | Oiwa, Y., Watanabe, H., Takagi, H., Kihara, B., Hayashi, T., and Y. Ioku, “Mutual Authentication Protocol for HTTP,” draft-ietf-httpauth-mutual-00 (work in progress), July 2013. |
| [RFC6265] | Barth, A., “HTTP State Management Mechanism,” RFC 6265, April 2011 (TXT). |
| [W3C.CR-webstorage-20111208] | Hickson, I., “Web Storage,” World Wide Web Consortium CR CR-webstorage-20111208, December 2011 (HTML). |


## Appendix A. (Informative) Applicability of features for each messages

This section provides cross-reference table about applicability of each features provided in this specification for each kinds of responses described in Section 2.1. The table provided in this section is for informative purposes only.

| | init. | success. | intermed. | neg. |
|---|---|---|---|---|
| Optional auth. | O | n | N | N |
| auth-style | O | - | - | O |
| loc.-when-unauth. | O | I | I | i |
| no-auth | O | I | I | i |
| loc.-when-logout | - | O | - | - |
| logout-timeout | - | O | - | - |

Legends:
O = MAY contain; n = SHOULD NOT contain; N = MUST NOT contain
i = SHOULD be ignored; I = MUST be ignored;
- = meaningless (to be ignored)

## Appendix B. (Informative) Draft Notes

Things which might be considered for future revisions:

- In [I-D.ietf-httpbis-p7-auth], meaning of WWW-Authenticate headers in non-401 responses are defined as "supplying credentials (or different credentials) might affect the response". This clarification change leaves a way for using 200-status responses along with a WWW-Authenticate header for providing optional authentication.

Incorporating this possibility, however, needs more detailed analysis on the behavior of existing clients and intermediate proxies for such possibly-confusing responses. Optional-WWW-Authenticate is safer, at least for minimum backward compatibility, because clients not supporting this extension will consider this header as an unrecognized entity-header, possibly providing opportunity for silently falling-back to application-level authentications.

# Appendix C. (Informative) Draft Change Log

## C.1. Changes in Httpauth revision 00

None.

## C.2. Changes in HttpBis revision 00

None.

## C.3. Changes in revision 02

- Added usage examples.

## C.4. Changes in revision 01

- Syntax notations and parsing semantics changed to match httpbis style.

## C.5. Changes in revision 00

- Separated from HTTP Mutual authentication proposal (-09).
- Adopting httpbis works as a referencing point to HTTP.
- Generalized, now applicable for all HTTP authentication schemes.
- Added "no-auth" and "auth-style" parameters.
- Loosened standardization requirements for parameter-name tokens registration.

# Authors' Addresses

Yutaka Oiwa
National Institute of Advanced Industrial Science and Technology
Research Institute for Secure Systems
3-11-46 Nakouji
Amagasaki, Hyogo
JP
Email: mutual-auth-contact-ml@aist.go.jp

Hajime Watanabe
National Institute of Advanced Industrial Science and Technology
Research Institute for Secure Systems
Tsukuba Central 2
1-1-1 Umezono
Tsukuba-shi, Ibaraki

JP

Hiromitsu Takagi
National Institute of Advanced Industrial Science and Technology
Research Institute for Secure Systems
Tsukuba Central 2
1-1-1 Umezono
Tsukuba-shi, Ibaraki
JP

Boku Kihara
Lepidum Co. Ltd.
#602, Village Sasazuka 3
1-30-3 Sasazuka
Shibuya-ku, Tokyo
JP

Tatsuya Hayashi
Lepidum Co. Ltd.
#602, Village Sasazuka 3
1-30-3 Sasazuka
Shibuya-ku, Tokyo
JP

Yuichi Ioku
Yahoo! Japan, Inc.
Midtown Tower
9-7-1 Akasaka
Minato-ku, Tokyo
JP