HTTP Working Group
Internet-Draft
Intended status: Experimental
Expires: September 12, 2019

I. Grigorik
Google

March 11, 2019

HTTP Client Hints
draft-ietf-httpbis-client-hints-07

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 two response headers, Accept-CH and Accept-CH-Lifetime, 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."

It also defines an initial set of 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/.

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

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, 2019.

Table of Contents

1. Introduction 3
1.1. Notational Conventions 4
2. Client Hint Request Header Fields 4
2.1. Sending Client Hints 4
2.2. Server Processing of Client Hints 5
2.2.1. Advertising Support via Accept-CH Header Field . . . 5
2.2.2. The Accept-CH-Lifetime Header Field 5
2.2.3. Interaction with Caches 6
3. Security Considerations 7
4. IANA Considerations 7
4.1. Accept-CH 8
4.2. Accept-CH-Lifetime 8
5. References 8
5.1. Normative References 8
5.2. Informative References 9
5.3. URIs 9
Appendix A. Interaction with Key Response Header Field 9
Appendix B. Changes 10
B.1. Since -00 10
B.2. Since -01 10
B.3. Since -02 10
B.4. Since -03 10
B.5. Since -04 10
B.6. Since -05 10
B.7. Since -06 10
B.8. Since -07 11

Acknowledgements 11
Author's Address 11

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.

One way to infer some of these capabilities is through User-Agent (Section 5.5.3 of [RFC7231]) header field detection against an established database of client signatures. However, this technique requires acquiring such a database, integrating it into the serving path, and keeping it up to date. However, even once this infrastructure is deployed, user agent sniffing has numerous limitations:

- User agent detection cannot reliably identify all static variables
- User agent detection cannot infer any dynamic client preferences
- User agent detection requires an external device database
- User agent detection is not cache friendly

A popular alternative strategy is to use HTTP cookies ([RFC6265]) to communicate some information about the user agent. However, this approach is also not cache friendly, bound by same origin policy, and often imposes additional client-side latency by requiring JavaScript execution to create and manage HTTP cookies.

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, proactive content negotiation requires clients to 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, for a period of time bounded by the Accept-CH-Lifetime response header. It also defines guidelines for content negotiation mechanisms that use it, colloquially referred to as Client Hints.

Client Hints mitigate the performance concerns by assuring that clients will only send the request headers when they're actually

going to be used, and the 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 the Client Hints infrastructure, a framework that enables servers to opt-in to specific proactive content negotiation features, which will enable them to adapt 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.

This document does not supersede or replace the User-Agent header field. Existing device detection mechanisms can continue to use both mechanisms if necessary. By advertising user agent capabilities within a request header field, Client Hints allow for cache friendly and proactive content negotiation.

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] with the list rule extension defined in [RFC7230], Appendix B. It includes by reference the DIGIT rule from [RFC5234] and the OWS and field-name rules from [RFC7230].

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 configuration 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 control which Client Hints are sent in requests, based on their default settings, user configuration, and server preferences. The client and server can use an opt-in mechanism outlined below to negotiate which fields should 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 fields.

Implementers should be aware of the passive fingerprinting implications when implementing support for Client Hints, and follow the considerations outlined in "Security Considerations" 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.

Further, depending on the hint used, the server can generate additional response header fields to convey related values to aid client processing.

2.2.1. Advertising Support via Accept-CH Header Field

Servers can advertise support for Client Hints using the Accept-CH header field or an equivalent HTML meta element with http-equiv attribute ([HTML5]).

```
Accept-CH = #field-name
```

For example:

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

When a client receives an HTTP response advertising support for Client Hints, it should process it as origin ([RFC6454]) opt-in to receive Client Hint header fields advertised in the field-value. The opt-in MUST be delivered over a secure transport.

For example, based on Accept-CH example above, a user agent could append the Sec-CH-Example and Sec-CH-Example-2 header fields to all same-origin resource requests initiated by the page constructed from the response.

2.2.2. The Accept-CH-Lifetime Header Field

Servers can ask the client to remember the set of Client Hints that the server supports for a specified period of time, to enable delivery of Client Hints on subsequent requests to the server's origin ([RFC6454]).

```
Accept-CH-Lifetime = #delta-seconds
```

When a client receives an HTTP response that contains Accept-CH-Lifetime header field, the field-value indicates that the Accept-CH preference SHOULD be persisted and bound to the origin, and be considered stale after response's age ([RFC7234], section 4.2) is greater than the specified number of seconds. The preference MUST be delivered over a secure transport, and MUST NOT be persisted for an origin that isn't HTTPS.

```
Accept-CH: Sec-CH-Example, Sec-CH-Example-2
Accept-CH: Sec-CH-Example-3
Accept-CH-Lifetime: 86400
```

For example, based on the Accept-CH and Accept-CH-Lifetime 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 SHOULD persist an Accept-CH preference bound to "https://example.com" for up to 86400 seconds (1 day), 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 SHOULD NOT extend to resource requests initiated to "https://example.com" from other origins.

If Accept-CH-Lifetime occurs in a message more than once, the last value overrides all previous occurrences.

2.2.3. Interaction with Caches

When selecting an optimized 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.

3. Security Considerations

The request header fields defined in this document, and those that extend it, expose information about the user's environment to enable proactive content negotiation. Such information may reveal new information about the user and implementers ought to consider the following considerations, recommendations, and best practices.

Transmitted Client Hints header fields SHOULD NOT provide new information that is otherwise not available to the application via other means, such as using HTML, CSS, or JavaScript. Further, sending highly granular data, such as image and viewport width may help identify users across multiple requests. Reducing the set of 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. However, such precautions can still be insufficient for some types of data, especially data that can change over time.

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 may balance privacy concerns with bandwidth limitations. However, implementers should also be aware that explaining the privacy implications of passive fingerprinting to users may 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, or similar, are cleared.

4. IANA Considerations

This document defines the "Accept-CH" and "Accept-CH-Lifetime" HTTP response fields, and registers them in the Permanent Message Header Fields registry.

4.1. Accept-CH

- Header field name: Accept-CH
- Applicable protocol: HTTP
- Status: standard
- Author/Change controller: IETF
- Specification document(s): Section 2.2.1 of this document
- Related information: for Client Hints

4.2. Accept-CH-Lifetime

- Header field name: Accept-CH-Lifetime
- Applicable protocol: HTTP
- Status: standard
- Author/Change controller: IETF
- Specification document(s): Section 2.2.2 of this document
- Related information: for Client Hints

5. References

5.1. Normative References

[HTML5] Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E., O'Connor, T., and S. Pfeiffer, "HTML5", World Wide Web Consortium Recommendation REC-html5-20141028, October 2014, <http://www.w3.org/TR/2014/REC-html5-20141028>.

[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>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/info/rfc6454>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[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>.

5.2. Informative References

[KEY] Fielding, R. and M. Nottingham, "The Key HTTP Response Header Field", draft-ietf-httpbis-key-01 (work in progress), March 2016.

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

Appendix A. Interaction with Key Response Header Field

Client Hints may be combined with Key response header field ([KEY]) to enable fine-grained control of the cache key for improved cache efficiency. For example, the server can return the following set of instructions:

```
Key: Sec-CH-Example;partition=1.5:2.5:4.0
```

Above example indicates that the cache key needs to include the value of the Sec-CH-Example header field with three segments: less than 1.5, 1.5 to less than 2.5, and 4.0 or greater.

```
Key: Width;Sec-CH-Example=320
```

Above example indicates that the cache key needs to include the value of the Sec-CH-Example header field and be partitioned into groups of 320: 0-320, 320-640, and so on.

Appendix B. Changes

B.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.

B.2. Since -01

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

B.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

B.4. Since -03

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

B.5. Since -04

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

B.6. Since -05

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

B.7. Since -06

- Issue 524: Save-Data is now defined by NetInfo spec, dropping

B.8. Since -07

- Removed specific features to be defined in other specifications

Acknowledgements

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

Author's Address

Ilya Grigorik
Google

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