Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 18, 2018

P. McManus
Mozilla

October 15, 2017

Bootstrapping WebSockets with HTTP/2
draft-mcmanus-httpbis-h2-websockets-00

Abstract

This document defines a mechanism for running the WebSocket Protocol [RFC6455] 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 April 18, 2018.

Copyright Notice

Table of Contents

1. Introduction

The Hypertext Transfer Protocol (HTTP) 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 [RFC7230] Upgrade mechanism to transition a TCP connection from HTTP into a WebSocket connection. A different approach must be taken with HTTP/2 [RFC7540]. Due to the multiplexing nature of HTTP/2 it does not allow connection wide header and status codes such as the Upgrade and Connection request headers or the 101 response code. These are all required by the [RFC6455] connection establishment process.

A server offering both HTTP/1.1 and WebSocket services can do so from the same instance and same port although they require separate TCP connections. Moving a server to HTTP/2 and WebSocket services requires a separate port and protocol stack for the sole purpose of bootstrapping WebSockets. This is a significant administrative burden and may not even be possible in the case of large amounts of deployed markup pointing at the old single name and port. Being able to bootstrap WebSockets from HTTP/2 allows one server, one port, and one TCP connection to be shared by both protocols.

This document extends the HTTP/2 CONNECT method. 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) while the other streams on the connection continue to carry HTTP/2 data.

Streams that have been successfully established as protocol tunnels proceed to establish and utilize the WebSocket Protocol using the procedure defined by [RFC6455] treating the stream as if were the connection in that specification.

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.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, [RFC2119].

3. The ENABLE_CONNECT_PROTOCOL SETTINGS Parameter

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

The new parameter is ENABLE_CONNECT_PROTOCOL (type = 0x8). The value of the parameter MUST be 0 or 1.

Upon receipt of 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 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]. If a client were to use the provisions of the extended CONNECT method defined in this document without first receiving a ENABLE_CONNECT_PROTOCOL parameter with the value of 1 it would be a protocol violation.

4. The Extended CONNECT Method

The CONNECT Method of [RFC7540] Section 8.3 is modified in the following ways:

- A new pseudo-header :protocol MAY be included on request HEADERS indicating the desired protocol to be spoken on the tunnel created by CONNECT. The pseudo-header is single valued and contains a value from the HTTP Upgrade Token Registry defined by [RFC7230].

- On requests bearing the :protocol pseudo-header, the :scheme and :path pseudo-header fields SHOULD be included.

- On requests bearing the :protocol pseudo-header, the :authority pseudo-header field is interpreted according to [RFC7540] Section 8.1.2.3 instead of [RFC7540] Section 8.3. In particular the server MUST not make a new TCP connection to the host and port indicated by the :authority.

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

4.1. Using Extended CONNECT To Bootstrap The WebSocket Protocol

The pseudo-header :protocol 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.

Upon successfully establishing a protocol tunnel the client should proceed with The WebSocket Protocol [RFC6455] using the HTTP/2 stream from the CONNECT transaction as if it were the TCP connection in [RFC6455]. Negotiation of WebSocket version and sub-protocols is done unmodified within that stream.

4.2. Example

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

                                           SETTINGS
                                           ENABLE_CONNECT_PROTOCOL = 1

  HEADERS + END_HEADERS
  :method = CONNECT
  :protocol = websocket
  :scheme = wss
  :path = /chat
  :authority = server.example.com:443

                                           HEADERS + END_HEADERS
                                           :status = 200

  DATA
  GET /chat HTTP/1.1
  Host: server.example.com
  Upgrade: websocket
  Connection: Upgrade
  Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
  Origin: http://example.com
  Sec-WebSocket-Protocol: chat, superchat
  Sec-WebSocket-Version: 13

                                           DATA
                                           HTTP/1.1 101 Plead The Fifth
                                           Upgrade: websocket
                                           Connection: Upgrade
                                           Sec-WebSocket-Accept:
                                            s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
                                           Sec-WebSocket-Protocol: chat

  DATA
  WebSocket Data

                                           DATA + END_STREAM
                                           WebSocket Data

  DATA + END_STREAM
  WebSocket Data
```

5. 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 not being able to run HTTP/2 and WebSockets on the same port and address.

6. About Intermediaries

This document does not change how WebSockets interacts with HTTP proxies. If a client wishing to speak WebSockets connects via HTTP/2 to a HTTP proxy it should continue to use a traditional (i.e. not with a :protocol pseudo-header) 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.

7. 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 headers that cannot be created by XMLHttpRequest based clients. This specification addresses that concern in two ways:

- The CONNECT method is prohibited from being used by XMLHttpRequest
- The use of a pseudo-header is something that is connection specific and HTTP/2 does not ever allow to be created outside of the protocol stack.

8. IANA Considerations

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

Name: ENABLE_CONNECT_PROTOCOL

Code: 0x8

Initial Value: 0

Specification: This document

9. Acknowledgments

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

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>.

[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>.

[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>.

[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>.

Author's Address

Patrick McManus
Mozilla

Email: mcmanus@ducksong.com