Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 1 January 2026

B. M. Schwartz
Meta Platforms, Inc.
奥 一穂 (K. Oku)
Fastly

30 June 2025

HTTP Version Translation of the Capsule Protocol
draft-kb-capsule-conversion-01

Abstract

This draft specifies how HTTP intermediaries can translate the Capsule Protocol between HTTP versions.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-kb-capsule-conversion/.

Source for this draft and an issue tracker can be found at https://github.com/bemasc/capsule-conversion.

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 1 January 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. Conventions and Definitions 3
3. Requirements 3
3.1. Converting an HTTP/1.1 Upgrade request to Extended CONNECT 3
3.2. Converting an Extended CONNECT request to HTTP/1.1 Upgrade 5
3.3. Converting an Extended CONNECT request to Extended CONNECT for a different HTTP version 6
4. Implications 6
5. Security Considerations 7
6. IANA Considerations 7
7. References 7
7.1. Normative References 7
7.2. Informative References 7
Acknowledgments 8
Authors' Addresses 8

1. Introduction

The Capsule Protocol [RFC9297] defines a framing layer that can be used for protocols running over HTTP. The Capsule Protocol consists of a linear stream of capsules, each with a specified type and size. Endpoints can establish a Capsule Protocol connection using HTTP Extended CONNECT or the HTTP/1.1 Upgrade process.

Intermediaries such as HTTP Gateways can play an active role in the Capsule Protocol. To inform intermediaries that this protocol is in use, endpoints can include a "Capsule-Protocol: ?1" header in their request or response. Intermediaries are obligated to pass unrecognized capsule types unmodified, but some capsule types do permit intermediaries to modify them.

The Capsule Protocol is defined for HTTP/1.1, HTTP/2, and HTTP/3, and intermediaries can translate Capsule Protocol streams between different versions of HTTP. For example, an HTTP Gateway receiving a request using the "connect-tcp-capsule" Upgrade Token

[I-D.ietf-httpbis-connect-tcp] over HTTP/3 might forward it to a backend server using HTTP/1.1 for further processing. However, this translation currently requires the intermediary to recognize the specified Upgrade Token. Unrecognized Upgrade Tokens cannot be translated between HTTP versions.

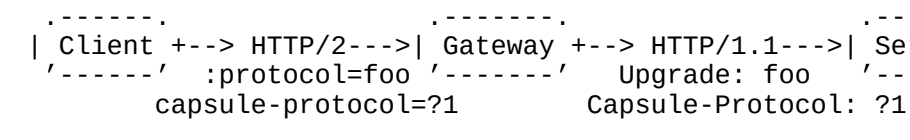

Figure 1: HTTP version translation of an Upgrade Token. The gateway can only perform this translation if it recognizes the "foo" Upgrade Token.

As a result of this limitation, HTTP intermediaries cannot forward unrecognized Capsule Protocol Upgrade Tokens (CPUTs) unless the backend supports the HTTP version used by the client. In practice, such HTTP version mismatches are common, so intermediaries have preferred not to support unrecognized tokens at all. As a result, each new CPUT requires the cooperation of any HTTP intermediaries. This increases the maintenance burden on intermediaries and impedes the deployment of novel CPUTs.

This draft specifies general rules for translating Capsule Protocol requests across HTTP versions, allowing intermediaries to perform such translations for unrecognized CPUTs.

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.

3. Requirements

This section describes requirements on HTTP intermediaries that change the HTTP version of a Capsule Protocol request.

3.1. Converting an HTTP/1.1 Upgrade request to Extended CONNECT

A Convertible Upgrade Request is a request that meets these criteria:

* The HTTP version is "1.1".

* The method is "GET".

* An "Upgrade" header is present and specifies exactly one Upgrade Token.

* The request is known to use the Capsule Protocol, because:

  - A "Capsule-Protocol" header is present with an item value of "?1" (with or without parameters), OR

  - The intermediary knows that this Upgrade Token always uses the Capsule Protocol.

* The request is otherwise well-formed for use with the Capsule Protocol.

Upon receiving a Convertible Upgrade Request, an HTTP intermediary MAY convert it into an Extended CONNECT request ([RFC9220][RFC8441]) using ordinary HTTP version translation with the following modifications:

* Change the method to "CONNECT".

* Add a ":protocol" pseudo-header containing the specified Upgrade Token.

* Add a "capsule-protocol: ?1" header if no "Capsule-Protocol" header is present.

(Note that ordinary HTTP version translation removes the "Connection" and "Upgrade" headers.)

If the intermediary receives a "200 (OK)" response, it MUST convert it to an HTTP/1.1 Upgrade response as follows:

* Change the response code to "101 (Switching Protocols)".

* Add an "Upgrade" header containing the specified Upgrade Token.

* Add a "Connection: Upgrade" header.

After sending this response, the intermediary MUST process all data to and from the client in accordance with the Capsule Protocol.

If the intermediary receives any other valid response, it MUST NOT convert it to an HTTP/1.1 Upgrade response, and MUST forward it using ordinary HTTP version translation. If the response status was not 1xx (Informational), the intermediary MAY accept additional HTTP/1.1 requests on this connection to the client.

3.2. Converting an Extended CONNECT request to HTTP/1.1 Upgrade

A Convertible Extended CONNECT request is a request that meets these criteria:

* The method is "CONNECT".
* The ":protocol" pseudo-header is present, providing an Upgrade Token.
* The request is known to use the Capsule Protocol, because:
  - A "capsule-protocol" header is present with an item value of "?1" (with or without parameters), OR
  - The intermediary recognizes the provided Upgrade Token and knows that it always uses the Capsule Protocol.
* The request is otherwise well-formed for use with the Capsule Protocol.

Upon receiving a Convertible Extended CONNECT Request, an HTTP intermediary MAY convert it into an HTTP/1.1 Upgrade request according to ordinary HTTP version translation, with the following modifications:

* Change the method to "GET".
* Replace the ":protocol" pseudo-header with an "Upgrade" header containing the same Upgrade Token.
* Add a "Connection: Upgrade" header.
* Add a "Capsule-Protocol: ?1" header if no "capsule-protocol" header is present.

If the intermediary receives a correctly formed "101 (Switching Protocols)" response, it MUST change the response code to "200 (OK)". If it receives a 2xx (Successful) response, it SHOULD return a "501 (Not Implemented)" status code, to indicate that the ":protocol" value was not accepted ([RFC9220], Section 3). Otherwise, it MUST forward any valid responses unmodified. After sending a "200" response, the intermediary MUST process all further data to and from the server in accordance with the Capsule Protocol.

3.3. Converting an Extended CONNECT request to Extended CONNECT for a different HTTP version

An HTTP intermediary MAY translate a Convertible Extended CONNECT Request between different HTTP versions using ordinary HTTP version translation.

4. Implications

Translation of unrecognized CPUTs across HTTP versions carries some implications for future specifications related to the Capsule Protocol:

* All CPUTs must treat "GET" in HTTP/1.1 as semantically equivalent to Extended CONNECT.

* The "Capsule-Protocol" response header has no effect and should be treated as a hint for later analysis. Intermediaries can process the response as the Capsule Protocol based entirely on the request headers and the response status code.

* Intermediaries' behavior regarding each capsule type is independent of the CPUT. CPUTs cannot change intermediaries' treatment of existing capsule types.

* A Capsule Type or CPUT cannot change the meaning of an HTTP extension. It can only rely on the behaviors that are defined as mandatory for any implementation of that extension. Extensions intended for use with the Capsule Protocol will likely need to define how HTTP version translation works.

* Capsule Protocol endpoints are defined independently of the HTTP version, like ordinary HTTP resources. If an origin server's Capsule Protocol support varies between HTTP versions, clients may observe inconsistent behavior when accessing the origin through a compliant intermediary.

* If a future CPUT specification intends for all clients to be compatible with HTTP version translation by pre-existing intermediaries under this specification, it will have to make the "Capsule-Protocol: ?1" request header mandatory, as permitted by [RFC9297], Section 3.4.

* A "Capsule-Protocol: ?0" request header cannot be used to disable HTTP version translation.

All existing CPUTs and Capsule Types already conform to these rules.

5. Security Considerations

When implemented incorrectly, HTTP/1.1 Upgrade carries a risk of request smuggling. (See [I-D.ietf-httpbis-optimistic-upgrade].)

Intermediary implementors should take care to avoid excessive resource consumption by malicious clients. For example, a client might be able to send many short requests over a single HTTP/2 or HTTP/3 connection, each of which requires opening a new, long-lived TCP connection to the backend.

6. IANA Considerations

This document has no IANA actions.

7. References

7.1. 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/rfc/rfc2119>.

[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>.

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

[RFC9297] 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>.

7.2. Informative References

[I-D.ietf-httpbis-connect-tcp] Schwartz, B. M., "Template-Driven HTTP CONNECT Proxying for TCP", Work in Progress, Internet-Draft, draft-ietf-httpbis-connect-tcp-08, 11 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-connect-tcp-08>.

[I-D.ietf-httpbis-optimistic-upgrade]
Schwartz, B. M., "Security Considerations for Optimistic Protocol Transitions in HTTP/1.1", Work in Progress, Internet-Draft, draft-ietf-httpbis-optimistic-upgrade-04, 11 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-optimistic-upgrade-04>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Benjamin M. Schwartz
Meta Platforms, Inc.
Email: ietf@bemasc.net

Kazuho Oku
Fastly
Email: kazuhooku@gmail.com

Additional contact information:

奥 一穂
Fastly