

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 13, 2013

Y. Nir
Check Point
November 9, 2012

HTTP/2.0 Discussion: Compact Header Encoding draft-nir-httpbis-che-01

Abstract

This document proposes an alternative encoding for HTTP headers. This encoding is considerably more compact than the uncompressed textual encoding in HTTP/1.1 and current HTTP/2.0 draft.

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 May 13, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Internet-Draft

compact-header

November 2012

Table of Contents

1.	Introduction	3
1.1.	Conventions Used in This Document	3
2.	The Binary Encoding	3
2.1.	Flags	4
2.2.	Short Type-Value	4
2.3.	Long Type-Value	4
2.4.	Type-Length-Value	4
3.	Header Encoding	4
4.	Custom Headers and Custom Enumerations	5
5.	Default Headers	6
6.	IANA Considerations	6
7.	Security Considerations	7
8.	Changes from Previous Versions	7
9.	Normative References	7
Appendix A.	Additional Examples	7
	Author's Address	10

Internet-Draft

compact-header

November 2012

[1.](#) Introduction

HTTP/1.x and the current draft of HTTP/2.0 encode headers using text labels and text values. HTTP/2.0 attempts to make this more efficient by compressing the textual headers. This proposes a binary-only alternative.

[1.1.](#) Conventions Used in This Document

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.](#) The Binary Encoding

The header block is formatted as follows:

+-----+		
1	version	8
+-----+		
	Flags (8)	Length (24 bits)
+-----+		
X	Stream-ID (31bits)	
+-----+		
	Sequence of headers	
+-----+		

The sequence of headers is just a list of the headers in one of 5 formats:

- o Flags - These are headers with no associated data. The only information they convey is by their mere presence.
- o Short Type-Value - where the header is associated with a 16-bit value.
- o Long Type-Value - where the header is associated with a 32-bit

value.

- o Type-Length-Value - where the length is specified in the header.

All formats include a 16-bit header identifier (see below), and those identifiers will be allocated through a new IANA registry (see [Section 6](#)). The header identifier specifies which format applies.

Nir

Expires May 13, 2013

[Page 3]

Internet-Draft

compact-header

November 2012

[2.1.](#) Flags

These headers are 16-bit numbers containing the header identifier.

```
+-----+
|header identifier|
+-----+
```

[2.2.](#) Short Type-Value

These headers have the 16-bit identifier, and also the 16-bit value.

```
+-----+
|header identifier| Value      |
+-----+
```

[2.3.](#) Long Type-Value

These headers have the 16-bit identifier, and also a 32-bit value.

```
+-----+
|header identifier| Value      |
+-----+
| Value (cont)    |
+-----+
```

[2.4.](#) Type-Length-Value

These headers have the 16-bit identifier, and also a 24-bit length

field, and a value of variable length.

+-----+		
header identifier	Length	
+-----+		
Length	Value...	
+-----+		

[3.](#) Header Encoding

The encoding of each header is specified in the specification that describes it. For convenience, this document describes some common encodings. Specification writers SHOULD use these formats whenever they are appropriate.

Unsigned integer numbers can be represented by either the short or long type-value, depending on their range. Cache ages measured in seconds, such as in HSTS should use the long type-value, whereas a

header specifying an age in days should probably use a short type-value. Either way, the encoding can be called "INT".

Headers that hold an enumeration (such as Method) SHOULD use a short type-value, and SHOULD reserve one value (0xffff) for custom values.

Time values should be encoded as strings using the [RFC3339](#) format.

Strings such as names should use the TLV format, and SHOULD be encoded as UTF-8. String headers should be specified by their encoding, so "UTF8", or "ASCII".

For headers with multiple values, the general format is always TLV, and the specification should list their type as either of three things:

- o Short values - a list of 16-bit values
- o Short strings - a sequence of strings, each prefixed by a 1-octet length field.
- o Long strings - a sequence of strings, each prefixed by 1 2-octet length field.

4. Custom Headers and Custom Enumerations

For each type of header, a range will be allocated for experimental and custom headers. To avoid collisions, we define here a special header to denote what kind of header this is. The header has identifier 49160 (0xC008), so it is TLV-formatted, and its value is formatted as follows:

Custom header format

```
+-----+
|header identifier| Flags  | Name... |
+-----+
```

For example, suppose [draft-nir-httpbis-copyright-notice](#) defines a header that contains a copyright notice for the content. I will use 65530 (0xFFFA). Note that the two headers don't have to be consecutive. If the sender knows that the receiver recognizes this header with this identifier, the Custom header MAY be omitted.

Nir

Expires May 13, 2013

[Page 5]

Internet-Draft

compact-header

November 2012

Custom and Copyright Headers

```
C0 08 00 00 0C FF FA 00 4C 4F 50 59 52 49 47 48 |.....COPYRIGHT|
54 FF FA 00 00 43 6f 70 79 72 69 67 68 74 20 28 |T....Copyright (|
63 29 20 32 30 31 32 20 49 45 54 46 20 54 72 75 |c) 2012 IETF Tru|
73 74 20 61 6e 64 20 74 68 65 20 70 65 72 73 6f |st and the perso|
6e 73 20 69 64 65 6e 74 69 66 69 65 64 20 61 73 |ns identified as|
20 74 68 65 20 64 6f 63 75 6d 65 6e 74 20 61 75 | the document au|
74 68 6f 72 73 2e 20 41 6c 6c 20 72 69 67 68 74 |thors. All right|
73 20 72 65 73 65 72 76 65 64 2e                |s reserved.      |
```

For custom values in enumerations we define the Custom-Value header with identifier 49161 (0xC009), where the content is the string name of the custom value. This header MUST follow the enumeration header.

[5.](#) Default Headers

Many requests share a lot of their headers. For example, the Cookie, User-Agents, Host, Connection, and Accept* headers pretty much remain constant between consecutive requests.

To make the per-stream Headers block even smaller, we allow a default Headers block. This block is distinguished by having Stream-ID fixed to all zeros. Additionally a new flag is defined:

0x02 = FLAG_UPD - marks that this frame updates the default headers

Every subsequent request is deemed to include all the default headers, except where such headers are overridden by that request. The default headers are persistent for the connection.

A default HEADERS with the FLAG_UPD flag cleared replaces the default headers for this connection. A default HEADERS block with the FLAG_UPD set updates the default headers for this connection by replacing those that had already been set, and adding those that had not been set. This is useful for example, if a cookie had been set by the server. The only way to delete a header from the default headers is by replacement - a default HEADERS block with FLAG_UPD cleared.

[6.](#) IANA Considerations

IANA is requested to set up a new registry of header identifiers. The value is 16-bit, and the range is partitioned as follows:

- o 0-16383 - these values are allocated to flag headers, where the format is as in [Section 2.1](#)
- o 16384-32767 - these values are allocated to short type-value headers, where the format is as in [Section 2.2](#)
- o 32768-49151 - these values are allocated to long type-value headers, where the format is as in [Section 2.3](#)
- o 49152-65535 - these values are allocated to type-length-value headers, where the format is as in [Section 2.4](#)

The ending quarter of each range shall be reserved for experimental and custom usage, and shall not be allocated by standards action. For example, the range 45056-49151 will be reserved for experimental and custom long type-value headers.

[7.](#) Security Considerations

There are no security considerations for this draft.

[8.](#) Changes from Previous Versions

First version

[9.](#) Normative References

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

[Appendix A.](#) Additional Examples

NOTE: Most of the below examples were shamelessly copied from [draft-snell-httpbis-bohe-01](#).

Assuming the following (intentionally incomplete) header registrations:

HTTP Header	ID	Hex	Format
Version	16384	4000	Major.Minor in 16-bit
Method	16385	4001	Enumeration
Host	49152	c000	UTF8
Path (Request URI)	49153	c001	UTF8
Status	16386	4002	uint16
Status-Text	49386	c0ea	UTF8
Content-Length	32768	8000	uint32
Content-Type	49154	c002	ASCII
Expect	16387	4003	uint16
Last-Modified	49155	c003	RFC3339
ETag	49156	c004	sequence of short strings
If-None-Match	49157	c005	sequence of short strings
Allow	49158	c006	sequence of uint16
Do-Not-Track	58	003a	flag

And the following values representing known HTTP Methods:

Method	Value
GET	1
POST	2
PUT	3
DELETE	4
PATCH	5
HEAD	6
OPTIONS	7
CONNECT	8

Here is what the encoding looks like:

Version Header:

40 00 02 00 |@...|

Method Header (GET Request)

40 01 00 01 |@...|

Method Header (PATCH Request)

40 01 00 05 |@...|

Method Header (Custom "FOO" Method)

```
40 01 FF FF C0 09 00 03 46 4F 4F      |@.....FOO      |
```

Do Not Track

```
00 3A      |.:|
```

Host Header:

```
C0 00 00 00 0F 77 77 77 2e 65 78 61 6d 70 6c 65 |.....www.example|
2e 6f 72 67      |.org      |
```

HTTP Response Status ("200 OK") as two separate headers, one containing the status code, the other containing the status text:

```
40 02 00 C8 C0 EA 00 00 02 4F 4B      |@.....OK      |
```

Content-Length Header (value encoded as uint32):

```
80 00 00 00 00 C8      |.....|
```

Content-Type Header (although maybe it should become an enum:

```
C0 02 00 00 0A 69 6d 61 67 65 2f 6a 70 65 67    |.....image/jpeg |
```

Expect Header (Expect: 100):

```
40 03 00 64      |...d|
```

Last-Modified (Using [RFC3339](#) Format):

```
C0 03 00 00 19 32 30 31 32 2d 30 38 2d 30 31 54 |.....2012-08-01T|
30 34 3a 32 33 3a 31 32 2e 31 32 33 34 5a      |04:23:12.1234Z  |
```

ETag (Strong Entity-Tag, String-format):

```
C0 04 00 00 06 05 61 62 63 64 65      |.....abcde      |
```

If-None-Match (Multiple values)

```
C0 05 00 00 0C 05 61 62 63 64 65 05 61 62 63 64 |.....abcde.abcd|
66      |f      |
```

Internet-Draft

compact-header

November 2012

Allow (GET, POST, F00):

```
C0 06 00 00 06 00 01 00 02 FF FF C0 09 00 00 04 | .....|
03 46 4f 4f                                     |.F00  |
```

Author's Address

Yoav Nir
Check Point Software Technologies Ltd.
5 Hasolelim st.
Tel Aviv 67897
Israel

Email: ynir@checkpoint.com

Nir

Expires May 13, 2013

[Page 10]