HTTP
Internet-Draft
Updates: 6455 (if approved)
Intended status: Standards Track
Expires: December 20, 2018

P. McManus
Mozilla

June 18, 2018

Bootstrapping WebSockets with HTTP/2
draft-ietf-httpbis-h2-websockets-07

Abstract

This document defines a mechanism for running the WebSocket Protocol (RFC 6455) over a single stream of an HTTP/2 connection.

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 December 20, 2018.

Copyright Notice

Table of Contents

1. Introduction

The Hypertext Transfer Protocol (HTTP) [RFC7230] provides compatible resource-level semantics across different versions but it does not offer compatibility at the connection management level. Other protocols, such as WebSockets, that rely on connection management details of HTTP must be updated for new versions of HTTP.

The WebSocket Protocol [RFC6455] uses the HTTP/1.1 Upgrade mechanism (Section 6.7 of [RFC7230]) to transition a TCP connection from HTTP into a WebSocket connection. A different approach must be taken with HTTP/2 [RFC7540]. HTTP/2 does not allow connection-wide header fields and status codes such as the Upgrade and Connection request header fields or the 101 (Switching Protocols) response code due to its multiplexing nature. These are all required by the [RFC6455] opening handshake.

Being able to bootstrap WebSockets from HTTP/2 allows one TCP connection to be shared by both protocols and extends HTTP/2's more efficient use of the network to WebSockets.

This document extends the HTTP CONNECT method (as specified for HTTP/2 in Section 8.3 of [RFC7540]). The extension allows the substitution of a new protocol name to connect to rather than the external host normally used by CONNECT. The result is a tunnel on a single HTTP/2 stream that can carry data for WebSockets (or any other protocol). The other streams on the connection may carry more extended CONNECT tunnels, traditional HTTP/2 data, or a mixture of both.

This tunneled stream will be multiplexed with other regular streams on the connection and enjoys the normal priority, cancellation, and flow control features of HTTP/2.

Streams that successfully establish a WebSocket connection using a tunneled stream and the modifications to the opening handshake defined in this document then use the traditional WebSocket Protocol, treating the stream as if were the TCP connection in that specification.

2. Terminology

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.

3. The SETTINGS_ENABLE_CONNECT_PROTOCOL SETTINGS Parameter

This document adds a new SETTINGS Parameter to those defined by [RFC7540], Section 6.5.2.

The new parameter name is SETTINGS_ENABLE_CONNECT_PROTOCOL. The value of the parameter MUST be 0 or 1.

Upon receipt of SETTINGS_ENABLE_CONNECT_PROTOCOL with a value of 1, a client MAY use the Extended CONNECT definition of this document when creating new streams. Receipt of this parameter by a server does not have any impact.

A sender MUST NOT send a SETTINGS_ENABLE_CONNECT_PROTOCOL parameter with the value of 0 after previously sending a value of 1.

The use of a SETTINGS Parameter to opt-in to an otherwise incompatible protocol change is a use of "Extending HTTP/2" defined by Section 5.5 of [RFC7540]. Specifically, the addition a new pseudo-header field ":protocol" and the change in meaning of the ":authority" pseudo-header field in Section 4 require opt-in negotiation. If a client were to use the provisions of the extended CONNECT method defined in this document without first receiving a SETTINGS_ENABLE_CONNECT_PROTOCOL parameter, a non-supporting peer would detect a malformed request and generate a stream error (Section 8.1.2.6 of [RFC7540]).

4. The Extended CONNECT Method

Usage of the CONNECT method in HTTP/2 is defined by Section 8.3 of [RFC7540]. This extension modifies the method in the following ways:

- A new pseudo-header field :protocol MAY be included on request HEADERS indicating the desired protocol to be spoken on the tunnel created by CONNECT. The pseudo-header field is single valued and contains a value from the HTTP Upgrade Token Registry located at https://www.iana.org/assignments/http-upgrade-tokens/http-upgrade-tokens.xhtml

- On requests that contain the :protocol pseudo-header field, the :scheme and :path pseudo-header fields of the target URI (See Section 5) MUST also be included.

- On requests bearing the :protocol pseudo-header field, the :authority pseudo-header field is interpreted according to Section 8.1.2.3 of [RFC7540] instead of Section 8.3 of [RFC7540]. In particular, the server MUST NOT create a tunnel to the host indicated by the :authority as it would with a CONNECT method request that was not modified by this extension.

Upon receiving a CONNECT request bearing the :protocol pseudo-header field the server establishes a tunnel to another service of the protocol type indicated by the pseudo-header field. This service may or may not be co-located with the server.

5. Using Extended CONNECT To Bootstrap the WebSocket Protocol

The :protocol pseudo-header field MUST be included in the CONNECT request and it MUST have a value of "websocket" to initiate a WebSocket connection on an HTTP/2 stream. Other HTTP request and response header fields, such as those for manipulating cookies, may be included in the HEADERS with the CONNECT method as usual. This request replaces the GET-based request in [RFC6455] and is used to process the WebSockets opening handshake.

The scheme of the target URI (Section 5.1 of [RFC7230]) MUST be "https" for "wss" schemed WebSockets and "http" for "ws" schemed WebSockets. The remainder of the Target URI is the same as the websocket URI. The websocket URI is still used for proxy autoconfiguration. The security requirements for the HTTP/2 connection used by this specification are established by [RFC7540] for https requests and [RFC8164] for http requests.

[RFC6455] requires the use of Connection and Upgrade header fields that are not part of HTTP/2. They MUST NOT be included in the CONNECT request defined here.

[RFC6455] requires the use of a Host header field which is also not part of HTTP/2. The Host information is conveyed as part of the :authority pseudo-header field which is required on every HTTP/2 transaction.

Implementations using this extended CONNECT to bootstrap WebSockets do not do the processing of the [RFC6455] Sec-WebSocket-Key and Sec-WebSocket-Accept header fields as that functionality has been superseded by the :protocol pseudo-header field.

The Origin [RFC6454], Sec-WebSocket-Version, Sec-WebSocket-Protocol, and Sec-WebSocket-Extensions header fields are used in the CONNECT request and response header fields in the same way as defined in [RFC6455]. Note that HTTP/1 header field names were case-insensitive and HTTP/2 requires they be encoded as lower case.

After successfully processing the opening handshake, the peers should proceed with the WebSocket Protocol [RFC6455] using the HTTP/2 stream from the CONNECT transaction as if it were the TCP connection referred to in [RFC6455]. The state of the WebSocket connection at this point is OPEN as defined by [RFC6455], Section 4.1.

The HTTP/2 stream closure is also analogous to the TCP connection closure of [RFC6455]. Orderly TCP level closures are represented as END_STREAM ([RFC7540], Section 6.1) flags and RST exceptions are represented with the RST_STREAM ([RFC7540], Section 6.4) frame with the CANCEL ([RFC7540], Section 7) error code.

5.1. Example

```
[[ From Client ]]                       [[ From Server ]]

                                        SETTINGS
                                        SETTINGS_ENABLE_CONNECT_[..] = 1

HEADERS + END_HEADERS
:method = CONNECT
:protocol = websocket
:scheme = https
:path = /chat
:authority = server.example.com
sec-websocket-protocol = chat, superchat
sec-websocket-extensions = permessage-deflate
sec-websocket-version = 13
origin = http://www.example.com

                                        HEADERS + END_HEADERS
                                        :status = 200
                                        sec-websocket-protocol = chat

DATA
WebSocket Data

                                        DATA + END_STREAM
                                        WebSocket Data

DATA + END_STREAM
WebSocket Data
```

6. Design Considerations

A more native integration with HTTP/2 is certainly possible with larger additions to HTTP/2. This design was selected to minimize the solution complexity while still addressing the primary concern of running HTTP/2 and WebSockets concurrently.

7. About Intermediaries

This document does not change how WebSockets interacts with HTTP forward proxies. If a client wishing to speak WebSockets connects via HTTP/2 to an HTTP proxy it should continue to use a traditional (i.e. not with a :protocol pseudo-header field) CONNECT to tunnel through that proxy to the WebSocket server via HTTP.

The resulting version of HTTP on that tunnel determines whether WebSockets is initiated directly or via a modified CONNECT request described in this document.

8. Security Considerations

[RFC6455] ensures that non-WebSockets clients, especially XMLHttpRequest based clients, cannot make a WebSocket connection. Its primary mechanism for doing that is the use of Sec- prefixed request header fields that cannot be created by XMLHttpRequest-based clients. This specification addresses that concern in two ways:

- XMLHttpRequest also prohibits use of the CONNECT method in addition to Sec- prefixed request header fields.
- The use of a pseudo-header field is something that is connection specific and HTTP/2 does not ever allow to be created outside of the protocol stack.

The security considerations of [RFC6455] section 10 continue to apply to the use of the WebSocket Protocol when using this specification with the exception of 10.8. That section is not relevant because it is specific to the boostrapping handshake that is changed in this document.

9. IANA Considerations

This document establishes an entry for the HTTP/2 Settings Registry that was established by Section 11.3 of [RFC7540].

Name: SETTINGS_ENABLE_CONNECT_PROTOCOL

Code: 0x8

Initial Value: 0

Specification: This document

10. Normative References

[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>.

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

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/info/rfc6455>.

[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>.

[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>.

[RFC8164] Nottingham, M. and M. Thomson, "Opportunistic Security for HTTP/2", RFC 8164, DOI 10.17487/RFC8164, May 2017, <https://www.rfc-editor.org/info/rfc8164>.

[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>.

Acknowledgments

The 2017 HTTP Workshop had a very productive discussion that helped determine the key problem and acceptable level of solution complexity.

Author's Address

Patrick McManus
Mozilla

Email: mcmanus@ducksong.com