Workgroup: Network Working Group
Internet-Draft: draft-ietf-webtrans-http3-05
Published: 13 March 2023
Intended Status: Standards Track
Expires: 14 September 2023
Authors: A. Frindell E. Kinnear V. Vasiliev
Facebook Apple Inc. Google

WebTransport over HTTP/3

Abstract

WebTransport [OVERVIEW] is a protocol framework that enables clients constrained by the Web security model to communicate with a remote server using a secure multiplexed transport. This document describes a WebTransport protocol that is based on HTTP/3 [HTTP3] and provides support for unidirectional streams, bidirectional streams and datagrams, all multiplexed within the same HTTP/3 connection.

Note to Readers

Discussion of this draft takes place on the WebTransport mailing list (webtransport@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/search/?email_list=webtransport>.

The repository tracking the issues for this draft can be found at <https://github.com/ietf-wg-webtrans/draft-ietf-webtrans-http3/issues>. The web API draft corresponding to this document can be found at <https://w3c.github.io/webtransport/>.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

HTTP/3 [HTTP3] is a protocol defined on top of QUIC [RFC9000] that can multiplex HTTP requests over a QUIC connection. This document defines a mechanism for multiplexing non-HTTP data with HTTP/3 in a manner that conforms with the WebTransport protocol requirements and semantics[OVERVIEW]. Using the mechanism described here, multiple WebTransport instances can be multiplexed simultaneously with regular HTTP traffic on the same HTTP/3 connection.

1.1. Terminology

The keywords "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 follows terminology defined in Section 1.2 of [OVERVIEW]. Note that this document distinguishes between a WebTransport server and an HTTP/3 server. An HTTP/3 server is the server that terminates HTTP/3 connections; a WebTransport server is an application that accepts WebTransport sessions, which can be accessed via an HTTP/3 server.

2. Protocol Overview

WebTransport servers in general are identified by a pair of authority value and path value (defined in [RFC3986] Sections 3.2 and 3.3 correspondingly).

When an HTTP/3 connection is established, both the client and server have to send a SETTINGS_ENABLE_WEBTRANSPORT setting in order to indicate that they both support WebTransport over HTTP/3.

WebTransport sessions are initiated inside a given HTTP/3 connection by the client, who sends an extended CONNECT request [RFC8441]. If the server accepts the request, a WebTransport session is established. The resulting stream will be further referred to as a *CONNECT stream*, and its stream ID is used to uniquely identify a given WebTransport session within the connection. The ID of the CONNECT stream that established a given WebTransport session will be further referred to as a *Session ID*.

After the session is established, the peers can exchange data using the following mechanisms:

* A client can create a bidirectional stream using a special indefinite-length HTTP/3 frame that transfers ownership of the stream to WebTransport.

* A server can create a bidirectional stream, which is possible since HTTP/3 does not define any semantics for server-initiated bidirectional streams.

* Both client and server can create a unidirectional stream using a special stream type.

* A datagram can be sent using HTTP Datagrams [HTTP-DATAGRAM].

A WebTransport session is terminated when the CONNECT stream that created it is closed.

3. Session Establishment

3.1. Establishing a Transport-Capable HTTP/3 Connection

In order to indicate support for WebTransport, both the client and the server MUST send a SETTINGS_ENABLE_WEBTRANSPORT value set to "1" in their SETTINGS frame. The SETTINGS_ENABLE_WEBTRANSPORT parameter value SHALL be either "0" or "1", with "0" being the default; an endpoint that receives a value other than "0" or "1" MUST close the connection with the H3_SETTINGS_ERROR error code.

The client MUST NOT send a WebTransport request until it has received the setting indicating WebTransport support from the server. Similarly, the server MUST NOT process any incoming WebTransport requests until the client settings have been received, as the client may be using a version of WebTransport extension that is different from the one used by the server.

In addition to the setting above, the server MUST send a SETTINGS_MAX_WEBTRANSPORT_SESSIONS parameter indicating the maximum number of concurrent sessions it is willing to receive. The default value for the SETTINGS_MAX_WEBTRANSPORT_SESSIONS parameter is "0", meaning that the server is not willing to receive any WebTransport sessions.

Because WebTransport over HTTP/3 requires support for HTTP/3 datagrams and the Capsule Protocol, both the client and the server MUST indicate support for HTTP/3 datagrams by sending a SETTINGS_H3_DATAGRAM value set to 1 in their SETTINGS frame (see Section 2.1.1 of [HTTP-DATAGRAM]).

WebTransport over HTTP/3 also requires support for QUIC datagrams. To indicate support, both the client and the server MUST send a max_datagram_frame_size transport parameter with a value greater than 0 (see Section 3 of [QUIC-DATAGRAM]).

3.2. Extended CONNECT in HTTP/3

[RFC8441] defines an extended CONNECT method in Section 4, enabled by the SETTINGS_ENABLE_CONNECT_PROTOCOL setting. That setting is defined for HTTP/3 by [RFC9220]. An endpoint supporting WebTransport over HTTP/3 MUST send both the SETTINGS_ENABLE_WEBTRANSPORT setting and the SETTINGS_ENABLE_CONNECT_PROTOCOL setting with values set to "1".

3.3. Creating a New Session

As WebTransport sessions are established over HTTP/3, they are identified using the https URI scheme ([HTTP], Section 4.2.2).

In order to create a new WebTransport session, a client can send an HTTP CONNECT request. The :protocol pseudo-header field ([RFC8441]) MUST be set to webtransport. The :scheme field MUST be https. Both the :authority and the :path value MUST be set; those fields indicate the desired WebTransport server. If the WebTransport session is coming from a browser client, an Origin header [RFC6454] MUST be provided within the request; otherwise, the header is OPTIONAL.

Upon receiving an extended CONNECT request with a :protocol field set to webtransport, the HTTP/3 server can check if it has a WebTransport server associated with the specified :authority and :path values. If it does not, it SHOULD reply with status code 404 (Section 15.5.5 of [HTTP]). When the request contains the Origin header, the WebTransport server MUST verify the Origin header to ensure that the specified origin is allowed to access the server in question. If the verification fails, the WebTransport server SHOULD reply with status code 403 (Section 15.5.4 of [HTTP]). If all checks pass, the WebTransport server MAY accept the session by replying with a 2xx series status code, as defined in Section 15.3 of [HTTP].

From the client's perspective, a WebTransport session is established when the client receives a 2xx response. From the server's perspective, a session is established once it sends a 2xx response.

Clients cannot initiate WebTransport in 0-RTT packets, as the CONNECT method is not considered safe; see Section 10.9 of [HTTP3]. However, WebTransport-related SETTINGS parameters may be retained from the previous session as described in Section 7.2.4.2 of [HTTP3]. If the server accepts 0-RTT, the server MUST NOT reduce the limit of maximum open WebTransport sessions from the one negotiated during the previous session; such change would be deemed incompatible, and MUST result in a H3_SETTINGS_ERROR connection error.

The webtransport HTTP Upgrade Token uses the Capsule Protocol as defined in [HTTP-DATAGRAM].

3.4. Limiting the Number of Simultaneous Sessions

This document defines a SETTINGS_MAX_WEBTRANSPORT_SESSIONS parameter that allows the server to limit the maximum number of concurrent WebTransport sessions on a single HTTP/3 connection. The client MUST NOT open more sessions than indicated in the server SETTINGS parameters. The server MUST NOT close the connection if the client opens sessions exceeding this limit, as the client and the server do not have a consistent view of how many sessions are open due to the asynchronous nature of the protocol; instead, it MUST reset all of the CONNECT streams it is not willing to process with the HTTP_REQUEST_REJECTED status defined in [HTTP3].

Just like other HTTP requests, WebTransport sessions, and data sent on those sessions, are counted against flow control limits. This document does not introduce additional mechanisms for endpoints to limit the relative amount of flow control credit consumed by different WebTransport sessions, however servers that wish to limit the rate of incoming requests on any particular session have alternative mechanisms:

* The HTTP_REQUEST_REJECTED error code defined in [HTTP3] indicates to the receiving HTTP/3 stack that the request was not processed in any way.

* HTTP status code 429 indicates that the request was rejected due to rate limiting [RFC6585]. Unlike the previous method, this signal is directly propagated to the application.

4. WebTransport Features

WebTransport over HTTP/3 provides the following features described in [OVERVIEW]: unidirectional streams, bidirectional streams and datagrams, initiated by either endpoint.

Session IDs are used to demultiplex streams and datagrams belonging to different WebTransport sessions. On the wire, session IDs are encoded using the QUIC variable length integer scheme described in [RFC9000].

The client MAY optimistically open unidirectional and bidirectional streams, as well as send datagrams, for a session that it has sent the CONNECT request for, even if it has not yet received the server's response to the request. On the server side, opening streams and sending datagrams is possible as soon as the CONNECT request has been received.

If at any point a session ID is received that cannot a valid ID for a client-initiated bidirectional stream, the recipient MUST close the connection with an H3_ID_ERROR error code.

4.1. Unidirectional streams

WebTransport endpoints can initiate unidirectional streams. The HTTP/3 unidirectional stream type SHALL be 0x54. The body of the stream SHALL be the stream type, followed by the session ID, encoded as a variable-length integer, followed by the user-specified stream data (Figure 1).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x54 (i)                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Session ID (i)                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Stream Body                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: Unidirectional WebTransport stream format

4.2. Bidirectional Streams

WebTransport endpoints can initiate bidirectional streams by opening an HTTP/3 bidirectional stream and then immediately sending a special signal value 0x41, followed by the associated session ID, both encoded as a variable-length integer; the rest of the stream is the application payload of the WebTransport stream (Figure 2).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           0x41 (i)                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Session ID (i)                       ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Stream Body                         ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Bidirectional WebTransport stream header

This document registers the special signal value 0x41 as a WEBTRANSPORT_STREAM frame type. While it is registered as an HTTP/3 frame type to avoid collisions, WEBTRANSPORT_STREAM is not a proper HTTP/3 frame, as it lacks length; it is an extension of HTTP/3 frame syntax that MUST be supported by any peer negotiating SETTINGS_ENABLE_WEBTRANSPORT. Any attempt to use WEBTRANSPORT_STREAM as a frame type outside of the very first byte of the stream MUST be treated as a connection error of type H3_FRAME_ERROR.

HTTP/3 does not by itself define any semantics for server-initiated bidirectional streams. If WebTransport setting is negotiated by both endpoints, the syntax of the server-initiated bidirectional streams SHALL be the same as the syntax of client-initiated bidirectional streams, that is, a sequence of HTTP/3 frames. The only frame defined by this document for use within server-initiated bidirectional streams is WEBTRANSPORT_STREAM.

TODO: move the paragraph above into a separate draft; define what happens with already existing HTTP/3 frames on server-initiated bidirectional streams.

4.3. Resetting Data Streams

A WebTransport endpoint may send a RESET_STREAM or a STOP_SENDING frame for a WebTransport data stream. Those signals are propagated by the WebTransport implementation to the application.

A WebTransport application SHALL provide an error code for those operations. Since WebTransport shares the error code space with HTTP/3, WebTransport application errors for streams are limited to an unsigned 8-bit integer, assuming values between 0x00 and 0xff. WebTransport implementations SHALL remap those error codes into an error range where 0x00 corresponds to 0x52e4a40fa8db, and 0xff corresponds to 0x52e4a40fa9e2. Note that there are code points inside that range of form "0x1f * N + 0x21" that are reserved by Section 8.1 of [HTTP3]; those have to be accounted for when mapping the error codes by skipping them (i.e. the two HTTP/3 error codepoints adjacent to a GREASE codepoint would map to two adjacent WebTransport application error codepoints). An example pseudocode can be seen in Figure 3.

```
first = 0x52e4a40fa8db
last = 0x52e4a40fa9e2

def webtransport_code_to_http_code(n):
    return first + n + floor(n / 0x1e)

def http_code_to_webtransport_code(h):
    assert(first <= h <= last)
    assert((h - 0x21) % 0x1f != 0)
    shifted = h - first
    return shifted - shifted // 0x1f
```

Figure 3: Pseudocode for converting between WebTransport application errors and HTTP/3 error codes; here, `//` is integer division

WebTransport data streams are associated with sessions through a header at the beginning of the stream; resetting a stream may result in that data being discarded. Because of that, WebTransport application error codes are best effort, as the WebTransport stack is not always capable of associating the reset code with a session. The only exception is the situation where there is only one session on a given HTTP/3 connection, and no intermediaries between the client and the server.

WebTransport implementations SHALL forward the error code for a stream associated with a known session to the application that owns that session; similarly, the intermediaries SHALL reset the streams with corresponding error code when receiving a reset from the peer. If a WebTransport implementation intentionally allows only one session over a given HTTP/3 connection, it SHALL forward the error codes within WebTransport application error code range to the application that owns the only session on that connection.

4.4. Datagrams

Datagrams can be sent using HTTP Datagrams. The WebTransport datagram payload is sent unmodified in the "HTTP Datagram Payload" field of an HTTP Datagram.

4.5. Buffering Incoming Streams and Datagrams

In WebTransport over HTTP/3, the client MAY send its SETTINGS frame, as well as multiple WebTransport CONNECT requests, WebTransport data streams and WebTransport datagrams, all within a single flight. As those can arrive out of order, a WebTransport server could be put into a situation where it receives a stream or a datagram without a corresponding session. Similarly, a client may receive a server-initiated stream or a datagram before receiving the CONNECT response headers from the server.

To handle this case, WebTransport endpoints SHOULD buffer streams and datagrams until those can be associated with an established session. To avoid resource exhaustion, the endpoints MUST limit the number of buffered streams and datagrams. When the number of buffered streams is exceeded, a stream SHALL be closed by sending a RESET_STREAM and/or STOP_SENDING with the H3_WEBTRANSPORT_BUFFERED_STREAM_REJECTED error code. When the number of buffered datagrams is exceeded, a datagram SHALL be dropped. It is up to an implementation to choose what stream or datagram to discard.

4.6. Interaction with HTTP/3 GOAWAY frame

HTTP/3 defines a graceful shutdown mechanism (Section 5.2 of [HTTP3]) that allows a peer to send a GOAWAY frame indicating that it will no longer accept any new incoming requests or pushes. This mechanism applies to the CONNECT requests for new WebTransport sessions. A GOAWAY frame does not affect data streams for existing WebTransport sessions; those can continue to be opened even after the GOAWAY frame has been sent or received.

To drain a WebTransport session, either endpoint can send a DRAIN_WEBTRANSPORT_SESSION capsule. After sending or receiving a DRAIN_WEBTRANSPORT_SESSION capsule, an endpoint MAY continue using the session but SHOULD attempt to gracefully terminate the session as soon as possible.

```
DRAIN_WEBTRANSPORT_SESSION Capsule {
  Type (i) = DRAIN_WEBTRANSPORT_SESSION,
  Length (i) = 0
}
```

5. Session Termination

A WebTransport session over HTTP/3 is considered terminated when either of the following conditions is met:

* the CONNECT stream is closed, either cleanly or abruptly, on either side; or

* a CLOSE_WEBTRANSPORT_SESSION capsule is either sent or received.

Upon learning that the session has been terminated, the endpoint MUST reset the send side and abort reading on the receive side of all of the streams associated with the session (see Section 2.4 of [RFC9000]) using the H3_WEBTRANSPORT_SESSION_GONE error code; it MUST NOT send any new datagrams or open any new streams.

To terminate a session with a detailed error message, an application MAY send an HTTP capsule [HTTP-DATAGRAM] of type CLOSE_WEBTRANSPORT_SESSION (0x2843). The format of the capsule SHALL be as follows:

```
CLOSE_WEBTRANSPORT_SESSION Capsule {
  Type (i) = CLOSE_WEBTRANSPORT_SESSION,
  Length (i),
  Application Error Code (32),
  Application Error Message (..8192),
}
```

CLOSE_WEBTRANSPORT_SESSION has the following fields:

Application Error Code: A 32-bit error code provided by the application closing the connection.

Application Error Message: A UTF-8 encoded error message string provided by the application closing the connection. The message takes up the remainder of the capsule, and its length MUST NOT exceed 1024 bytes.

An endpoint that sends a CLOSE_WEBTRANSPORT_SESSION capsule MUST immediately send a FIN. The endpoint MAY send a STOP_SENDING to indicate it is no longer reading from the CONNECT stream. The recipient MUST close the stream upon receiving a FIN. If any additional stream data is received on the CONNECT stream after receiving a CLOSE_WEBTRANSPORT_SESSION capsule, the stream MUST be reset with code H3_MESSAGE_ERROR.

Cleanly terminating a CONNECT stream without a CLOSE_WEBTRANSPORT_SESSION capsule SHALL be semantically equivalent to terminating it with a CLOSE_WEBTRANSPORT_SESSION capsule that has an error code of 0 and an empty error string.

In some scenarios, an endpoint might want to send a CLOSE_WEBTRANSPORT_SESSION with detailed close information and then immediately close the underlying QUIC connection. If the endpoint were to do both of those simultaneously, the peer could potentially receive the CONNECTION_CLOSE before receiving the CLOSE_WEBTRANSPORT_SESSION, thus never receiving the application error data contained in the latter. To avoid this, the endpoint SHOULD wait until all of the data on the CONNECT stream is acknowledged before sending the CONNECTION_CLOSE; this gives CLOSE_WEBTRANSPORT_SESSION properties similar to that of the QUIC CONNECTION_CLOSE mechanism as a best-effort mechanism of delivering application close metadata.

6. Negotiating the Draft Version

[[RFC editor: please remove this section before publication.]]

WebTransport over HTTP/3 uses two different mechanisms to negotiate versions for the different parts of the draft.

The hop-by-hop wire format aspects of the protocol are negotiated by changing the codepoint used for the SETTINGS_ENABLE_WEBTRANSPORT parameter. Because of that, any WebTransport endpoint MUST wait for the peer's SETTINGS frame before sending or processing any WebTransport traffic. When multiple versions are supported by both of the peers, the most recent version supported by both is selected.

The data exchanged over the CONNECT stream is transmitted across intermediaries, and thus cannot be versioned using a SETTINGS parameter. To indicate support for different versions of the protocol defined in this draft, the clients SHALL send a header for each version of the draft supported. The header corresponding to the version described in this draft is Sec-Webtransport-Http3-Draft02; its value SHALL be 1. The server SHALL reply with a Sec-Webtransport-Http3-Draft header indicating the selected version; its value SHALL be draft02 for the version described in this draft.

7. Security Considerations

WebTransport over HTTP/3 satisfies all of the security requirements imposed by [OVERVIEW] on WebTransport protocols, thus providing a secure framework for client-server communication in cases when the client is potentially untrusted.

WebTransport over HTTP/3 requires explicit opt-in through the use of an HTTP/3 setting; this avoids potential protocol confusion attacks by ensuring the HTTP/3 server explicitly supports it. It also requires the use of the Origin header, providing the server with the ability to deny access to Web-based clients that do not originate from a trusted origin.

Just like HTTP traffic going over HTTP/3, WebTransport pools traffic to different origins within a single connection. Different origins imply different trust domains, meaning that the implementations have to treat each transport as potentially hostile towards others on the same connection. One potential attack is a resource exhaustion attack: since all of the transports share both congestion control and flow control context, a single client aggressively using up those resources can cause other transports to stall. The user agent thus SHOULD implement a fairness scheme that ensures that each transport within connection gets a reasonable share of controlled resources; this applies both to sending data and to opening new streams.

A client could attempt to exhaust resources by opening too many WebTransport sessions at once. In cases when the client is untrusted, the user agent SHOULD limit the number of outgoing sessions the client can open.

8. IANA Considerations

8.1. Upgrade Token Registration

The following entry is added to the "Hypertext Transfer Protocol (HTTP) Upgrade Token Registry" registry established by Section 16.7 of [HTTP].

The "webtransport" label identifies HTTP/3 used as a protocol for WebTransport:

Value: webtransport

Description: WebTransport over HTTP/3

Reference: This document and [I-D.ietf-webtrans-http2]

8.2. HTTP/3 SETTINGS Parameter Registration

The following entries are added to the "HTTP/3 Settings" registry established by [HTTP3]:

The SETTINGS_ENABLE_WEBTRANSPORT parameter indicates that the specified HTTP/3 connection is WebTransport-capable.

Setting Name: ENABLE_WEBTRANSPORT

Value: 0x2b603742

Default: 0

Specification: This document

The SETTINGS_WEBTRANSPORT_MAX_SESSIONS parameter indicates that the specified HTTP/3 server is WebTransport-capable and the number of concurrent sessions it is willing to receive.

Setting Name: WEBTRANSPORT_MAX_SESSIONS

Value: 0x2b603743

Default: 0

Specification: This document

8.3. Frame Type Registration

The following entry is added to the "HTTP/3 Frame Type" registry established by [HTTP3]:

The WEBTRANSPORT_STREAM frame allows HTTP/3 client-initiated and server-initiated bidirectional streams to be used by WebTransport:

Code: 0x41

Frame Type: WEBTRANSPORT_STREAM

Specification: This document

8.4. Stream Type Registration

The following entry is added to the "HTTP/3 Stream Type" registry established by [HTTP3]:

The "WebTransport stream" type allows unidirectional streams to be used by WebTransport:

Code: 0x54

Stream Type: WebTransport stream

Specification: This document

Sender: Both

8.5. HTTP/3 Error Code Registration

The following entry is added to the "HTTP/3 Error Code" registry established by [HTTP3]:

Name: H3_WEBTRANSPORT_BUFFERED_STREAM_REJECTED

Value: 0x3994bd84

Description: WebTransport data stream rejected due to lack of associated session.

Specification: This document.

Name: H3_WEBTRANSPORT_SESSION_GONE

Value: 0x170d7b68

Description: WebTransport data stream aborted because the associated WebTransport session has been closed.

Specification: This document.

In addition, the following range of entries is registered:

Name: H3_WEBTRANSPORT_APPLICATION_00 ... H3_WEBTRANSPORT_APPLICATION_FF

Value: 0x52e4a40fa8db to 0x52e4a40fa9e2 inclusive, with the exception of 0x52e4a40fa8f9, 0x52e4a40fa918, 0x52e4a40fa937, 0x52e4a40fa956, 0x52e4a40fa975, 0x52e4a40fa994, 0x52e4a40fa9b3, and 0x52e4a40fa9d2.

Description: WebTransport application error codes.

Specification:
This document.

8.6. Capsule Types

The following entries are added to the "HTTP Capsule Types" registry established by [HTTP-DATAGRAM]:

The CLOSE_WEBTRANSPORT_SESSION capsule.

Value: 0x2843
Capsule Type: CLOSE_WEBTRANSPORT_SESSION
Status: permanent
Specification: This document
Change Controller: IETF
Contact: WebTransport Working Group webtransport@ietf.org
Notes: None

9. References

9.1. Normative References

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[HTTP-DATAGRAM] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", RFC 9297, DOI 10.17487/RFC9297, August 2022, <https://www.rfc-editor.org/rfc/rfc9297>.

[HTTP3] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

[OVERVIEW] Vasiliev, V., "The WebTransport Protocol Framework", Work in Progress, Internet-Draft, draft-ietf-webtrans-overview-05, 24 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-overview-05>.

[QUIC-DATAGRAM] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/rfc/rfc9221>.

[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/rfc/rfc2119>.

[RFC3986]
Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

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

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/rfc/rfc6585>.

[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/rfc/rfc8174>.

[RFC8441] McManus, P., "Bootstrapping WebSockets with HTTP/2", RFC 8441, DOI 10.17487/RFC8441, September 2018, <https://www.rfc-editor.org/rfc/rfc8441>.

[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/rfc/rfc9000>.

[RFC9220] Hamilton, R., "Bootstrapping WebSockets with HTTP/3", RFC 9220, DOI 10.17487/RFC9220, June 2022, <https://www.rfc-editor.org/rfc/rfc9220>.

9.2. Informative References

[I-D.ietf-webtrans-http2]
Frindell, A., Kinnear, E., Pauly, T., Thomson, M., Vasiliev, V., and G. Xie, "WebTransport using HTTP/2", Work in Progress, Internet-Draft, draft-ietf-webtrans-http2-04, 19 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-webtrans-http2-04>.

Authors' Addresses

Alan Frindell
Facebook

Email: afrind@fb.com

Eric Kinnear
Apple Inc.

Email: ekinnear@apple.com

Victor Vasiliev
Google

Email: vasilvv@google.com