HyBi Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 4, 2012

T. Yoshino
Google, Inc.

April 2, 2012

WebSocket Per-frame Compression
draft-ietf-hybi-websocket-perframe-compression-00

Abstract

This specification defines a general scheme to add per-frame compression functionality to the WebSocket Protocol using its extension mechanism, and one specific compression extension using DEFLATE. In this scheme, the "Application data" part of WebSocket data frames is compressed using specified compression algorithm, and one reserved bit in the WebSocket frame header is allocated to control application of compression for each frame.

Please send feedback to the hybi@ietf.org mailing list.

Status of this Memo

This Internet-Draft is submitted to IETF 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 4, 2012.

Table of Contents

1. Introduction

This section is non-normative.

As well as other protocols, the WebSocket Protocol [RFC6455] can benefit from compression technology. This specification defines a scheme to apply compression algorithms to octets exchanged over the WebSocket Protocol using its extension framework, and then defines one specific compression extension using DEFLATE [RFC1951].

The per-frame compression scheme applies the specified compression algorithm to the octets in the "Application data" part of data frames. It also specifies the use of the RSV1 bit of the WebSocket frame header to indicate whether any compression is applied to the frame or not, so that we can choose to skip frames with incompressible contents without applying extra compression. By specifying extension negotiation and how to transform octets in "Application data", we can define per-frame compression extensions for various compression algorithms based on this scheme.

We also introduce one specific extension in this specification by applying DEFLATE to the scheme. It is called "Per-frame DEFLATE extension". DEFLATE algorithm is widely available as library on various platforms. Overhead it adds for each chunk is small. So, it's chosen for the first example of the compression extension for the WebSocket Protocol. To align the end of compressed data to octet boundary, the extension uses the algorithm described in the Section 2.1 of the PPP Deflate Protocol [RFC1979]. Endpoints can take over the LZ77 sliding window [LZ77] used to build previous frames to get better compression ratio.

The simplest "Sec-WebSocket-Extensions" header in the client's opening handshake to request per-frame DEFLATE extension is the following:

```
Sec-WebSocket-Extensions: deflate-frame
```

The simplest header from the server to accept this extension is the same.

2. Conformance Requirements

Everything in this specification except for sections explicitly marked non-normative is normative.

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].

3. General Per-frame Compression Scheme

This section describes a general scheme to apply a compression algorithm to the contents of WebSocket frames.

This scheme allocates one bit field called "Per-frame compressed" at the RSV1 bit. This bit indicates whether any kind of per-frame compression is applied to the frame or not. Because of this, compression extensions based on this scheme are incompatible with each other.

This scheme operates only on data frames, and only on the "Application data" therein (it does not affect the "Extension data" portion of the "Payload data").

3.1. Sending

To send a frame with the compression applied, an endpoint MUST use the following algorithm.

1. Apply the compression to the "Application data" portion of the frame.

2. Build a frame by putting the resulting octets in the "Application data" portion instead of the original octets. The payload length field of the frame MUST be the sum of the size of the "Extension data" portion and one of these resulting octets. "Per-frame compressed" bit MUST be set to 1.

To send a frame with the compression not applied, an endpoint MUST set "Per-frame compressed" bit of the frame to 0 and send the "Application data" portion as-is without applying any compression.

3.2. Receiving

To receive a frame with "Per-frame compressed" bit set to 1, an endpoint MUST decompress the octets in the "Application data" portion based on the compression algorithm.

An endpoint MUST receive a frame with "Per-frame compressed" bit set to 0 as-is without any compression processing.

4. Per-frame DEFLATE Extension

This section defines one specific compression extension by applying DEFLATE to the scheme described in Section 3.

4.1. Extension Negotiation

The registered extension token for this extension is "deflate-frame".

To request use of per-frame DEFLATE extension, a client MUST include the "deflate-frame" extension token in the "Sec-WebSocket-Extensions" header in its opening handshake.

To accept use of per-frame DEFLATE extension requested by the client, a server MUST include the "deflate-frame" extension token in the "Sec-WebSocket-Extensions" header in its opening handshake.

An endpoint MAY attach one or more extension parameters as defined below to the extension token.

Maximum LZ77 sliding window size
: An endpoint MAY attach "max_window_bits" extension parameter to limit the LZ77 sliding window size that the other peer uses to build frames. This parameter MUST have an integer value in the range between 8 to 15 indicating the base-2 logarithm of the LZ77 sliding window size. An endpoint that received this parameter MUST NOT use LZ77 sliding window size greater than this value to build frames.

Disallow compression context takeover
: An endpoint MAY attach "no_context_takeover" extension parameter to disallow the other peer to take over the LZ77 sliding window used to build previous frames. This parameter has no value. An endpoint that received this parameter MUST use an empty LZ77 sliding window to build every frame.

A server MUST ignore any unknown extension parameter attached to "deflate-frame" extension token in the client's opening handshake.

A client MUST _Fail the WebSocket Connection_ if any unknown extension parameter is attached to "deflate-frame" extension token in the server's opening handshake.

Once per-frame DEFLATE extension is accepted, both endpoints MUST use the algorithm described in Section 4.2 to exchange frames.

4.2. Application Data Transformation

This extension transforms the "Application data" portion by using the scheme described in Section 3 with DEFLATE as follows.

4.2.1. Compression

An endpoint MUST use the following algorithm to compress the "Application data" portion.

1. Apply DEFLATE [RFC1951] to all the octets. Multiple blocks MAY be used. Any type of block MAY be used. Both block with "BFINAL" set to 0 and 1 MAY be used.

2. If the resulting data does not end with an empty block with no compression ("BTYPE" set to 0), append an empty block with no compression to the tail.

3. Remove 4 octets (that are 0x00 0x00 0xff 0xff) from the tail.

An endpoint MUST NOT use LZ77 sliding window greater than 32,768 byte to build frames to send.

If an endpoint received the "max_window_bits" extension parameter on opening handshake, it MUST NOT use LZ77 sliding window greater than the "max_window_bits"-th power of 2 byte to build frames to send.

Unless it's prohibited by the other peer by the "no_context_takeover" extension parameter on opening handshake, an endpoint MAY take over the LZ77 sliding window used to build the last frame to send with DEFLATE applied.

4.2.2. Decompression

An endpoint MUST use the following algorithm to decompress the "Application data" portion.

1. Append 4 octets of 0x00 0x00 0xff 0xff to the tail.

2. Decompress the resulting octets using DEFLATE.

Unless an endpoint sent the "max_window_bits" extension parameter on opening handshake, the endpoint MUST use 32,768 byte LZ77 sliding window to decode received frames.

If an endpoint sent the "max_window_bits" extension parameter on opening handshake, it MAY reduce the size of LZ77 sliding window to decode received frames down to the "max_window_bits"-th power of 2

byte.

Unless the endpoint sent the "no_context_takeover" extension parameter on opening handshake, an endpoint MUST take over the LZ77 sliding window used to decode the last received frame with DEFLATE applied.

4.2.3. Examples

This section is non-normative.

These are examples of resulting data after applying the algorithm above.

- "Hello" in one compressed block
  - 0xf2 0x48 0xcd 0xc9 0xc9 0x07 0x00

  "Hello" in one compressed block in the next frame
  - 0xf2 0x00 0x11 0x00 0x00
- "Hello" in one block with no compression
  - 0x00 0x05 0x00 0xfa 0xff 0x48 0x65 0x6c 0x6c 0x6f 0x00
- "Hello" in one block with "BFINAL" set to 1
  - 0xf3 0x48 0xcd 0xc9 0xc9 0x07 0x00 0x00
- "He" and "llo" in separate blocks
  - 0xf2 0x48 0x05 0x00 0x00 0x00 0xff 0xff 0xca 0xc9 0xc9 0x07 0x00

4.3. Intermediaries

Intermediaries MAY decompress and/or compress frames when they forward them under constraints negotiated on opening handshake as described in Section 4.2.

4.4. Implementation Note

This section is non-normative.

On common software development platforms, the operation of aligning compressed data to octet boundary using an empty block with no compression is available as library. For example, Zlib [Zlib] does

this when "Z_SYNC_FLUSH" is passed to deflate function.

To get sufficient compression ratio, LZ77 sliding window size of 1,024 or more is recommended.

5. Security Considerations

There's no security concern for now.

6. IANA Considerations

6.1. Registration of the "deflate-frame" WebSocket Extension Name

This section describes a WebSocket extension name registration in the WebSocket Extension Name Registry. [RFC6455].

Extension Identifier
deflate-frame

Extension Common Name
WebSocket Per-frame DEFLATE

Extension Definition
Section 4.1 and Section 4.2 of this document.

Known Incompatible Extensions
None

The "deflate-frame" token is used in the "Sec-WebSocket-Extensions" header in the WebSocket opening handshake to negotiate use of per-frame DEFLATE compression extension.

6.2. Registration of the "Per-frame compressed" WebSocket Framing Header Bit

This section describes a WebSocket framing header bit registration in the WebSocket Framing Header Bits Registry. [RFC6455]

Header Bit
RSV1

Common Name
Per-frame compressed

Meaning
Compression is applied to the frame or not.

Reference
Section 3 of this document.

The "Per-frame compressed" framing header bit is used to indicate whether any negotiated per-frame compression extension applied compression to the "Application data" portion of the frame or not.

7. Acknowledgements

Special thanks to Patrick McManus who wrote up the initial specification of DEFLATE based compression extension for the WebSocket Protocol which I referred to write this specification.

8. References

8.1. Normative References

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[LZ77] Ziv, J. and A. Lempel, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343.

8.2. Informative References

[RFC1951] Deutsch, P., "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, May 1996.

[RFC1979] Woods, J., "PPP Deflate Protocol", RFC 1979, August 1996.

[Zlib] Gailly, J. and M. Adler, "Zlib", <http://zlib.net/>.

Author's Address

Takeshi Yoshino
Google, Inc.

Email: tyoshino@google.com