Workgroup: HTTP
Internet-Draft:
draft-davidben-http-client-hint-reliability-02
Updates: ietf-httpbis-client-hints
        (if approved)
Published: 30 November 2020
Intended Status: Experimental
Expires: 3 June 2021
Authors: D. Benjamin
         Google LLC

Client Hint Reliability

Abstract

This document defines the Critical-CH HTTP response header, and the ACCEPT_CH HTTP/2 and HTTP/3 frames to allow HTTP servers to reliably specify their Client Hint preferences, with minimal performance overhead.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/davidben/http-client-hint-reliability.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction
2. Conventions and Definitions
3. The Critical-CH Response Header Field
  3.1. Example
4. The ACCEPT_CH Frame
  4.1. HTTP/2 ACCEPT_CH Frame
  4.2. HTTP/3 ACCEPT_CH Frame
  4.3. Processing ACCEPT_CH Frames
  4.4. Interaction with Critical-CH
5. Security Considerations
6. IANA Considerations
7. References
  7.1. Normative References
  7.2. Informative References
Acknowledgments
Author's Address

1. Introduction

[I-D.ietf-httpbis-client-hints] defines a response header, Accept-CH, for servers to advertise a set of request headers used for proactive content negotiation. This allows user agents to send request headers only when used, improving their performance overhead as well as reducing passive fingerprinting surface.

However, on the first HTTP request to a server, the user agent will not have received the Accept-CH header and may not take the server preferences into account. More generally, the server's configuration may have changed since the most recent HTTP request to the server. This document defines a pair of mechanisms to resolve this:

1. an HTTP response header, Critical-CH, for the server to instruct the user agent to retry the request

2. an alternate delivery mechanism for Accept-CH in HTTP/2 [RFC7540] and HTTP/3 [I-D.ietf-quic-http], which can avoid the performance hit of a retry in most cases

2. Conventions and Definitions

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].

This document uses the variable-length integer encoding and frame diagram format from [I-D.ietf-quic-transport].

3. The Critical-CH Response Header Field

When a user agent requests a resource based on a missing or outdated Accept-CH value, it may not send a desired request header field. Neither user agent nor server has enough information to reliably and efficiently recover from this situation. The server can observe that the header is missing, but the user agent may not have supported the header, or may have chosen not to send it. Triggering a new request in these cases would risk an infinite loop or an unnecessary round-trip.

Conversely, the user agent can observe that a request header appears in the Accept-CH (Section 3.1 of [I-D.ietf-httpbis-client-hints]) and Vary (Section 7.1.4 of [RFC7231]) response header fields. However, retrying based on this information would waste resources if the resource only used the Client Hint as an optional optimization.

This document introduces critical Client Hints. These are the Client Hints which meaningfully change the resulting resource. For example, a server may use the Device-Memory Client Hint [DEVICE-MEMORY] to select simple and complex variants of a resource to different user agents. Such a resource should be fetched consistently across page loads to avoid jarring user-visible switches.

The server specifies critical Client Hints with the Critical-CH response header field. It is a Structured Header [I-D.ietf-httpbis-header-structure] whose value MUST be an sf-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:

```
Critical-CH = sf-list
```

For example:

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

Each token listed in the Critical-CH header SHOULD additionally be present in the Accept-CH and Vary response headers.

When a user agent receives an HTTP response containing a Critical-CH header, it first processes the Accept-CH header as described in Section 3.1 of [I-D.ietf-httpbis-client-hints]. It then performs the following steps:

1. If the request did not use a safe method (Section 4.2.1 of [RFC7231]), ignore the Critical-CH header and continue processing the response as usual.

2. If the response was already the result of a retry, ignore the Critical-CH header and continue processing the response as usual.

3. Determine the Client Hints that would have been sent given the updated Accept-CH value, incorporating the user agent's local policy and user preferences. See also Section 2.1 of [I-D.ietf-httpbis-client-hints].

4. Compare this result to the Client Hints which were sent. If any Client Hint listed in the Critical-CH header was not previously sent and would now have been sent, retry the request with the new preferences. Otherwise, continue processing the response as usual.

Note this procedure does not cause the user agent to send Client Hints it would not otherwise send.

3.1. Example

For example, if the user agent loads https://example.com with no knowledge of the server's Accept-CH preferences, it may send the following response:

```
GET / HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Content-Type: text/html
Accept-CH: Sec-CH-Example, Sec-CH-Example-2
Vary: Sec-CH-Example
Critical-CH: Sec-CH-Example
```

In this example, the server, across the whole origin, uses both Sec-CH-Example and Sec-CH-Example-2 Client Hints. However, this resource only uses Sec-CH-Example, which it considers critical.

The user agent now processes the Accept-CH header and determines it would have sent both headers. Sec-CH-Example is listed in Critical-CH, so the user agent retries the request, and receives a more specific response.

```
GET / HTTP/1.1
Host: example.com
Sec-CH-Example: 1
Sec-CH-Example-2: 2

HTTP/1.1 200 OK
Content-Type: text/html
Accept-CH: Sec-CH-Example, Sec-CH-Example-2
Vary: Sec-CH-Example
Critical-CH: Sec-CH-Example
```

4. The ACCEPT_CH Frame

While Critical-CH header provides reliability, it requires a retry on some requests. This document additionally introduces the ACCEPT_CH HTTP/2 and HTTP/3 frames as an optimization so the server's Client Hint preferences are usually available before the client's first request.

HTTP/2 and HTTP/3 servers which request Client Hints SHOULD send an ACCEPT_CH frame as early as possible. Connections using TLS [RFC8446] which negotiate the Application Layer Protocol Settings (ALPS) [I-D.vvv-tls-alps] extension SHOULD include the ACCEPT_CH frame in the ALPS value as described in [I-D.vvv-httpbis-alps]. This ensures the information is available to the user agent when it makes the first request.

[[TODO: Alternatively, is it time to revive draft-bishop-httpbis-extended-settings?]]

4.1. HTTP/2 ACCEPT_CH Frame

The HTTP/2 ACCEPT_CH frame type is TBD (decimal TBD) and contains zero or more entries, each consisting of a pair of length-delimited strings:

```
+-------------------------------+
|         Origin-Len (16)       |
+-------------------------------+-------------------------------+
|         Origin                                              ...
+-------------------------------+-------------------------------+
|         Value-Len (16)        |
+-------------------------------+-------------------------------+
|         Value                                               ...
+---------------------------------------------------------------+
```

The fields are defined as follows:

Origin-Len: An unsigned, 16-bit integer indicating the length, in octets, of the Origin field.

Origin: A sequence of characters containing the ASCII serialization of an origin (Section 6.2 of [RFC6454]) that the sender is providing an Accept-CH value for.

Value-Len: An unsigned, 16-bit integer indicating the length, in octets, of the Value field.

Value: A sequence of characters containing the Accept-CH value for the corresponding origin. This value MUST satisfy the Accept-CH ABNF defined in Section 3.1 of [I-D.ietf-httpbis-client-hints].

Clients MUST NOT send ACCEPT_CH frames. Servers which receive an ACCEPT_CH frame MUST respond with a connection error (Section 5.4.1 of [RFC7540]) of type PROTOCOL_ERROR.

ACCEPT_CH frames always apply to a single connection, never a single stream. The identifier in the ACCEPT_CH frame MUST be zero. The flags field of an ACCEPT_CH field is unused and MUST be zero. If a user agent receives an ACCEPT_CH frame whose stream identifier or flags field is non-zero, it MUST respond with a connection error of type PROTOCOL_ERROR.

4.2. HTTP/3 ACCEPT_CH Frame

The HTTP/3 ACCEPT_CH frame type is TBD (decimal TBD) and contains zero or more entries, each containing an origin and a corresponding Accept-CH value.

```
HTTP/3 ACCEPT_CH Entry {
  Origin Length (i),
  Origin (..)
  Value Length (i),
  Value (..),
}
```

```
HTTP/3 ACCEPT_CH Frame {
  Type (i) = TBD,
  Length (i),
  HTTP/3 ACCEPT_CH Entry (..) ...,
}
```

The fields of each HTTP/3 ACCEPT_CH Entry are defined as follows:

Origin Length: A variable-length integer containing the length, in bytes, of the Origin field.

Origin:
A sequence of characters containing the ASCII serialization of an origin (Section 6.2 of [RFC6454]) that the sender is providing an Accept-CH value for.

Value Length: A variable-length integer containing the length, in bytes, of the Value field.

Value: A sequence of characters containing the Accept-CH value for the corresponding origin. This value MUST satisfy the Accept-CH ABNF defined in Section 3.1 of [I-D.ietf-httpbis-client-hints].

Clients MUST NOT send ACCEPT_CH frames. Servers which receive an ACCEPT_CH frame MUST respond with a connection error (Section 8 of [I-D.ietf-quic-http]) of type H3_FRAME_UNEXPECTED.

ACCEPT_CH frames may only be sent on the control stream. Clients which receive an ACCEPT_CH frame on any other stream MUST respond with a connection error of type H3_FRAME_UNEXPECTED.

4.3. Processing ACCEPT_CH Frames

The user agent remembers the most recently received ACCEPT_CH frame for each HTTP/2 or HTTP/3 connection. When it receives a new ACCEPT_CH frame, either in application data or ALPS, it overwrites this value. As this is an optimization, the user agent MAY bound the size and ignore or forget entries to reduce resource usage.

When the user agent makes an HTTP request to a particular origin over an HTTP/2 or HTTP/3 connection, it looks up the origin in the remembered ACCEPT_CH, if any. If it finds a match, it determines additional Client Hints to send, incorporating its local policy and user preferences. See Section 2.1 of [I-D.ietf-httpbis-client-hints].

If there are additional Client Hints, the user agent restarts the request with updated headers. The connection has already been established, so this restart does not incur any additional network latency. Note it may result in a different secondary HTTP cache key (see Section 4.1 of [RFC7234]) and select a different cached response. If the new cached response does not need revalidation, it may not use the connection at all.

User agents MUST NOT process Client Hint preferences in ACCEPT_CH frames corresponding to origins for which the connection is not authoritative. Note the procedure above implicitly satisfies this by deferring processing to after the connection has been chosen for a corresponding request. Unauthoritative origins and other unmatched entries are ignored.

[[TODO: Some variations on this behavior we could choose instead:

- Do new ACCEPT_CH frames override the whole set or implement some kind of update? Overriding the whole set seems simplest and most consistent with an EXTENDED_SETTINGS variant.

- Should the user agent reject the ACCEPT_CH frame if there are unexpected origins in there? Deferring avoids needing to worry about this, and ignoring the unused ones may interact better with secondary certs.

- Should ACCEPT_CH frames be deferred or just written to the cache when received? Deferred simplifies reasoning about bad origins, predictive connections, etc., but means interactions between ACCEPT_CH and Accept-CH are more complex (see below).

- How should ACCEPT_CH and Accept-CH interact? The document currently proposes unioning them, which is easy. Accept-CH first would work, but unnecessarily ignore newer connection-level ACCEPT_CHs. ACCEPT_CH would not; a stale connection-level preference would get stuck. Whichever is received earlier would also work, but requires tracking timestamps if deferred (see above).]]

4.4. Interaction with Critical-CH

The ACCEPT_CH frame avoids a round-trip, so relying on it over Critical-CH would be preferable. However, this is not always possible:

- The server may be running older software without support for ACCEPT_CH or ALPS.

- The server's Accept-CH preferences may change while existing connections are open. Those connections will have outdated ACCEPT_CH frames. While the server could send a new frame, it may not arrive in time for the next request. Moreover, if the HTTP serving frontend is an intermediary like a CDN, it may not be proactively notified of origin server changes.

- HTTP/2 and HTTP/3 allow connection reuse across multiple origins (Section 9.1.1 of [RFC7540] and Section 3.4 of [I-D.ietf-quic-http]). Some origins may not be listed in the ACCEPT_CH frame, particularly if the server used a wildcard X.509 certificate.

Thus this document defines both mechanisms. Critical-CH provides reliable Client Hint delivery, while the ACCEPT_CH frame avoids the retry in most cases.

5. Security Considerations

Request header fields may expose sensitive information about the user's environment. Section 4.1 of [I-D.ietf-httpbis-client-hints] discusses some of these considerations. The document augments the capabilities of Client Hints, but does not change these considerations. The procedure described in Section 3 does not result in the user agent sending request headers it otherwise would not have.

The ACCEPT_CH frame does introduce a new way for HTTP/2 or HTTP/3 connections to make assertions about origins they are not authoritative for, but the procedure in Section 4.3 defers processing until after the user agent has decided to use the connection for a particular request (Section 9.1.1 of [RFC7540] and Section 3.4 of [I-D.ietf-quic-http]). The user agent will thus only use information from an ACCEPT_CH frame if it considers the connection authoritative for the origin.

6. IANA Considerations

This specification adds an entry to the "HTTP/2 Frame Type" registry [RFC7540] with the following parameters:

- Frame Type: ACCEPT_CH
- Code: TBD
- Allowed in ALPS: Yes
- Reference: [[this document]]

This specification adds an entry to the "HTTP/3 Frame Type" registry [I-D.ietf-quic-http] with the following parameters:

- Frame Type: ACCEPT_CH
- Code: TBD
- Allowed in ALPS: Yes
- Reference: [[this document]]

[[TODO: As of writing, the Frame Type registries do not include Allowed in ALPS columns, but [I-D.vvv-httpbis-alps] adds them. This document should be updated as that design evolves.]]

7. References

7.1. Normative References

[I-D.ietf-httpbis-client-hints]
Grigorik, I. and Y. Weiss, "HTTP Client Hints", Work in Progress, Internet-Draft, draft-ietf-httpbis-client-hints-15, 3 July 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-client-hints-15.txt>.

[I-D.ietf-httpbis-header-structure]
Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpbis-header-structure-19, 3 June 2020, <http://www.ietf.org/internet-drafts/draft-ietf-httpbis-header-structure-19.txt>.

[I-D.ietf-quic-http] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-http-32.txt>.

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-32, 20 October 2020, <http://www.ietf.org/internet-drafts/draft-ietf-quic-transport-32.txt>.

[I-D.vvv-httpbis-alps] Vasiliev, V., "Using TLS Application-Layer Protocol Settings (ALPS) in HTTP", Work in Progress, Internet-Draft, draft-vvv-httpbis-alps-00, 6 July 2020, <http://www.ietf.org/internet-drafts/draft-vvv-httpbis-alps-00.txt>.

[I-D.vvv-tls-alps] Benjamin, D. and V. Vasiliev, "TLS Application-Layer Protocol Settings Extension", Work in Progress, Internet-Draft, draft-vvv-tls-alps-01, 21 September 2020, <http://www.ietf.org/internet-drafts/draft-vvv-tls-alps-01.txt>.

[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>.

[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>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[DEVICE-MEMORY] Panicker, S., "Device Memory", n.d., <https://w3c.github.io/device-memory/>.

Acknowledgments

This document has benefited from contributions and suggestions from Ilya Grigorik, Nick Harper, Matt Menke, Aaron Tagliaboschi, Victor Vasiliev, Yoav Weiss, and others.

Author's Address

David Benjamin
Google LLC

Email: davidben@google.com