HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2020

I. Grigorik
Y. Weiss
Google

March 11, 2020

HTTP Client Hints
draft-ietf-httpbis-client-hints-12

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, clients 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."

Note to Readers

Discussion of this draft takes place on the HTTP working group mailing list (ietf-http-wg@w3.org), which is archived at https://lists.w3.org/Archives/Public/ietf-http-wg/ [1].

Working Group information can be found at http://httpwg.github.io/ [2]; source code and issues list for this draft can be found at https://github.com/httpwg/http-extensions/labels/client-hints [3].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2020.

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 client preferences. Historically, applications that wanted to allow the server to optimize content delivery and user experience based on such capabilities had to rely on passive identification (e.g., by matching User-Agent (Section 5.5.3 of [RFC7231]) header field against an established database of client signatures), used 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 setup and maintain, and are not portable across both applications and servers. They also make it hard for both client and server to reason about which data is required and is in use during the negotiation:

- User agent detection cannot reliably identify all static variables, cannot infer dynamic client preferences, requires 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.

However, traditional proactive content negotiation techniques often mean that clients 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 agent.

This document defines a new response header, Accept-CH, that allows an origin server to explicitly ask that clients send these headers in requests. It also defines guidelines for content negotiation mechanisms that use it, colloquially referred to as Client Hints.

Client Hints mitigate performance concerns by assuring that clients will only send the request headers when they're actually going to be used, and 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.

This document defines Client Hints, a framework that enables servers to opt-in to specific proactive content negotiation features, adapting their content accordingly. However, it does not define any specific features that will use that infrastructure. Those features will be defined in their respective specifications.

One example of such a feature is the User Agent Client Hints feature [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 Hint Request Header Fields

A Client Hint request header field is a HTTP header field that is used by HTTP clients to indicate data that can be used by the server to select an appropriate response. Each one conveys client preferences that the server can use to adapt and optimize the response.

2.1. Sending Client Hints

Clients choose what Client Hints to send in a request based on their default settings, user configuration, and server preferences expressed in "Accept-CH". The client 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 optionally use additional mechanisms to negotiate delegation policies that control access of third parties to same header fields.

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

2.2. Server Processing of Client Hints

When presented with a request that contains one or more client hint 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.

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.

Accept-CH is a Structured Header [I-D.ietf-httpbis-header-structure]. Its value MUST be an sh-list (Section 3.1 of [I-D.ietf-httpbis-header-structure]) whose members are tokens (Section 3.3.4 of [I-D.ietf-httpbis-header-structure]). Its ABNF is:

```
Accept-CH = sh-list
```

For example:

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

When a client 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 or for an origin with 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 example:

```
Accept-CH: Sec-CH-Example, Sec-CH-Example-2
Accept-CH: Sec-CH-Example-3
```

Based on the Accept-CH example above, which is received in response to a user agent navigating to "https://example.com", and delivered over a secure transport: a user agent will have to persist an Accept-CH preference bound to "https://example.com" and use it for user agent navigations to "https://example.com" and any same-origin resource requests initiated by the page constructed from the navigation's response. This preference will not extend to resource requests initiated to "https://example.com" from other origins.

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
```

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

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

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 proactive content negotiation. Such information 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 to the capability of that request's origin to access that information by other means and transmit it to itself.

Therefore, features relying on this document to define Client Hint headers MUST NOT provide new information that is otherwise not available to the application via other means, such as existing request headers, HTML, CSS, or JavaScript.

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

- 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 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 SHOULD consider the value provided by a particular feature vs these considerations, and MAY have different policies regarding that tradeoff on a per-feature 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 MAY provide user choice mechanisms so that users can balance privacy concerns with bandwidth limitations. However, implementers SHOULD also be aware that explaining the privacy implications of passive fingerprinting to users can be challenging.
- Implementations specific to certain use cases or threat models MAY avoid transmitting some or all of Client Hints header fields. For

example, avoid transmission of header fields that can carry higher risks of linkability.

Implementers SHOULD support Client Hints opt-in mechanisms and MUST clear persisted opt-in preferences when any one of site data, browsing history, 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 header field name
- Properties of the data communicated in 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 they 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 unrecognised hints from requests by privacy filters).

4.3. Abuse Detection

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

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

5. Cost of Sending Hints

While HTTP header compression schemes reduce the cost of adding HTTP header fields, sending Client Hints to the server incurs an increase in request byte size. 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

This document defines the "Accept-CH" HTTP response header field, and registers it in the Permanent Message Header Fields registry.

6.1. Accept-CH

- Header field name: Accept-CH
- Applicable protocol: HTTP
- Status: standard
- Author/Change controller: IETF
- Specification document(s): Section 3.1 of this document
- Related information: for Client Hints

7. References

7.1. Normative References

[FETCH] van Kesteren, A., "Fetch", n.d., <https://fetch.spec.whatwg.org/>.

[I-D.ietf-httpbis-header-structure] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", draft-ietf-httpbis-header-structure-16 (work in progress), March 2020.

[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>.

[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>.

7.2. Informative References

[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", n.d., <https://wicg.github.io/ua-client-hints/>.

7.3. URIs

[1] https://lists.w3.org/Archives/Public/ietf-http-wg/

[2] http://httpwg.github.io/

[3] https://github.com/httpwg/http-extensions/labels/client-hints

Appendix A. Changes

A.1. Since -00

- Issue 168 (make Save-Data extensible) updated ABNF.
- Issue 163 (CH review feedback) editorial feedback from httpwg list.
- Issue 153 (NetInfo API citation) added normative reference.

A.2. Since -01

- Issue 200: Moved Key reference to informative.
- Issue 215: Extended passive fingerprinting and mitigation considerations.
- Changed document status to experimental.

A.3. Since -02

- Issue 239: Updated reference to CR-css-values-3
- Issue 240: Updated reference for Network Information API
- Issue 241: Consistency in IANA considerations
- Issue 250: Clarified Accept-CH

A.4. Since -03

- Issue 284: Extended guidance for Accept-CH
- Issue 308: Editorial cleanup
- Issue 306: Define Accept-CH-Lifetime

A.5. Since -04

- Issue 361: Removed Downlink
- Issue 361: Moved Key to appendix, plus other editorial feedback

A.6. Since -05

- Issue 372: Scoped CH opt-in and delivery to secure transports
- Issue 373: Bind CH opt-in to origin

A.7. Since -06

- Issue 524: Save-Data is now defined by NetInfo spec, dropping
- PR 775: Removed specific features to be defined in other specifications

A.8. Since -07

- Issue 761: Clarified that the defined headers are response headers.
- Issue 730: Replaced Key reference with Variants.
- Issue 700: Replaced ABNF with structured headers.
- PR 878: Removed Accept-CH-Lifetime based on feedback at IETF 105

A.9. Since -08

- PR 985: Describe the bytesize cost of hints.
- PR 776: Add Sec- and CH- prefix considerations.
- PR 1001: Clear CH persistence when cookies are cleared.

A.10. Since -09

- PR 1064: Fix merge issues with "cost of sending hints".

A.11. Since -10

- PR 1072: LC feedback from Julian Reschke.
- PR 1080: Improve list style.
- PR 1082: Remove section mentioning Variants.
- PR 1097: Editorial feedback from mnot.
- PR 1131: Remove unused references.
- PR 1132: Remove nested list.

A.12. Since -11

- PR 1134: Re-insert back section.

Acknowledgements

Thanks to Mark Nottingham, Julian Reschke, Chris Bentzel, Ben Greenstein, Tarun Bansal, Roy Fielding, Vasiliy Faronov, Ted Hardie, Jonas Sicking, Martin Thomson, and numerous other members of the IETF HTTP Working Group for invaluable help and feedback.

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/