HTTP Client Hints

Abstract

   HTTP defines proactive content negotiation to allow servers to select
   the appropriate response for a given request, based upon the user
   agent's characteristics, as expressed in request headers. In
   practice, user agents are often unwilling to send those request
   headers, because it is not clear whether they will be used, and
   sending them impacts both performance and privacy.

   This document defines an Accept-CH response header that servers can
   use to advertise their use of request headers for proactive content
   negotiation, along with a set of guidelines for the creation of such
   headers, colloquially known as "Client Hints."

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

There are thousands of different devices accessing the web, each with
different device capabilities and preference information. These
device capabilities include hardware and software characteristics, as
well as dynamic user and user agent preferences. Historically,
applications that wanted the server to optimize content delivery and
user experience based on such capabilities had to rely on passive
identification (e.g., by matching the User-Agent header field
(Section 5.5.3 of [RFC7231]) against an established database of user
agent signatures), use HTTP cookies [RFC6265] and URL parameters, or
use some combination of these and similar mechanisms to enable ad hoc
content negotiation.

Such techniques are expensive to set up and maintain and are not
portable across both applications and servers. They also make it
hard for both user agent and server to understand which data are
required and are in use during the negotiation:

* User agent detection cannot reliably identify all static
   variables, cannot infer dynamic user agent preferences, requires
   an external device database, is not cache friendly, and is reliant
   on a passive fingerprinting surface.

* Cookie-based approaches are not portable across applications and
      servers, impose additional client-side latency by requiring
      JavaScript execution, and are not cache friendly.
   * URL parameters, similar to cookie-based approaches, suffer from
      lack of portability and are hard to deploy due to a requirement to
      encode content negotiation data inside of the URL of each
      resource.

   Proactive content negotiation (Section 3.4.1 of [RFC7231]) offers an
   alternative approach; user agents use specified, well-defined request
   headers to advertise their capabilities and characteristics, so that
   servers can select (or formulate) an appropriate response based on
   those request headers (or on other, implicit characteristics).

   However, traditional proactive content negotiation techniques often
   mean that user agents send these request headers prolifically. This
   causes performance concerns (because it creates "bloat" in requests),
   as well as privacy issues; passively providing such information
   allows servers to silently fingerprint the user.

   This document defines Client Hints, a framework that enables servers
   to opt-in to specific proactive content negotiation features,
   adapting their content accordingly, as well as guidelines for content
   negotiation mechanisms that use the framework. This document also
   defines a new response header, Accept-CH, that allows an origin
   server to explicitly ask that user agents send these headers in
   requests.

   Client Hints mitigate performance concerns by assuring that user
   agents will only send the request headers when they're actually going
   to be used, and they mitigate privacy concerns of passive
   fingerprinting by requiring explicit opt-in and disclosure of
   required headers by the server through the use of the Accept-CH
   response header, turning passive fingerprinting vectors into active
   ones.

   The document does not define specific usages of Client Hints. Such
   usages need to be defined in their respective specifications.

   One example of such usage is the User-Agent Client Hints [UA-CH].

1.1. Notational Conventions

   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
   BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all
   capitals, as shown here.

   This document uses the Augmented Backus-Naur Form (ABNF) notation of
   [RFC5234].

2. Client Hints Request Header Fields

   A Client Hints request header field is an HTTP header field that is
   used by HTTP user agents to indicate data that can be used by the
   server to select an appropriate response. Each one conveys user-
   agent preferences that the server can use to adapt and optimize the
   response.

2.1. Sending Client Hints

   User agents choose what Client Hints to send in a request based on
   their default settings, user configuration, and server preferences
   expressed in "Accept-CH". The user agent and server can use an opt-
   in mechanism outlined below to negotiate which header fields need to
   be sent to allow for efficient content adaption, and they can
   optionally use additional mechanisms (e.g., as outlined in
   [CLIENT-HINTS-INFRASTRUCTURE]) to negotiate delegation policies that
   control access of third parties to those same header fields. User
   agents SHOULD require an opt-in to send any hints that are not
   considered low-entropy. See the low-entropy hint table at
   [CLIENT-HINTS-INFRASTRUCTURE] for examples of hints that expose low
   amounts of entropy.

   Implementers need to be aware of the fingerprinting implications when
   implementing support for Client Hints and follow the considerations
   outlined in the Security Considerations section of this document (see
   Section 4).

2.2. Server Processing of Client Hints

   When presented with a request that contains one or more Client Hints
   header fields, servers can optimize the response based upon the
   information in them. When doing so, and if the resource is
   cacheable, the server MUST also generate a Vary response header field
   (Section 7.1.4 of [RFC7231]) to indicate which hints can affect the
   selected response and whether the selected response is appropriate
   for a later request.

   Servers MUST ignore hints they do not understand nor support. There
   is no mechanism for servers to indicate to user agents that hints
   were ignored.

   Furthermore, the server can generate additional response header
   fields (as specified by the hint or hints in use) that convey related
   values to aid client processing.

3. Advertising Server Support

   Servers can advertise support for Client Hints using the mechanism
   described below.

3.1. The Accept-CH Response Header Field

The Accept-CH response header field indicates server support for the hints indicated in its value. Servers wishing to receive user agent information through Client Hints SHOULD add the Accept-CH response header to their responses as early as possible.

Accept-CH is a Structured Header [RFC8941]. Its value MUST be an sf-list (Section 3.1 of [RFC8941]) whose members are Tokens (Section 3.3.4 of [RFC8941]). Its ABNF is:

```
Accept-CH = sf-list
```

For example:

```
Accept-CH: Sec-CH-Example, Sec-CH-Example-2
```

When a user agent receives an HTTP response containing "Accept-CH", it indicates that the origin opts-in to receive the indicated request header fields for subsequent same-origin requests. The opt-in MUST be ignored if delivered over non-secure transport (using a scheme different from HTTPS). It SHOULD be persisted and bound to the origin to enable delivery of Client Hints on subsequent requests to the server's origin, for the duration of the user's session (as defined by the user agent). An opt-in overrides previous persisted opt-in values and SHOULD be persisted in its stead.

Based on the Accept-CH example above, which is received in response to a user agent navigating to "https://site.example", and delivered over a secure transport, persisted Accept-CH preferences will be bound to "https://site.example". It will then use it for navigations to for example, "https://site.example/foobar.html", but not to, for example, "https://foobar.site.example/". It will similarly use the preference for any same-origin resource requests (e.g., to "https://site.example/image.jpg") initiated by the page constructed from the navigation's response, but not to cross-origin resource requests (e.g., "https://thirdparty.example/resource.js"). This preference will not extend to resource requests initiated to "https://site.example" from other origins (e.g., from navigations to "https://other.example/").

3.2. Interaction with Caches

When selecting a response based on one or more Client Hints, and if the resource is cacheable, the server needs to generate a Vary response header field [RFC7234] to indicate which hints can affect the selected response and whether the selected response is appropriate for a later request.

```
Vary: Sec-CH-Example
```

The above example indicates that the cache key needs to include the
Sec-CH-Example header field.

Vary: Sec-CH-Example, Sec-CH-Example-2

The above example indicates that the cache key needs to include the
Sec-CH-Example and Sec-CH-Example-2 header fields.

4. Security Considerations

4.1. Information Exposure

Request header fields used in features relying on this document
expose information about the user's environment to enable privacy-
preserving proactive content negotiation and avoid exposing passive
fingerprinting vectors. However, implementers need to bear in mind
that in the worst case, uncontrolled and unmonitored active
fingerprinting is not better than passive fingerprinting. In order
to provide user privacy benefits, user agents need to apply further
policies that prevent abuse of the information exposed by features
using Client Hints.

The information exposed by features might reveal new information
about the user, and implementers ought to consider the following
considerations, recommendations, and best practices.

The underlying assumption is that exposing information about the user
as a request header is equivalent (from a security perspective) to
exposing this information by other means. (For example, if the
request's origin can access that information using JavaScript APIs
and transmit it to its servers.)

Because Client Hints is an explicit opt-in mechanism, it means that
servers wanting access to information about the user's environment
need to actively ask for it, enabling clients and privacy researchers
to keep track of which origins collect that data, and potentially act
upon it. The header-based opt-in means that removal of passive
fingerprinting vectors is possible. As an example, the user agent
can reduce the information exposed by the User-Agent string, while
enabling active access to that information through User-Agent Client
Hints [UA-CH]. Otherwise, the user agent can expose information
already available through script (e.g., the Save-Data Client Hints
<https://wicg.github.io/savedata/#save-data-request-header-field>),
without increasing the passive fingerprinting surface. User agents
supporting Client Hints features which send certain information to
opted-in servers SHOULD avoid sending the equivalent information
passively.

Therefore, features relying on this document to define Client Hint
headers MUST NOT provide new information that is otherwise not made

available to the application by the user agent, such as existing
request headers, HTML, CSS, or JavaScript.

Such features need to take into account the following aspects of the
exposed information:

Entropy: Exposing highly granular data can be used to help identify
   users across multiple requests to different origins. Reducing the
   set of header field values that can be expressed, or restricting
   them to an enumerated range where the advertised value is close to
   but is not an exact representation of the current value, can
   improve privacy and reduce risk of linkability by ensuring that
   the same value is sent by multiple users.

Sensitivity: The feature SHOULD NOT expose user-sensitive
   information. To that end, information available to the
   application, but gated behind specific user actions (e.g., a
   permission prompt or user activation), SHOULD NOT be exposed as a
   Client Hint.

Change over time: The feature SHOULD NOT expose user information
   that changes over time, unless the state change itself is also
   exposed (e.g., through JavaScript callbacks).

Different features will be positioned in different points in the
space between low-entropy, non-sensitive, and static information
(e.g., user agent information) and high-entropy, sensitive, and
dynamic information (e.g., geolocation). User agents need to
consider the value provided by a particular feature vs. these
considerations and may wish to have different policies regarding that
tradeoff on a per-feature or other fine-grained basis.

Implementers ought to consider both user- and server-controlled
mechanisms and policies to control which Client Hints header fields
are advertised:

* Implementers SHOULD restrict delivery of some or all Client Hints
   header fields to the opt-in origin only, unless the opt-in origin
   has explicitly delegated permission to another origin to request
   Client Hints header fields.

* Implementers that consider providing user-choice mechanisms that
   allow users to balance privacy concerns against bandwidth
   limitations need to also consider that explaining the privacy
   implications involved to users, such as the risks of passive
   fingerprinting, may be challenging or even impractical.

* Implementations specific to certain use cases or threat models MAY
   avoid transmitting some or all of the Client Hints header fields.
   For example, avoid transmission of header fields that can carry
   higher risks of linkability.

User agents MUST clear persisted opt-in preferences when any one of
site data, browsing cache, cookies, or similar are cleared.

4.2. Deployment and Security Risks

Deployment of new request headers requires several considerations:

*   Potential conflicts due to existing use of a header field name
*   Properties of the data communicated in a header field value

Authors of new Client Hints are advised to carefully consider whether
they need to be able to be added by client-side content (e.g.,
scripts) or whether the Client Hints need to be exclusively set by
the user agent. In the latter case, the Sec- prefix on the header
field name has the effect of preventing scripts and other application
content from setting them in user agents. Using the "Sec-" prefix
signals to servers that the user agent -- and not application content
-- generated the values. See [FETCH] for more information.

By convention, request headers that are Client Hints are encouraged
to use a CH- prefix, to make them easier to identify as using this
framework; for example, CH-Foo or, with a "Sec-" prefix, Sec-CH-Foo.
Doing so makes them easier to identify programmatically (e.g., for
stripping unrecognized hints from requests by privacy filters).

A Client Hints request header negotiated using the Accept-CH opt-in
mechanism MUST have a field name that matches sf-token (Section 3.3.4
of [RFC8941]).

4.3. Abuse Detection

A user agent that tracks access to active fingerprinting information
SHOULD consider emission of Client Hints headers similar to the way
it would consider access to the equivalent API.

Research into abuse of Client Hints might look at how HTTP responses
to requests that contain Client Hints differ from those with
different values and from those without values. This might be used
to reveal which Client Hints are in use, allowing researchers to
further analyze that use.

5. Cost of Sending Hints

Sending Client Hints to the server incurs an increase in request byte
size. Some of this increase can be mitigated by HTTP header
compression schemes, but each new hint sent will still lead to some
increased bandwidth usage. Servers SHOULD take that into account
when opting in to receive Client Hints and SHOULD NOT opt-in to
receive hints unless they are to be used for content adaptation
purposes.

Due to request byte size increase, features relying on this document
to define Client Hints MAY consider restricting sending those hints
to certain request destinations [FETCH], where they are more likely
to be useful.

6. IANA Considerations

Features relying on this document are expected to register added
request header fields in the "Permanent Message Header Field Names"
registry [RFC3864].

This document defines the "Accept-CH" HTTP response header field;
IANA has registered it in the same registry.

6.1. Accept-CH

Header field name: Accept-CH

Applicable protocol: HTTP

Status: experimental

Author/Change controller: IETF

Specification document(s): Section 3.1 of this RFC

Related information: for Client Hints

7. References

7.1. Normative References

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119,
               DOI 10.17487/RFC2119, March 1997,
               <https://www.rfc-editor.org/info/rfc2119>.

   [RFC3864]   Klyne, G., Nottingham, M., and J. Mogul, "Registration
               Procedures for Message Header Fields", BCP 90, RFC 3864,
               DOI 10.17487/RFC3864, September 2004,
               <https://www.rfc-editor.org/info/rfc3864>.

   [RFC5234]   Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
               Specifications: ABNF", STD 68, RFC 5234,
               DOI 10.17487/RFC5234, January 2008,
               <https://www.rfc-editor.org/info/rfc5234>.

   [RFC7231]   Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
               Protocol (HTTP/1.1): Semantics and Content", RFC 7231,
               DOI 10.17487/RFC7231, June 2014,

                    <https://www.rfc-editor.org/info/rfc7231>.

   [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke,
              Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching",
              RFC 7234, DOI 10.17487/RFC7234, June 2014,
              <https://www.rfc-editor.org/info/rfc7234>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

   [RFC8941] Nottingham, M. and P-H. Kamp, "Structured Field Values for
              HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021,
              <https://www.rfc-editor.org/info/rfc8941>.

7.2. Informative References

   [CLIENT-HINTS-INFRASTRUCTURE]
              Weiss, Y., "Client Hints Infrastructure", July 2020,
              <https://wicg.github.io/client-hints-infrastructure/>.

   [FETCH]   WHATWG, "Fetch - Living Standard",
              <https://fetch.spec.whatwg.org/>.

   [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265,
              DOI 10.17487/RFC6265, April 2011,
              <https://www.rfc-editor.org/info/rfc6265>.

   [UA-CH]   West, M. and Y. Weiss, "User-Agent Client Hints", August
              2020, <https://wicg.github.io/ua-client-hints/>.

Acknowledgements

Authors' Addresses

   Ilya Grigorik
   Google

   Email: ilya@igvita.com
   URI: https://www.igvita.com/

   Yoav Weiss
   Google

Email: yoav@yoav.ws
URI: https://blog.yoav.ws/