HTTP
Internet-Draft
Updates: ietf-httpbis-client-hints (if approved)
Intended status: Experimental
Expires: 25 August 2025

V. Tan
Google LLC

21 February 2025

Client Hint Reliability ACCEPT_CH Frame
draft-victortan-httpbis-chr-accept-ch-frame-00

Abstract

This document defines the ACCEPT_CH HTTP/2 and HTTP/3 frames to allow HTTP servers to reliably specify their Client Hint preferences. It aims to improve the performance to deliver Client Hint.

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/Tanych/http-client-hint-reliability.

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 25 August 2025.

Table of Contents

1. Introduction

To address the problem of delivering Client Hints preferences on the first request, the Critical-CH mechanism allows servers to specify critical Client Hints for rendering the requested resource. This ensures that the user agent retries the request if those hints are not initially provided.

While Critical-CH header provides reliability, it requires a retry on some requests. This document introduces the ACCEPT_CH HTTP/2 [RFC7540] and HTTP/3 [RFC9114] frames as an optimization for Accept-CH [RFC8942], which can avoid the performance hit of a retry in most cases, so the server's Client Hint preferences are usually available before the client's first request.

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 [RFC9000].

3. The ACCEPT_CH Frame

This document specifies a new ACCEPT_CH frame for HTTP/2 [RFC7540] and HTTP/3 [RFC9114]. It carries Client Hint preference for the servers. HTTP/2 and HTTP/3 servers which request Client Hints SHOULD send an ACCEPT_CH frame as early as possible. This ensures the information is available to the user agent when it makes the first request.

3.1. HTTP/2 ACCEPT_CH Frame

The HTTP/2 ACCEPT_CH frame (type=0x89) is used to specify the server's Client Hint preferences and contains zero or more entries, each consisting of a pair of length-delimited strings.

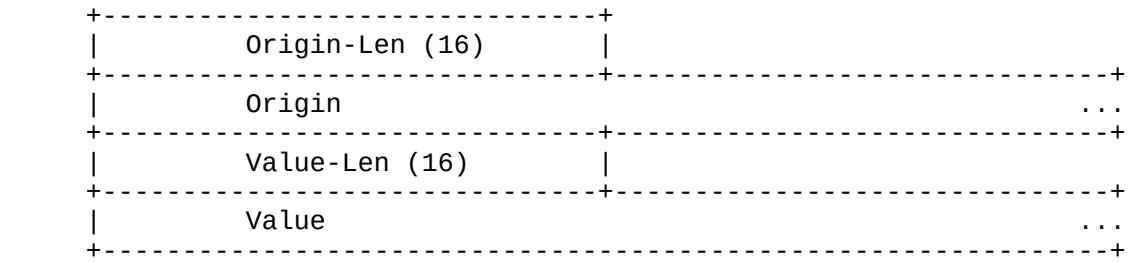
The Stream Identifier field (see Section 5.1.1 of [RFC9113]) in the ACCEPT_CH frame header MUST be zero (0x0). Receiving a ACCEPT_CH frame with a field of any other value MUST be treated as a connection error of type PROTOCOL_ERROR.

```
HTTP/2 ACCEPT_CH Frame {
  Length (24),
  Type (8) = 0x89,

  Unused Flags (8),

  Reserved (1),
  Stream Identifier (31),

  ACCEPT_CH Payload (..),
```

```
}
```

Figure 1: HTTP/2 ACCEPT_CH Frame Format

The Length, Type, Unused Flag(s), Reserved, and Stream Identifier fields are described in Section 4 of [RFC9113]. The ACCEPT_CH frame payload contains the following additional fields:



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 [RFC8942].

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 stream 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.

3.2. HTTP/3 ACCEPT_CH Frame

The HTTP/3 ACCEPT_CH frame (type=0x89) is used to specify the server’s Client Hint preferences 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) = 0x89,
  Length (i),
  HTTP/3 ACCEPT_CH Entry (..) ...,
}
```

Figure 2: HTTP/3 ACCEPT_CH Frame

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 [RFC8942].

Clients MUST NOT send ACCEPT_CH frames. Servers which receive an ACCEPT_CH frame MUST respond with a connection error (Section 8 of [RFC9114]) of type H3_FRAME_UNEXPECTED.

ACCEPT_CH frames may only be sent on the control stream (Section 6.2.1 of [RFC9114]) Clients which receive an ACCEPT_CH frame on any other stream MUST respond with a connection error of type H3_FRAME_UNEXPECTED.

3.3. 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, 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 [RFC8942].

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.

There are some variations on this behavior user agents can choose:

* Overriding ACCEPT_CH frames: This document RECOMMENDED new ACCEPT_CH frame overrides the existing set for corresponding origin because it’s simplest and most consistent with an EXTENDED_SETTINGS variant.

* Handling unexpected ACCEPT_CH frames: If ACCEPT_CH frames contains unexpected origins which the request is not interested, this feature proposes ignoring those entries because it may interact better with secondary certificates. If a request doesn’t find corresponding entries in ACCEPT_CH frames, it SHOULD be treated as

a no-op.

* Deferring ACCEPT_CH frames: ACCEPT_CH frames may be deferred or immediately written to the cache, this document RECOMMENDED deferred instead of caching them, this can avoid acting on potentially incorrect preferences from bad origins or predictive connections, but this also mean interactions between ACCEPT_CH and Accept-CH are more complex (see below).

* ACCEPT_CH frames and Accept-CH interactions: The document currently proposes merging Client Hint preferences in ACCEPT_CH frames and Accept-CH when user agents make the initial request, which is easy. However, Client Hint preferences in ACCEPT_CH frames are _NOT_ stored in the persistent Client Hint cache as ACCEPT_CH frames are strictly a connection-level setting. It would introduce unnecessary complexity if allowing more than one mechanism to affect the cache stroage. ACCEPT_CH frames are introduced as an optimization mechanism for Accept-CH and not as a replacement.

3.4. 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 frame.

* 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 3.3 of [RFC9114]). 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 an altertive delivery mechanism ACCEPT_CH frame to avoid the retry in most cases, and Critical-CH provides reliable Client Hint delivery.

4. Security Considerations

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 3.3 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 3.3 of [RFC9114]). The user agent will thus only use information from an ACCEPT_CH frame if it considers the connection authoritative for the origin.

5. IANA Considerations

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

* Frame Type: ACCEPT_CH

* Code: 0x89

* Reference: [[this document]]

This specification adds an entry to the "HTTP/3 Frame Type" registry [RFC9114] with the following parameters:

* Frame Type: ACCEPT_CH

* Code: 0x89

* Reference: [[this document]]

6. Normative References

[I-D.davidben-http-client-hint-reliability]
Benjamin, D., "Client Hint Reliability", Work in Progress, Internet-Draft, draft-davidben-http-client-hint-reliability-03, 1 June 2021, <https://datatracker.ietf.org/doc/html/draft-davidben-http-client-hint-reliability-03>.

[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>.

[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>.

[RFC8942] Grigorik, I. and Y. Weiss, "HTTP Client Hints", RFC 8942, DOI 10.17487/RFC8942, February 2021, <https://www.rfc-editor.org/info/rfc8942>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/info/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

Acknowledgments

This document has benefited from the work of David Benjamin in

[I-D.davidben-http-client-hint-reliability], contributions and suggestions from Ilya Grigorik, Nick Harper, Matt Menke, Aaron Tagliaboschi, Victor Vasiliev, Yoav Weiss, and others.

Author’s Address

Victor Tan
Google LLC
Email: victortan@google.com