Network Working Group
Internet-Draft
Intended status: Experimental
Expires: October 14, 2016

M. Kerwin

April 12, 2016

HTTP/2 Gzipped Data
draft-kerwin-http2-encoded-data-08

Abstract

This document introduces a new frame type for transporting gzip-encoded data between peers in the Hypertext Transfer Protocol Version 2 (HTTP/2), and an associated error code for handling invalid encoding.

Note to Readers

The issues list for this draft can be found at https://github.com/phluid61/internet-drafts/labels/HTTP%2F2%20Gzipped%20Data

The most recent (often unpublished) draft is at http://phluid61.github.io/internet-drafts/http2-encoded-data/

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 http://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 October 14, 2016.

Copyright Notice

Table of Contents

1. Introduction

This document introduces a mechanism for applying gzip encoding [RFC1952] to data transported between two endpoints in the Hypertext Transfer Protocol Version 2 (HTTP/2) [RFC7540], analogous to Transfer-Encoding in HTTP/1.1 [RFC7230].

1.1. Notational Conventions

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

2. Additions to HTTP/2

This document introduces a new HTTP/2 frame type ([RFC7540], Section 11.2), a new HTTP/2 setting ([RFC7540], Section 11.3), and a new HTTP/2 error code ([RFC7540], Section 7), to allow the compression of data.

Note that while compressing some or all data in a stream might affect the total length of the corresponding HTTP message body, the "content-length" header, if present, should continue to reflect the total length of the _uncompressed_ data. This is particularly relevant when detecting malformed messages ([RFC7540], Section 8.1.2.6).

2.1. SETTINGS_ACCEPT_GZIPPED_DATA

SETTINGS_ACCEPT_GZIPPED_DATA (0xTBA) is used to indicate the sender's ability and willingness to receive GZIPPED_DATA frames. An endpoint MUST NOT send a GZIPPED_DATA frame unless it receives this setting with a value of 1.

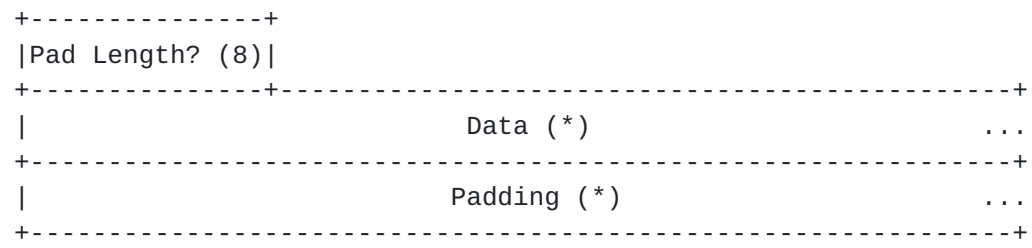
The initial value is 0, which indicates that GZIPPED_DATA frames are not supported. Any value other than 0 or 1 MUST be treated as a connection error ([RFC7540], Section 5.4.1) of type PROTOCOL_ERROR.

An endpoint may advertise support for GZIPPED_DATA frames and later decide that it no longer supports them. After sending an ACCEPT_GZIPPED_DATA setting with the value 0, the endpoint SHOULD continue to accept GZIPPED_DATA frames for a reasonable amount of time to account for frames that may already be in flight.

2.2. GZIPPED_DATA

GZIPPED_DATA frames (type code=0xTBA) are semantically identical to DATA frames ([RFC7540], Section 6.1), but their payload is encoded using gzip compression. Significantly: the order of DATA and GZIPPED_DATA frames is semantically significant; and GZIPPED_DATA frames are subject to flow control ([RFC7540], Section 5.2). Gzip compression is an LZ77 coding with a 32 bit CRC that is commonly produced by the gzip file compression program [RFC1952].

Any compression or decompression context for a GZIPPED_DATA frame is unique to that frame. An endpoint MAY interleave DATA and GZIPPED_DATA frames on a single stream.



GZIPPED_DATA Frame Payload

The GZIPPED_DATA frame contains the following fields:

- Pad Length: An 8-bit field containing the length of the frame padding in units of octets. This field is optional and is only present if the PADDED flag is set.

- Data: Encoded application data. The amount of encoded data is the remainder of the frame payload after subtracting the length of the other fields that are present.

- Padding: Padding octets that contain no application semantic value. Padding octets MUST be set to zero when sending and ignored when receiving.

The GZIPPED_DATA frame defines the following flags:

- "END_STREAM" (0x1): Bit 1 being set indicates that this frame is the last that the endpoint will send for the identified stream. Setting this flag causes the stream to enter one of the "half closed" states or the "closed" state ([RFC7540], Section 5.1).

- "PADDED" (0x8): Bit 4 being set indicates that the Pad Length field is present.

A GZIPPED_DATA frame MUST NOT be sent if the ACCEPT_GZIPPED_DATA setting of the peer is set to 0. See Section 3.

An intermediary, on receiving a GZIPPED_DATA frame, MAY decode the data and forward it to its downstream peer in one or more DATA frames. If the downstream peer has not advertised support for GZIPPED_DATA frames (by sending an ACCEPT_GZIPPED_DATA setting with the value 1) the intermediary MUST decode the data before forwarding it.

If an endpoint detects that the payload of a GZIPPED_DATA frame is not encoded correctly, for example with an incorrect checksum, the endpoint MUST treat this as a stream error (see [RFC7540], Section 5.4.2) of type DATA_ENCODING_ERROR (Section 2.3). The

endpoint MAY then choose to immediately send an ACCEPT_GZIPPED_DATA setting with the value 0.

If an intermediary propagates a GZIPPED_DATA frame from the source peer to the destination peer without modifying the payload or its encoding, and receives a DATA_ENCODING_ERROR from the receiving peer, it SHOULD pass the error on to the source peer.

GZIPPED_DATA frames MUST be associated with a stream. If a GZIPPED_DATA frame is received whose stream identifier field is 0x0, the recipient MUST respond with a connection error ([RFC7540], Section 5.4.1) of type PROTOCOL_ERROR.

GZIPPED_DATA frames are subject to flow control and can only be sent when a stream is in the "open" or "half closed (remote)" states. The entire GZIPPED_DATA frame payload is included in flow control, including the Pad Length and Padding fields if present. If a GZIPPED_DATA frame is received whose stream is not in "open" or "half closed (local)" state, the recipient MUST respond with a stream error ([RFC7540], Section 5.4.2) of type STREAM_CLOSED.

GZIPPED_DATA frames can include padding. Padding fields and flags are identical to those defined for DATA frames ([RFC7540], Section 6.1).

2.3. DATA_ENCODING_ERROR

The following new error code is defined:

- "DATA_ENCODING_ERROR" (0xTBA): The endpoint detected that its peer sent a GZIPPED_DATA frame with an invalid encoding.

3. Experimental Status

This extension is classified as an experiment because it alters the base semantics of HTTP/2; a change that, if specified insufficiently or implemented incorrectly, could result in data loss that is hard to detect or diagnose.

[RFC7540], Section 5.5, mandates that "implementations MUST discard frames that have unknown or unsupported types"; so if an endpoint or intermediary mishandles GZIPPED_DATA frames, for example by incorrectly emitting an ACCEPT_GZIPPED_DATA setting or propagating GZIPPED_DATA frames, and those frames are subsequently discarded, data will be lost. There is no reliable mechanism to detect such a loss[*].

The experiment therefore is to explore the robustness of the HTTP/2 ecosystem in the presence of such potential failures.

[*] For some unreliable mechanisms (i.e. not guaranteed to be in use in all cases, and/or requiring inspection of HTTP headers) see:

- Section 8.1.2.6 of [RFC7540], for using the content-length header to detect malformed messages

- [RFC3230], for HTTP instance digests

4. Security Considerations

Further to the Use of Compression in HTTP/2 ([RFC7540], Section 10.6), intermediaries MUST NOT apply compression to DATA frames, or alter the compression of GZIPPED_DATA frames other than decompressing, unless additional information is available that allows the intermediary to identify the source of data. In particular, frames that are not compressed cannot be compressed, and frames that are separately compressed cannot be merged into a single compressed frame.

5. IANA Considerations

This document updates the registries for frame types, settings, and error codes in the "Hypertext Transfer Protocol (HTTP) 2 Parameters" section.

5.1. HTTP/2 Frame Type Registry Update

This document updates the "HTTP/2 Frame Type" registry ([RFC7540], Section 11.2). The entries in the following table are registered by this document.

Frame Type	Code	Section
GZIPPED_DATA	TBD	Section 2.2

5.2. HTTP/2 Settings Registry Update

This document updates the "HTTP/2 Settings" registry ([RFC7540], Section 11.3). The entries in the following table are registered by this document.

Frame Type	Code	Initial Value	Specification
ACCEPT_GZIPPED_DATA	TBD	0	Section 2.1

5.3. HTTP/2 Error Code Registry Update

This document updates the "HTTP/2 Error Code" registry ([RFC7540], Section 11.4). The entries in the following table are registered by this document.

Name	Code	Description	Specification
DATA_ENCODING_ERROR	TBD	Invalid encoding detected	Section 2.3

6. Acknowledgements

Thanks to Keith Morgan for his advice, input, and editorial contributions.

7. References

7.1. Normative References

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <http://www.rfc-editor.org/info/rfc1952>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

7.2. Informative References

[RFC3230] Mogul, J. and A. Van Hoff, "Instance Digests in HTTP", RFC 3230, DOI 10.17487/RFC3230, January 2002, <http://www.rfc-editor.org/info/rfc3230>.

[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, <http://www.rfc-editor.org/info/rfc7230>.

Appendix A. Changelog

Since -07:

- define "reliable" in the 'experimental' section, and provide pointers to potential workarounds
- remove fragmentation, since the text added no value

Since -06:

- change document title from "Encoded" to "Gzipped"
- improve text under GZIPPED_DATA (Section 2.2)
- clarify that GZIPPED_DATA and DATA can be interleaved
- explain experimental status and risks of broken implementations

Since -05:

- changed ACCEPT_ENCODED_DATA back from a frame to a setting, since it carries a single scalar value now

Since -04:

- reduced encoding options to only gzip (suggested by Martin Thomson)
- remove fragmentation and segment stuff, including reference to 'http2-segments' I-D
- updated HTTP/2 reference from I-D to (freshly published) RFC7230

Since -03:

- added 'identity' encoding; removed 'compress' and 'zlib' (suggested by PHK)
- added SEGMENT flag, for segments that don't continue
- clarified that ACCEPT is for a connection, and ENCODED_DATA is for a stream

- copied "padding" text from HTTP/2 draft

Since -02:

- moved all discussion of fragmentation and segments to its own section

Since -01:

- referenced new draft-kerwin-http2-segments to handle fragmentation

Since -00:

- changed ACCEPT_ENCODED_DATA from a complex setting to a frame
- improved IANA Considerations section (with lots of input from Keith Morgan)

Author's Address

Matthew Kerwin

Email: matthew@kerwin.net.au
URI: http://matthew.kerwin.net.au/