HyBi Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 11, 2012

T. Yoshino
Google, Inc.

May 10, 2012

WebSocket Per-frame Compression
draft-ietf-hybi-websocket-perframe-compression-03

Abstract

This specification defines a WebSocket extension that adds per-frame compression functionality to the WebSocket Protocol. It compresses the "Application data" portion of WebSocket data frames using specified compression algorithm. One reserved bit RSV1 in the WebSocket frame header is allocated to control application of compression for each frame. This specification provides one compression method available for the extension using DEFLATE.

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 November 11, 2012.

Table of Contents

1. Introduction

This section is non-normative.

As well as other communication protocols, the WebSocket Protocol [RFC6455] can benefit from compression technology. This specification defines a WebSocket extension that applies a compression algorithm to octets exchanged over the WebSocket Protocol using its extension framework. This extension negotiates what compression method to use on opening handshake, and then compresses to the octets in the "Application data" portion of data frames using the method. We can apply this extension to various compression algorithms by specifying how to negotiate parameters and transform "Application data". A client may offer multiple compression methods on opening handshake, and then the server chooses one from them. This extension uses the RSV1 bit of the WebSocket frame header to indicate whether the frame is compressed or not, so that we can choose to skip frames with incompressible contents without applying extra compression.

This specification provides one specific compression method for this extension "deflate" which is based on DEFLATE [RFC1951]. We chose DEFLATE since it's widely available as library on various platforms and the overhead it adds for each chunk is small. To align the end of compressed data to octet boundary, this method 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. For resource-limited devices, method parameters to limit the usage of memory for compression context are provided.

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

```
Sec-WebSocket-Extensions: perframe-compress; method=deflate
```

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. Extension Negotiation

The registered extension token for this extension is "perframe-compress".

To request use of the Per-frame Compression Extension, a client MUST include an element with the "perframe-compress" extension token as its extension identifier in the "Sec-WebSocket-Extensions" header in its opening handshake. The element MUST contain exactly one extension parameter named "method". The value of the "method" extension parameter is a list of compression method descriptions, ordered by preference. Each compression method description has a method name and optional method parameters. The grammar of the list is "requested-method-list" defined in the following ABNFs.

```
requested-method-list = 1#method-desc
method-desc = method-name *(";" method-param)
method-name = token
method-param = token ["=" (token | quoted-string)]
```

To accept use of the Per-frame Compression Extension, a server MUST include an element with the "perframe-compress" extension token as its extension identifier in the "Sec-WebSocket-Extensions" header in its opening handshake. The element MUST contain exactly one extension parameter named "method". The value of the "method" extension parameter MUST be a compression method description. The compression method description MUST be a valid response for any of the methods listed by the client in its opening handshake. Its grammar is "accepted-method-list" defined in the following ABNF.

```
accepted-method = method-desc
```

The value of the "method" parameter MUST be quoted by using "quoted-string" syntax if it doesn't conform to token syntax.

Once the extension is accepted, both endpoints MUST use the algorithm described in Section 4 to exchange frames.

3.1. Negotiation Example

This section is non-normative.

These are "Sec-WebSocket-Extensions" header value examples that negotiate the per-frame compression extension.

- Request foo method. Since foo matches token syntax, it doesn't need to be quoted.

```
perframe-compress; method=foo
```

- Request foo method with a parameter x with 10 as its value. Since the method parameter value contains a semicolon, it doesn't match token syntax. Quotation is needed.

```
perframe-compress; method="foo; x=10"
```

- Request foo method and bar method. Since the method parameter value contains a comma, it doesn't match token syntax. Quotation is needed.

```
perframe-compress; method="foo, bar"
```

- Request foo method with parameter x with "Hello World" (quotation for clarification) as its value and bar method. Since "Hello World" contains a space, it needs to be quoted. Since quoted "Hello World" contains double quotations and a space, it needs to be quoted again.

```
perframe-compress; method="foo; x=\"Hello World\", bar"
```

4. Framing

This section describes how to apply the negotiated compression method to the contents of WebSocket frames.

This extension allocates the RSV1 bit of the WebSocket header and names it the "Per-frame Compressed" bit. Any extension requiring the use of the RSV1 bit is incompatible with this extension. This bit indicates whether the compression is applied to the frame or not. Frames with the "Per-frame Compressed" bit set are called "per-frame compressed frames". They have compressed data in their "Application data" portion. Frames with the bit unset are called "per-frame uncompressed frames". They have uncompressed data in their "Application data" portion.

This extension operates only on data frames. This extension doesn't modify the "Extension data" portion.

4.1. Sending

To send a frame as a per-frame compressed frame, an endpoint MUST use the following algorithm.

1. Compress the octets in "Application data" portion of the frame using the compression method.

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 the size of the resulting octets.

3. Set the "Per-frame Compressed" bit of the frame to 1.

To send a frame as a per-frame uncompressed frame, an endpoint MUST set "Per-frame Compressed Bit" of the frame to 0. "Application data" portion MUST be sent as-is without applying the compression method.

4.2. Receiving

To receive a per-frame compressed frame, an endpoint MUST decompress the octets in the "Application data" portion.

An endpoint MUST receive a per-frame uncompressed frame as-is without decompression.

5. DEFLATE method

This section defines a method named "deflate" for this extension that compresses "Application data" using DEFLATE [RFC1951] and byte boundary alignment method introduced in [RFC1979].

5.1. Method Parameters

An endpoint MAY include one or more method parameters in the method description as defined below.

Maximum LZ77 sliding window size
: An endpoint MAY attach the "max_window_bits" method 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 the size specified by this parameter to build frames.

Disallow compression context takeover
: An endpoint MAY attach the "no_context_takeover" method 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 reset its LZ77 sliding window for sending to empty for each frame.

A server MUST ignore any unknown method parameter in the "deflate" method description in the client's opening handshake.

A client MUST _Fail the WebSocket Connection_ if there is any unknown method parameter in the "deflate" method description in the server's opening handshake.

5.2. Application Data Transformation

5.2.1. Compression

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

1. Compress all the octets in the "Application data" portion using DEFLATE. 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 an LZ77 sliding window greater than 32,768 bytes to build frames to send.

If an endpoint received the "max_window_bits" method parameter, the endpoint MUST NOT use an LZ77 sliding window greater than the "max_window_bits"-th power of 2 bytes to build frames to send.

Unless an endpoint received the "no_context_takeover" method parameter, the endpoint MAY take over the LZ77 sliding window used to build the last per-frame compressed frame.

If an endpoint received the "no_context_takeover" method parameter, the endpoint MUST reset its LZ77 sliding window for sending to empty for each frame.

5.2.2. 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" method parameter, the endpoint MUST use a 32,768 byte LZ77 sliding window to decompress received frames.

If an endpoint sent the "max_window_bits" method parameter, the endpoint MAY reduce the size of the LZ77 sliding window to decompress received frames down to the "max_window_bits"-th power of 2 bytes.

Unless an endpoint sent the "no_context_takeover" method parameter, the endpoint MUST take over the LZ77 sliding window used to parse the last per-frame compressed frame.

If an endpoint sent the "no_context_takeover" method parameter, the endpoint MAY reset its LZ77 sliding window for receiving to empty for each frame.

5.2.3. 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

5.3. Intermediaries

When intermediaries forward frames, they MAY decompress and/or compress the frames according to the constraints negotiated during the opening handshake of the connection(s).

5.4. Implementation Notes

This section is non-normative.

On most common software development platforms, the operation of aligning compressed data to byte boundaries using an empty block with no compression is available as a 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.

6. Security Considerations

There are no security concerns for now.

7. IANA Considerations

7.1. Registration of the "perframe-compress" WebSocket Extension Name

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

Extension Identifier
perframe-compress

Extension Common Name
WebSocket Per-frame Compression

Extension Definition
This document.

Known Incompatible Extensions
None

The "perframe-compress" token is used in the "Sec-WebSocket-Extensions" header in the WebSocket opening handshake to negotiate use of the Per-frame Compression Extension.

7.2. 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
The frame is compressed or not.

Reference
Section 4 of this document.

The "Per-frame Compressed" framing header bit is used to indicate whether the "Application data" portion of the frame is compressed by the Per-frame Compression Extension or not.

7.3. WebSocket Per-frame Compression Method Name Registry

This specification creates a new IANA registry for names of compression methods to be used with the WebSocket Per-frame Compression Extension in accordance with the principles set out in [RFC5226].

As part of this registry, IANA maintains the following information:

Method Identifier
: The identifier of the method, as will be used in the method description as defined Section 3 of this specification. The value must conform to the method-name ABNF as defined in Section 3 of this specification.

Method Common Name
: The name of the method, as the method is generally referred to.

Method Definition
: A reference to the document in which the method being used with this extension is defined.

WebSocket Per-frame Compression method names are to be subject to the "First Come First Served" IANA registration policy [RFC5226].

IANA has added initial values to the registry as follows.

Identifier	Common Name	Definition
deflate	DEFLATE	This document

8. Acknowledgements

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

9. References

9.1. Normative References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[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.

9.2. 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