HTTPbis Working Group
Internet-Draft
Intended status: Informational
Expires: April 24, 2014

R. Peon
Google, Inc
H. Ruellan
Canon CRF

October 21, 2013

HPACK - Header Compression for HTTP/2.0
draft-ietf-httpbis-header-compression-04

Abstract

This document describes HPACK, a format adapted to efficiently represent HTTP header fields in the context of HTTP/2.0.

Editorial Note (To be removed by RFC Editor)

Discussion of this draft takes place on the HTTPBIS working group mailing list (ietf-http-wg@w3.org), which is archived at <http://lists.w3.org/Archives/Public/ietf-http-wg/>.

Working Group information and related documents can be found at <http://tools.ietf.org/wg/httpbis/> (Wiki) and <https://github.com/http2/http2-spec> (source code and issues tracker).

The changes in this draft are summarized in Appendix A.1.

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 April 24, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document describes HPACK, a format adapted to efficiently represent HTTP header fields in the context of HTTP/2.0 (see [HTTP2]).

2. Overview

In HTTP/1.X (see [HTTP-p1]), header fields are sent without any form of compression. As web pages have grown to include dozens to hundreds of requests, the redundant header fields in these requests now pose a problem of measurable latency and unnecessary bandwidth (see [PERF1] and [PERF2]).

SPDY [SPDY] initially addressed this redundancy by compressing header fields with Deflate, which proved very effective at eliminating the redundant header fields. However, that aproach exposed a security risk as demonstrated by the CRIME [CRIME].

In this document, we propose a new compressor for header fields which eliminates redundant header fields, is not vulnerable to CRIME style attacks, and which also has a bounded memory cost for use in constrained environments.

2.1. Outline

The HTTP header field encoding described in this document is based on a header table that map (name, value) pairs to index values. Header tables are incrementally updated during the HTTP/2.0 session.

The encoder is responsible for deciding which header fields to insert as new entries in the header table. The decoder then does exactly what the encoder prescribes, ending in a state that exactly matches the encoder's state. This enables decoders to remain simple and understand a wide variety of encoders.

As two consecutive sets of header fields often have header fields in common, each set of header fields is coded as a difference from the previous set of header fields. The goal is to only encode the changes (header fields present in one of the set and not in the other) between the two sets of header fields.

Examples illustrating the use of these different mechanisms to represent header fields are available in Appendix E.

3. Header Field Encoding

3.1. Encoding Concepts

The encoding and decoding of header fields relies on some components and concepts:

Header Field: A key, value pair. HPACK allows a header field value to be either a value as specified by HTTP/1.X (see [HTTP-p1]), or a NULL-separated ordered list of HTTP/1.X values.

Header Table: The header table (see Section 3.1.2) is a component used to associate stored header fields to index values. The data stored in this table is in first-in, first-out order.

Static Table: The static table (see Appendix B) is a component used to associate static header fields to index values. This data is ordered, read-only, always accessible, and may be shared amongst all encoding contexts.

Reference Set: The reference set (see Section 3.1.3) is a component containing an unordered set of references to entries in the header

table or static table. This is used for the differential encoding of a new header set.

Header Set: A header set is a potentially ordered group of header fields that are encoded jointly. A complete set of key-value pairs contained in a HTTP request or response is a header set.

Header Field Representation: A header field can be represented in encoded form either as a literal or as an index (see Section 3.1.4).

Header Block: The entire set of encoded header field representations which, when decoded, yield a complete header set.

Header Field Emission: When decoding a set of header field representations, some operations emit a header field (see Section 3.1.5). Emitted headers can be safely passed to the upper processing layers as part of the current Header Set.

3.1.1. Encoding Context

The set of mutable structures used within an encoding context include a header table and a reference set. Everything else is either immutable or conceptual.

Using HTTP, messages are exchanged between a client and a server in

both direction. To keep the encoding of header fields in each direction independent from the other direction, there is one encoding context for each direction.

The header fields contained in a PUSH_PROMISE frame sent by a server to a client are encoded within the same context as the header fields contained in the HEADERS frame corresponding to a response sent from the server to the client.

3.1.2. Header Table

A header table consists of a list of header fields maintained in first-in, first-out order. The first and newest entry in a header table is always at index 0, and the oldest entry of a header table is at the index len(header table)-1.

The header table is initially empty.

There is typically no need for the header table to contain duplicate entries. However, duplicate entries MUST NOT be treated as an error by a decoder.

The encoder decides how to update the header table and as such can control how much memory is used by the header table. To limit the memory requirements on the decoder side, the header table size is strictly bounded (see Section 3.3.1).

The header table is updated during the processing of a set of header field representations (see header field representation processing (Section 3.2.1).

3.1.3. Reference Set

A reference set is an unordered set of references to entries either within the header table or the static table.

The reference set is initially empty.

The reference set is updated during the processing of a set of header field representations (see header field representation processing (Section 3.2.1).

The reference set enables differential encoding, whereby only differences between the previous header set and the current header set need to be encoded.

When an entry is evicted from the header table, if it was referenced from the reference set, its reference is removed from the reference

set.

3.1.4. Header Field Representation

An encoded header field can be represented either as a literal or as an index.

Literal Representation: A literal representation defines a new

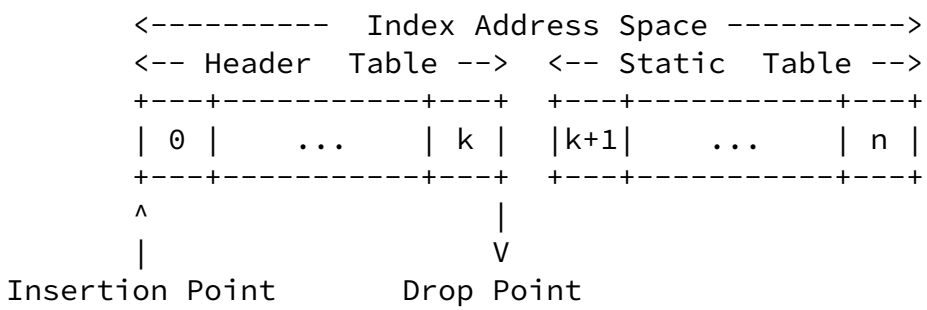
header field. The header field name is represented either literally or as a reference to an entry of the header table. The header field value is represented literally.

Two different literal representations are provided:

* A literal representation that does not add the header field to the header table (see Section 4.3.1).

* A literal representation that adds the header field as a new entry at the beginning of the header table (see Section 4.3.2).

Indexed Representation: The indexed representation defines a header field as a reference to an entry in either the header table or the static table(see Section 4.2).



Index Address Space

Indices between 0 and len(header table)-1, inclusive, refer to elements in the header table, with index 0 referring to the beginning of the table.

Indices between len(header table) and len(header table)+ len(static table)-1, inclusive, refer to elements in the static table, where the index len(header table) refers to the first entry in the static table.

Any other indices MUST be treated as erroneous, and the

compression context considered corrupt and unusable.

3.1.5. Header Field Emission

The emission of a header field is the process of passing that header field to the application, so that the application can process and react to header field data.

By emitting header fields instead of emitting header sets, the decoder can be implemented in a streaming way, and as such must only keep in memory the header table and the reference set. This bounds the amount of memory used by the decoder, even in presence of a very large set of header fields. The management of memory for handling very large sets of header fields can therefore be deferred to the application.

When a header field is a NULL-separated list of values, each value within the list MAY be emitted separately, with the same header field name, and the order of emission MUST be the order of appearance in the list.

3.2. Header Block Decoding

The processing of a header block to obtain a header set is defined in this section. To ensure that the decoding will successfully produce a header set, a decoder MUST obey the following rules.

3.2.1. Header Field Representation Processing

All the header field representations contained in a header block are processed in the order in which they are presented, as specified below.

An _indexed representation_ corresponding to an entry _present_ in the reference set entails the following actions:

- The reference to the entry is removed from the reference set.

An _indexed representation_ corresponding to an entry _not present_ in the reference set entails the following actions:

- If referencing an element of the static table:
  - The header field corresponding to the referenced entry is emitted.

    * The referenced static entry is added to the header table.

    * If the new entry fits within the header table, a reference to the header table entry is added to the reference set.

- If referencing an element of the header table:

    * The header field corresponding to the referenced entry is emitted.

    * The referenced header table entry is added to the header table.

A _literal representation_ that is _not added_ to the header table entails the following action:

- The header field is emitted.

A _literal representation_ that is _added_ to the header table entails the following actions:

- The header field is inserted at the beginning of the header table.
- A reference to the new entry is added to the reference set.
- The header field is emitted.

3.2.2. Reference Set Emission

Once all the representations contained in a header block have been processed, the header fields referenced in the reference set which have not previously been emitted during this processing are emitted.

3.2.3. Header Set Completion

Once all of the header field representations have been processed, and the remaining items in the reference set have been emitted, the header set is complete.

3.3. Header Table Management

3.3.1. Maximum Table Size

To limit the memory requirements on the decoder side, the size of the the header table is bounded. The size of the header table MUST stay lower than or equal to the value of the HTTP/2.0 setting SETTINGS_HEADER_TABLE_SIZE (see [HTTP2]).

The size of the the header table is the sum of the size of its

entries.

The size of an entry is the sum of its name's length in bytes (as defined in Section 4.1.2), of its value's length in bytes (Section 4.1.2) and of 32 bytes.

The lengths are measured on the non-encoded entry name and entry value (for the case when a Huffman encoding is used to transmit string values).

The 32 bytes are an accounting for the entry structure overhead. For example, an entry structure using two 64-bits pointers to reference the name and the value and the entry, and two 64-bits integer for counting the number of references to these name and value would use 32 bytes.

3.3.2. Entry Eviction When Header Table Size Changes

Whenever an entry is evicted from the header table, any reference to that entry contained by the reference set is removed.

Whenever SETTINGS_HEADER_TABLE_SIZE is made smaller, entries are evicted from the end of the header table until the size of the header table is less than or equal to SETTINGS_HEADER_TABLE_SIZE.

The eviction of an entry from the header table causes the index of the entries in the static table to be reduced by one.

3.3.3. Entry Eviction when Adding New Entries

Whenever a new entry is to be added to the table, any name referenced by the representation is cached, and then entries are evicted from the end of the header table until the size of the header table is less than or equal to SETTINGS_HEADER_TABLE_SIZE - new entry size, or until the table is empty.

If the size of the new entry is less than or equal to SETTINGS_HEADER_TABLE_SIZE, that entry is added to the table. It is not an error to attempt to add an entry that is larger than

SETTINGS_HEADER_TABLE_SIZE.

4. Detailed Format

4.1. Low-level representations

4.1.1. Integer representation

Integers are used to represent name indexes, pair indexes or string lengths. To allow for optimized processing, an integer representation always finishes at the end of a byte.

An integer is represented in two parts: a prefix that fills the current byte and an optional list of bytes that are used if the integer value does not fit within the prefix. The number of bits of the prefix (called N) is a parameter of the integer representation.

The N-bit prefix allows filling the current byte. If the value is small enough (strictly less than 2^N-1), it is encoded within the N-bit prefix. Otherwise all the bits of the prefix are set to 1 and the value is encoded using an unsigned variable length integer [1] representation. N is always between 1 and 8 bits. An integer starting at a byte-boundary will have an 8-bit prefix.

The algorithm to represent an integer I is as follows:

```
If I < 2^N - 1, encode I on N bits
Else
    encode 2^N - 1 on N bits
    I = I - (2^N - 1)
    While I >= 128
         Encode (I % 128 + 128) on 8 bits
         I = I / 128
    encode (I) on 8 bits
```

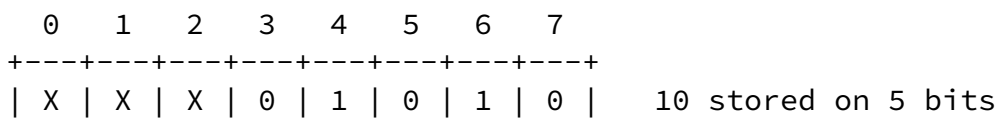
This integer representation allows for values of indefinite size. It is also possible for an encoder to send a large number of zero values, which can waste bytes and could be used to overflow integer

values. Excessively large integer encodings - in value or octet length - MUST be treated as a decoding error. Different limits can be set for each of the different uses of integers, based on implementation constraints.

4.1.1.1. Example 1: Encoding 10 using a 5-bit prefix

The value 10 is to be encoded with a 5-bit prefix.

- 10 is less than 31 (= 2^5 - 1) and is represented using the 5-bit prefix.



4.1.1.2. Example 2: Encoding 1337 using a 5-bit prefix

The value I=1337 is to be encoded with a 5-bit prefix.

1337 is greater than 31 (= 2^5 - 1).

The 5-bit prefix is filled with its max value (31).

I = 1337 - (2^5 - 1) = 1306.

I (1306) is greater than or equal to 128, the while loop body executes:

I % 128 == 26

26 + 128 == 154

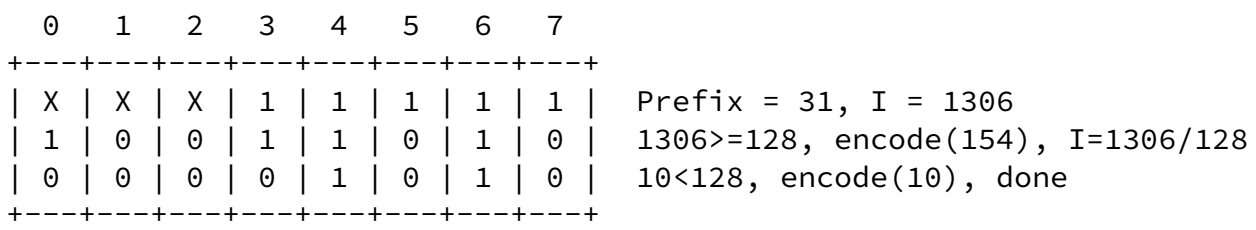
154 is encoded in 8 bits as: 10011010

I is set to 10 (1306 / 128 == 10)

I is no longer greater than or equal to 128, the while loop terminates.

I, now 10, is encoded on 8 bits as: 00001010

The process ends.



4.1.2. String Literal Representation

Header field names and header field values are encoded as sequences of bytes. A header field name or a header field value is encoded in three parts:

1. One bit, H, indicating whether or not the bytes are huffman encoded.

2. The number of bytes required to hold the result of the next step, represented as a variable-length-quantity (Section 4.1.1), starting with a 7-bit prefix immediately following the first bit.

3. The encoded data of the string:

    1. If H is true, then the the encoded string data is the bitwise concatenation of the canonical [CANON]huffman code [HUFF] corresponding to each character of the data, followed by

between 0-7 bits of padding.

2. If H is false, then the encoded string is the bytes of the field value without modification.

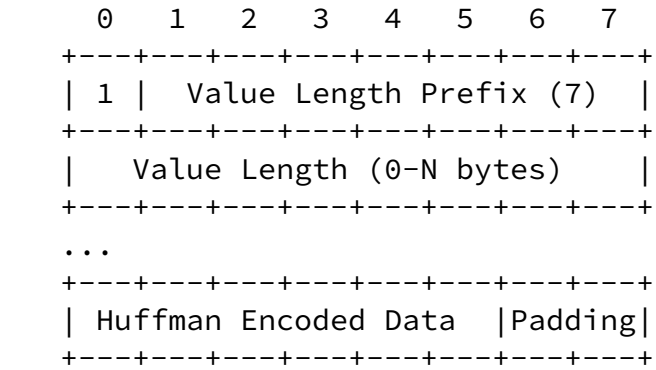
Padding is necessary when doing huffman encoding to ensure that the remaining bits between the actual end of the data and the next byte boundary are not misinterpreted as part of the input data.

When padding for huffman encoding, use the bits from the EOS (end-of-string) entry in the Huffman table, starting with the MSB. This entry is guaranteed to be at least 8 bits long.

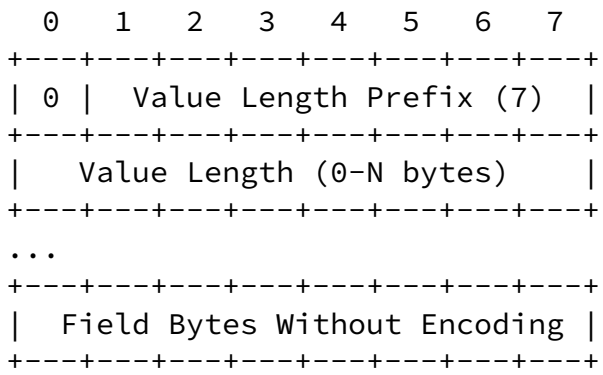
String literals sent in the client to server direction which use huffman encoding are encoded with the codes within the request huffman code table (Appendix C) (see Request Decoding With Huffman Example (Appendix E.3)).

String literals sent in the server to client direction which use huffman encoding are encoded with the codes within the response huffman code table (Appendix D) (see Response Decoding With Huffman Example (Appendix E.3)).

The EOS symbol is represented with value 256, and is used solely to signal the end of the huffman-encoded key data or the end of the huffman-encoded value data. Given that only between 0-7 bits of the EOS symbol is included in any huffman-encoded string, and given that the EOS symbol is at least 8 bits long, it is expected that it should never be successfully decoded.



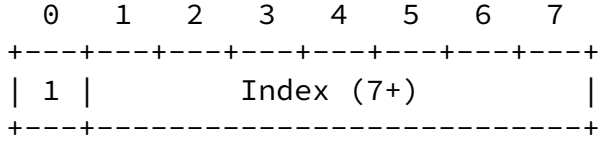
String Literal With Huffman Encoding



String Literal Without Huffman Encoding

4.2. Indexed Header Field Representation

An indexed header field representation either identifies an entry in the header table or static table. The specified entry is emitted and a reference to that entry is added to the reference set if it is not currently in the reference set. If it is present in the reference set then the reference is removed and the entry is not emitted.



Indexed Header Field

This representation starts with the '1' 1-bit pattern, followed by the index of the matching pair, represented as an integer with a 7-bit prefix.

4.3. Literal Header Field Representation

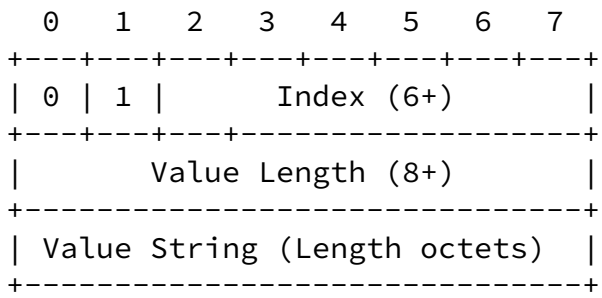
Literal header field representations contain a literal header field value. Header field names are either provided as a literal or by

reference to an existing header table or static table entry.

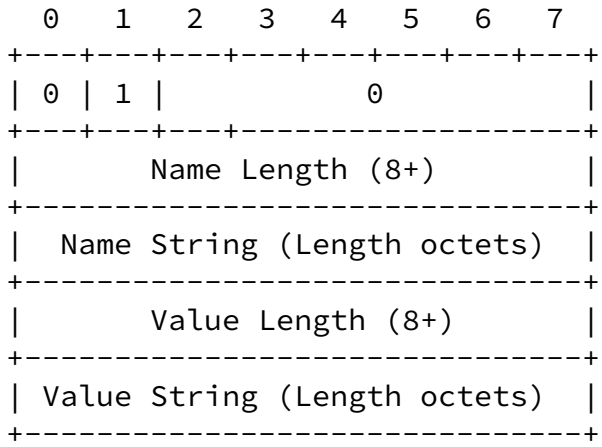
Literal representations all result in the emission of a header field when decoded.

4.3.1. Literal Header Field without Indexing

A literal header field without indexing causes the emission of a header field without altering the header table.



Literal Header Field without Indexing - Indexed Name



Literal Header Field without Indexing - New Name

This representation starts with the '01' 2-bit pattern.

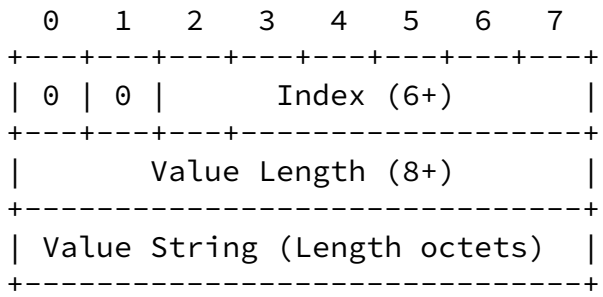
If the header field name matches the header field name of a (name, value) pair stored in the Header Table or Static Table, the index of that entry, increased by one (index + 1), is represented as an integer with a 6-bit prefix. Note that if the index is strictly below 63, only one byte is used for this representation.

If the header field name does not match a header field name entry, the value 0 is represented on 6 bits followed by the header field name (Section 4.1.2).

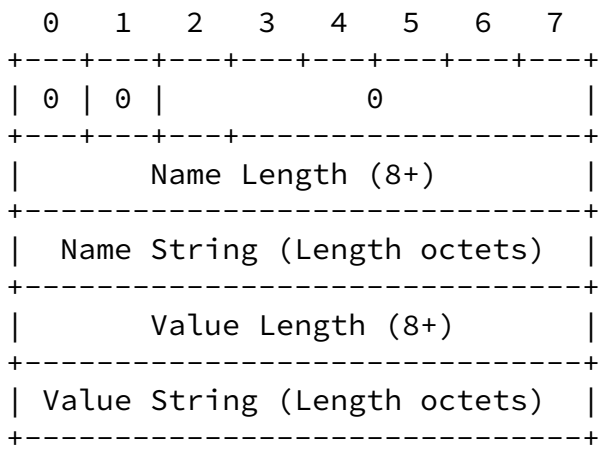
The header field name representation is followed by the header field value represented as a literal string as described in Section 4.1.2.

4.3.2. Literal Header Field with Incremental Indexing

A literal header field with incremental indexing adds a new entry to the header table.



Literal Header Field with Incremental Indexing - Indexed Name



Literal Header Field with Incremental Indexing - New Name

This representation starts with the '00' 2-bit pattern.

If the header field name matches the header field name of a (name, value) pair stored in the header table or static table, the index of the pair increased by one (index + 1) is represented as an integer with a 6-bit prefix.

If the header field name does not match a header field name entry, the value 0 is represented on 6 bits followed by the header field name (Section 4.1.2).

The header field name representation is followed by the header field value represented as a literal string as described in Section 4.1.2.

5. Security Considerations

This compressor exists to solve security issues present in stream compressors such as DEFLATE whereby the compression context can be efficiently probed to reveal secrets. A conformant implementation of this specification should be fairly safe against that kind of attack, as the reaping of any information from the compression context requires more work than guessing and verifying the plaintext data directly with the server. As with any secret, however, the longer the length of the secret, the more difficult the secret is to guess. It is inadvisable to have short cookies that are relied upon to remain secret for any duration of time.

A proper security-conscious implementation will also need to prevent timing attacks by ensuring that the amount of time it takes to do string comparisons is always a function of the total length of the strings, and not a function of the number of matched characters.

A decoder needs to ensure that larger values or encodings of integers do not permit exploitation. Decoders MUST limit the size of integers, both in value and encoded length, that it accepts (see Section 4.1.1).

Another common security problem is when the remote endpoint successfully causes the local endpoint to exhaust its memory. This compressor attempts to deal with the most obvious ways that this could occur by limiting both the peak and the steady-state amount of memory consumed in the compressor state, by providing ways for the application to consume/flush the emitted header fields in small chunks, and by considering overhead in the state size calculation. Implementors must still be careful in the creation of APIs to an implementation of this compressor by ensuring that header field keys and values are either emitted as a stream, or that the compression

implementation have a limit on the maximum size of a key or value. Failure to implement these kinds of safeguards may still result in a scenario where the local endpoint exhausts its memory.

6. References

6.1. Normative References

[HTTP-p1] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", draft-ietf-httpbis-p1-messaging-24 (work in progress), September 2013.

[HTTP2] Belshe, M., Peon, R., Thomson, M., and A. Melnikov, "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-06 (work in progress), August 2013.

6.2. Informative References

[CANON] Schwartz, E. and B. Kallick, "Generating a canonical prefix encoding", Communications of the ACM Volume 7 Issue 3, pp. 166-169, March 1964, <http://dl.acm.org/citation.cfm?id=363991>.

[CRIME] Rizzo, J. and T. Duong, "The Crime Attack", September 2012, <https://docs.google.com/a/twist.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/edit#slide=id.g1eb6c1b5_3_6>.

[HUFF] Huffman, D., "A Method for the Construction of Minimim Redundancy Codes", Proceedings of the Institute of Radio Engineers Volume 40, Number 9, pp. 1098-1101, September 1952, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4051119>.

[PERF1] Belshe, M., "IETF83: SPDY and What to Consider for HTTP/2.0", March 2012, <http://www.ietf.org/proceedings/

83/slides/slides-83-httpbis-3>.

[PERF2] McManus, P., "SPDY: What I Like About You", September 2011, <http://bitsup.blogspot.com/2011/09/spdy-what-i-like-about-you.html>.

[SPDY] Belshe, M. and R. Peon, "SPDY Protocol", draft-mbelshe-httpbis-spdy-00 (work in progress), February 2012.

URIs

[1] <http://en.wikipedia.org/wiki/Variable-length_quantity>

Appendix A. Change Log (to be removed by RFC Editor before publication

A.1. Since draft-ietf-httpbis-header-compression-03

- A large number of editorial changes; changed the description of evicting/adding new entries.
- Removed substitution indexing
- Changed 'initial headers' to 'static headers', as per issue #258
- Merged 'request' and 'response' static headers, as per issue #259
- Changed text to indicate that new headers are added at index 0 and expire from the largest index, as per issue #233

A.2. Since draft-ietf-httpbis-header-compression-02

- Corrected error in integer encoding pseudocode.

A.3. Since draft-ietf-httpbis-header-compression-01

- Refactored of Header Encoding Section: split definitions and processing rule.
- Backward incompatible change: Updated reference set management as

per issue #214. This changes how the interaction between the reference set and eviction works. This also changes the working of the reference set in some specific cases.

- Backward incompatible change: modified initial header list, as per issue #188.

- Added example of 32 bytes entry structure (issue #191).

- Added Header Set Completion section. Reflowed some text. Clarified some writing which was akward. Added text about duplicate header entry encoding. Clarified some language w.r.t Header Set. Changed x-my-header to mynewheader. Added text in the HeaderEmission section indicating that the application may also be able to free up memory more quickly. Added information in Security Considerations section.

A.4. Since draft-ietf-httpbis-header-compression-01

Fixed bug/omission in integer representation algorithm.

Changed the document title.

Header matching text rewritten.

Changed the definition of header emission.

Changed the name of the setting which dictates how much memory the compression context should use.

Removed "specific use cases" section

Corrected erroneous statement about what index can be contained in one byte

Added descriptions of opcodes

Removed security claims from introduction.

Appendix B. Static Table

The static table consists of an unchangable ordered list of (name, value) pairs. The first entry in the table is always represented by the index len(header table), and the last entry in the table is represented by the index len(header table)+len(static table)-1.

[[anchor9: The ordering of these tables is currently arbitrary. The tables in this section should be updated and ordered such that the table entries with the smallest indices are those which, based on a statistical analysis of the frequency of use weighted by size, achieve the largest decrease in bytes transmitted subject to HTTP 2.0 header field rules (like removal of some header fields). This set of header fields is currently very likely incomplete, and should be made complete.]]

The following table lists the pre-defined header fields that make-up the static header table.

Index	Header Name	Header Value
0	:authority	
1	:method	GET
2	:method	POST
3	:path	/
4	:path	/index.html
5	:scheme	http
6	:scheme	https
7	:status	200

8	:status	500
9	:status	404
10	:status	403
11	:status	400
12	:status	401
13	accept-charset	
14	accept-encoding	
15	accept-language	
16	accept-ranges	
17	accept	
18	access-control-allow-origin	
19	age	

20	allow	
21	authorization	
22	cache-control	
23	content-disposition	
24	content-encoding	
25	content-language	
26	content-length	
27	content-location	
28	content-range	
29	content-type	
30	cookie	
31	date	
32	etag	
33	expect	
34	expires	
35	from	
36	if-match	
37	if-modified-since	
38	if-none-match	
39	if-range	
40	if-unmodified-since	
41	last-modified	
42	link	
43	location	
44	max-forwards	
45	proxy-authenticate	
46	proxy-authorization	
47	range	
48	referer	
49	refresh	
50	retry-after	
51	server	
52	set-cookie	
53	strict-transport-security	
54	transfer-encoding	
55	user-agent	

56	vary	
57	via	
58	www-authenticate	

Table 1: Static Table Entries

The table give the index of each entry in the static table. The full index of each entry, to be used for encoding a reference to this entry, is computed by adding the number of entries in the header table to this index.

Appendix C. Huffman Codes For Requests

The following Huffman codes are used when encoding string literals in the client to server direction.

[[anchor10: This table is out of date and needs updating. In particular, EOS needs to be at least 7-bits long and currently is not.]]

sym	aligned to MSB as bits	len in bits	aligned to LSB as hex	len in bits
(0)	|11111111|11111111|11110111|010	[27]	7ffffba	[27]
(1)	|11111111|11111111|11110111|011	[27]	7ffffbb	[27]
(2)	|11111111|11111111|11110111|100	[27]	7ffffbc	[27]
(3)	|11111111|11111111|11110111|101	[27]	7ffffbd	[27]
(4)	|11111111|11111111|11110111|110	[27]	7ffffbe	[27]
(5)	|11111111|11111111|11110111|111	[27]	7ffffbf	[27]
(6)	|11111111|11111111|11111000|000	[27]	7ffffc0	[27]
(7)	|11111111|11111111|11111000|001	[27]	7ffffc1	[27]
(8)	|11111111|11111111|11111000|010	[27]	7ffffc2	[27]
(9)	|11111111|11111111|11111000|011	[27]	7ffffc3	[27]
(10)	|11111111|11111111|11111000|100	[27]	7ffffc4	[27]
(11)	|11111111|11111111|11111000|101	[27]	7ffffc5	[27]
(12)	|11111111|11111111|11111000|110	[27]	7ffffc6	[27]
(13)	|11111111|11111111|11111000|111	[27]	7ffffc7	[27]
(14)	|11111111|11111111|11111001|000	[27]	7ffffc8	[27]
(15)	|11111111|11111111|11111001|001	[27]	7ffffc9	[27]
(16)	|11111111|11111111|11111001|010	[27]	7ffffca	[27]
(17)	|11111111|11111111|11111001|011	[27]	7ffffcb	[27]
(18)	|11111111|11111111|11111001|100	[27]	7ffffcc	[27]
(19)	|11111111|11111111|11111001|101	[27]	7ffffcd	[27]
(20)	|11111111|11111111|11111001|110	[27]	7ffffce	[27]
(21)	|11111111|11111111|11111001|111	[27]	7ffffcf	[27]
(22)	|11111111|11111111|11111010|000	[27]	7ffffd0	[27]

```
    ( 23) |11111111|11111111|11111010|001 [27]      7ffffd1 [27]
    ( 24) |11111111|11111111|11111010|010 [27]      7ffffd2 [27]
    ( 25) |11111111|11111111|11111010|011 [27]      7ffffd3 [27]
    ( 26) |11111111|11111111|11111010|100 [27]      7ffffd4 [27]
    ( 27) |11111111|11111111|11111010|101 [27]      7ffffd5 [27]
    ( 28) |11111111|11111111|11111010|110 [27]      7ffffd6 [27]
    ( 29) |11111111|11111111|11111010|111 [27]      7ffffd7 [27]
    ( 30) |11111111|11111111|11111011|000 [27]      7ffffd8 [27]
    ( 31) |11111111|11111111|11111011|001 [27]      7ffffd9 [27]
' ' ( 32) |11101000| [8]                                 e8 [8]
'!' ( 33) |11111111|1100 [12]                           ffc [12]
'"' ( 34) |11111111|111010 [14]                        3ffa [14]
'#' ( 35) |11111111|1111100 [15]                       7ffc [15]
'$' ( 36) |11111111|1111101 [15]                       7ffd [15]
'%' ( 37) |100100 [6]                                    24 [6]
'&' ( 38) |1101110 [7]                                   6e [7]
''' ( 39) |11111111|1111110 [15]                       7ffe [15]
'(' ( 40) |11111111|010 [11]                            7fa [11]
')' ( 41) |11111111|011 [11]                            7fb [11]
'*' ( 42) |11111110|10 [10]                             3fa [10]
'+' ( 43) |11111111|100 [11]                            7fc [11]
',' ( 44) |11101001| [8]                                 e9 [8]
'-' ( 45) |100101 [6]                                    25 [6]
'.' ( 46) |00100 [5]                                      4 [5]
'/' ( 47) |0000 [4]                                       0 [4]
'0' ( 48) |00101 [5]                                      5 [5]
'1' ( 49) |00110 [5]                                      6 [5]
'2' ( 50) |00111 [5]                                      7 [5]
'3' ( 51) |100110 [6]                                    26 [6]
'4' ( 52) |100111 [6]                                    27 [6]
'5' ( 53) |101000 [6]                                    28 [6]
'6' ( 54) |101001 [6]                                    29 [6]
'7' ( 55) |101010 [6]                                    2a [6]
'8' ( 56) |101011 [6]                                    2b [6]
'9' ( 57) |101100 [6]                                    2c [6]
':' ( 58) |11110110|0 [9]                               1ec [9]
';' ( 59) |11101010| [8]                                 ea [8]
'<' ( 60) |11111111|11111111|10 [18]                  3fffe [18]
'=' ( 61) |101101 [6]                                    2d [6]
'>' ( 62) |11111111|11111110|0 [17]                   1fffc [17]
'?' ( 63) |11110110|1 [9]                                1ed [9]
'@' ( 64) |11111111|111011 [14]                         3ffb [14]
'A' ( 65) |1101111 [7]                                   6f [7]
'B' ( 66) |11101011| [8]                                 eb [8]
'C' ( 67) |11101100| [8]                                 ec [8]
'D' ( 68) |11101101| [8]                                 ed [8]
'E' ( 69) |11101110| [8]                                 ee [8]
'F' ( 70) |1110000 [7]                                   70 [7]
```

```
'G' ( 71) |11110111|0 [9]                                1ee [9]
'H' ( 72) |11110111|1 [9]                                1ef [9]
'I' ( 73) |11111000|0 [9]                                1f0 [9]
'J' ( 74) |11111000|1 [9]                                1f1 [9]
'K' ( 75) |11111110|11 [10]                              3fb [10]
'L' ( 76) |11111001|0 [9]                                1f2 [9]
'M' ( 77) |11101111| [8]                                  ef [8]
'N' ( 78) |11111001|1 [9]                                1f3 [9]
'O' ( 79) |11111010|0 [9]                                1f4 [9]
'P' ( 80) |11111010|1 [9]                                1f5 [9]
'Q' ( 81) |11111011|0 [9]                                1f6 [9]
'R' ( 82) |11111011|1 [9]                                1f7 [9]
'S' ( 83) |11110000| [8]                                  f0 [8]
'T' ( 84) |11110001| [8]                                  f1 [8]
'U' ( 85) |11111100|0 [9]                                1f8 [9]
'V' ( 86) |11111100|1 [9]                                1f9 [9]
'W' ( 87) |11111101|0 [9]                                1fa [9]
'X' ( 88) |11111101|1 [9]                                1fb [9]
'Y' ( 89) |11111110|0 [9]                                1fc [9]
'Z' ( 90) |11111111|00 [10]                              3fc [10]
'[' ( 91) |11111111|111100 [14]                          3ffc [14]
'\' ( 92) |11111111|11111111|11111011|010 [27]      7ffffda [27]
']' ( 93) |11111111|11100 [13]                           1ffc [13]
'^' ( 94) |11111111|111101 [14]                          3ffd [14]
'_' ( 95) |101110 [6]                                      2e [6]
'`' ( 96) |11111111|11111111|110 [19]                  7fffe [19]
'a' ( 97) |01000 [5]                                        8 [5]
'b' ( 98) |101111 [6]                                      2f [6]
'c' ( 99) |01001 [5]                                        9 [5]
'd' (100) |110000 [6]                                      30 [6]
'e' (101) |0001 [4]                                         1 [4]
'f' (102) |110001 [6]                                      31 [6]
'g' (103) |110010 [6]                                      32 [6]
'h' (104) |110011 [6]                                      33 [6]
'i' (105) |01010 [5]                                        a [5]
'j' (106) |1110001 [7]                                     71 [7]
'k' (107) |1110010 [7]                                     72 [7]
'l' (108) |01011 [5]                                        b [5]
'm' (109) |110100 [6]                                      34 [6]
'n' (110) |01100 [5]                                        c [5]
'o' (111) |01101 [5]                                        d [5]
'p' (112) |01110 [5]                                        e [5]
```

```
     'q' (113) |11110010| [8]                                        f2 [8]
     'r' (114) |01111 [5]                                          f [5]
     's' (115) |10000 [5]                                         10 [5]
     't' (116) |10001 [5]                                         11 [5]
     'u' (117) |110101 [6]                                        35 [6]
     'v' (118) |1110011 [7]                                       73 [7]
```



```
     'w' (119) |110110 [6]                                        36 [6]
     'x' (120) |11110011| [8]                                     f3 [8]
     'y' (121) |11110100| [8]                                     f4 [8]
     'z' (122) |11110101| [8]                                     f5 [8]
     '{' (123) |11111111|11111110|1 [17]                       1fffd [17]
     '|' (124) |11111111|101 [11]                                7fd [11]
     '}' (125) |11111111|11111111|0 [17]                       1fffe [17]
     '~' (126) |11111111|1101 [12]                                ffd [12]
         (127) |11111111|11111111|11111011|011 [27]          7ffffdb [27]
         (128) |11111111|11111111|11111011|100 [27]          7ffffdc [27]
         (129) |11111111|11111111|11111011|101 [27]          7ffffdd [27]
         (130) |11111111|11111111|11111011|110 [27]          7ffffde [27]
         (131) |11111111|11111111|11111011|111 [27]          7ffffdf [27]
         (132) |11111111|11111111|11111100|000 [27]          7ffffe0 [27]
         (133) |11111111|11111111|11111100|001 [27]          7ffffe1 [27]
         (134) |11111111|11111111|11111100|010 [27]          7ffffe2 [27]
         (135) |11111111|11111111|11111100|011 [27]          7ffffe3 [27]
         (136) |11111111|11111111|11111100|100 [27]          7ffffe4 [27]
         (137) |11111111|11111111|11111100|101 [27]          7ffffe5 [27]
         (138) |11111111|11111111|11111100|110 [27]          7ffffe6 [27]
         (139) |11111111|11111111|11111100|111 [27]          7ffffe7 [27]
         (140) |11111111|11111111|11111101|000 [27]          7ffffe8 [27]
         (141) |11111111|11111111|11111101|001 [27]          7ffffe9 [27]
         (142) |11111111|11111111|11111101|010 [27]          7ffffea [27]
         (143) |11111111|11111111|11111101|011 [27]          7ffffeb [27]
         (144) |11111111|11111111|11111101|100 [27]          7ffffec [27]
         (145) |11111111|11111111|11111101|101 [27]          7ffffed [27]
         (146) |11111111|11111111|11111101|110 [27]          7ffffee [27]
         (147) |11111111|11111111|11111101|111 [27]          7fffef [27]
         (148) |11111111|11111111|11111110|000 [27]          7fffff0 [27]
         (149) |11111111|11111111|11111110|001 [27]          7fffff1 [27]
         (150) |11111111|11111111|11111110|010 [27]          7fffff2 [27]
         (151) |11111111|11111111|11111110|011 [27]          7fffff3 [27]
         (152) |11111111|11111111|11111110|100 [27]          7fffff4 [27]
         (153) |11111111|11111111|11111110|101 [27]          7fffff5 [27]
```

```
(154) |11111111|11111111|11111110|110 [27]      7fffff6 [27]
(155) |11111111|11111111|11111110|111 [27]      7fffff7 [27]
(156) |11111111|11111111|11111111|000 [27]      7fffff8 [27]
(157) |11111111|11111111|11111111|001 [27]      7fffff9 [27]
(158) |11111111|11111111|11111111|010 [27]      7fffffa [27]
(159) |11111111|11111111|11111111|011 [27]      7fffffb [27]
(160) |11111111|11111111|11111111|100 [27]      7fffffc [27]
(161) |11111111|11111111|11111111|101 [27]      7fffffd [27]
(162) |11111111|11111111|11111111|110 [27]      7fffffe [27]
(163) |11111111|11111111|11111111|111 [27]      7ffffff [27]
(164) |11111111|11111111|11100000|00 [26]       3ffff80 [26]
(165) |11111111|11111111|11100000|01 [26]       3ffff81 [26]
(166) |11111111|11111111|11100000|10 [26]       3ffff82 [26]
```



```
(167) |11111111|11111111|11100000|11 [26]       3ffff83 [26]
(168) |11111111|11111111|11100001|00 [26]       3ffff84 [26]
(169) |11111111|11111111|11100001|01 [26]       3ffff85 [26]
(170) |11111111|11111111|11100001|10 [26]       3ffff86 [26]
(171) |11111111|11111111|11100001|11 [26]       3ffff87 [26]
(172) |11111111|11111111|11100010|00 [26]       3ffff88 [26]
(173) |11111111|11111111|11100010|01 [26]       3ffff89 [26]
(174) |11111111|11111111|11100010|10 [26]       3ffff8a [26]
(175) |11111111|11111111|11100010|11 [26]       3ffff8b [26]
(176) |11111111|11111111|11100011|00 [26]       3ffff8c [26]
(177) |11111111|11111111|11100011|01 [26]       3ffff8d [26]
(178) |11111111|11111111|11100011|10 [26]       3ffff8e [26]
(179) |11111111|11111111|11100011|11 [26]       3ffff8f [26]
(180) |11111111|11111111|11100100|00 [26]       3ffff90 [26]
(181) |11111111|11111111|11100100|01 [26]       3ffff91 [26]
(182) |11111111|11111111|11100100|10 [26]       3ffff92 [26]
(183) |11111111|11111111|11100100|11 [26]       3ffff93 [26]
(184) |11111111|11111111|11100101|00 [26]       3ffff94 [26]
(185) |11111111|11111111|11100101|01 [26]       3ffff95 [26]
(186) |11111111|11111111|11100101|10 [26]       3ffff96 [26]
(187) |11111111|11111111|11100101|11 [26]       3ffff97 [26]
(188) |11111111|11111111|11100110|00 [26]       3ffff98 [26]
(189) |11111111|11111111|11100110|01 [26]       3ffff99 [26]
(190) |11111111|11111111|11100110|10 [26]       3ffff9a [26]
(191) |11111111|11111111|11100110|11 [26]       3ffff9b [26]
(192) |11111111|11111111|11100111|00 [26]       3ffff9c [26]
(193) |11111111|11111111|11100111|01 [26]       3ffff9d [26]
(194) |11111111|11111111|11100111|10 [26]       3ffff9e [26]
```

```
(195) |11111111|11111111|11100111|11 [26]       3ffff9f [26]
(196) |11111111|11111111|11101000|00 [26]       3ffffa0 [26]
(197) |11111111|11111111|11101000|01 [26]       3ffffa1 [26]
(198) |11111111|11111111|11101000|10 [26]       3ffffa2 [26]
(199) |11111111|11111111|11101000|11 [26]       3ffffa3 [26]
(200) |11111111|11111111|11101001|00 [26]       3ffffa4 [26]
(201) |11111111|11111111|11101001|01 [26]       3ffffa5 [26]
(202) |11111111|11111111|11101001|10 [26]       3ffffa6 [26]
(203) |11111111|11111111|11101001|11 [26]       3ffffa7 [26]
(204) |11111111|11111111|11101010|00 [26]       3ffffa8 [26]
(205) |11111111|11111111|11101010|01 [26]       3ffffa9 [26]
(206) |11111111|11111111|11101010|10 [26]       3ffffaa [26]
(207) |11111111|11111111|11101010|11 [26]       3ffffab [26]
(208) |11111111|11111111|11101011|00 [26]       3ffffac [26]
(209) |11111111|11111111|11101011|01 [26]       3ffffad [26]
(210) |11111111|11111111|11101011|10 [26]       3ffffae [26]
(211) |11111111|11111111|11101011|11 [26]       3ffffaf [26]
(212) |11111111|11111111|11101100|00 [26]       3ffffb0 [26]
(213) |11111111|11111111|11101100|01 [26]       3ffffb1 [26]
(214) |11111111|11111111|11101100|10 [26]       3ffffb2 [26]
```



```
(215) |11111111|11111111|11101100|11 [26]       3ffffb3 [26]
(216) |11111111|11111111|11101101|00 [26]       3ffffb4 [26]
(217) |11111111|11111111|11101101|01 [26]       3ffffb5 [26]
(218) |11111111|11111111|11101101|10 [26]       3ffffb6 [26]
(219) |11111111|11111111|11101101|11 [26]       3ffffb7 [26]
(220) |11111111|11111111|11101110|00 [26]       3ffffb8 [26]
(221) |11111111|11111111|11101110|01 [26]       3ffffb9 [26]
(222) |11111111|11111111|11101110|10 [26]       3ffffba [26]
(223) |11111111|11111111|11101110|11 [26]       3ffffbb [26]
(224) |11111111|11111111|11101111|00 [26]       3ffffbc [26]
(225) |11111111|11111111|11101111|01 [26]       3ffffbd [26]
(226) |11111111|11111111|11101111|10 [26]       3ffffbe [26]
(227) |11111111|11111111|11101111|11 [26]       3ffffbf [26]
(228) |11111111|11111111|11110000|00 [26]       3ffffc0 [26]
(229) |11111111|11111111|11110000|01 [26]       3ffffc1 [26]
(230) |11111111|11111111|11110000|10 [26]       3ffffc2 [26]
(231) |11111111|11111111|11110000|11 [26]       3ffffc3 [26]
(232) |11111111|11111111|11110001|00 [26]       3ffffc4 [26]
(233) |11111111|11111111|11110001|01 [26]       3ffffc5 [26]
(234) |11111111|11111111|11110001|10 [26]       3ffffc6 [26]
(235) |11111111|11111111|11110001|11 [26]       3ffffc7 [26]
```

```
      (236) |11111111|11111111|11110010|00 [26]       3ffffc8 [26]
      (237) |11111111|11111111|11110010|01 [26]       3ffffc9 [26]
      (238) |11111111|11111111|11110010|10 [26]       3ffffca [26]
      (239) |11111111|11111111|11110010|11 [26]       3ffffcb [26]
      (240) |11111111|11111111|11110011|00 [26]       3ffffcc [26]
      (241) |11111111|11111111|11110011|01 [26]       3ffffcd [26]
      (242) |11111111|11111111|11110011|10 [26]       3ffffce [26]
      (243) |11111111|11111111|11110011|11 [26]       3ffffcf [26]
      (244) |11111111|11111111|11110100|00 [26]       3ffffd0 [26]
      (245) |11111111|11111111|11110100|01 [26]       3ffffd1 [26]
      (246) |11111111|11111111|11110100|10 [26]       3ffffd2 [26]
      (247) |11111111|11111111|11110100|11 [26]       3ffffd3 [26]
      (248) |11111111|11111111|11110101|00 [26]       3ffffd4 [26]
      (249) |11111111|11111111|11110101|01 [26]       3ffffd5 [26]
      (250) |11111111|11111111|11110101|10 [26]       3ffffd6 [26]
      (251) |11111111|11111111|11110101|11 [26]       3ffffd7 [26]
      (252) |11111111|11111111|11110110|00 [26]       3ffffd8 [26]
      (253) |11111111|11111111|11110110|01 [26]       3ffffd9 [26]
      (254) |11111111|11111111|11110110|10 [26]       3ffffda [26]
      (255) |11111111|11111111|11110110|11 [26]       3ffffdb [26]
EOS   (256) |11111111|11111111|11110111|00 [26]       3ffffdc [26]
```


Appendix D. Huffman Codes for Responses

The following Huffman codes are used when encoding string literals in the server to client direction.

[[anchor11: This table is out of date and needs updating. In particular, EOS needs to be at least 7-bits long and currently is not.]]

sym	aligned to MSB as bits	len in bits	aligned to LSB as hex	len in bits

```
      (  0) |11111111|11111111|11011110|0 [25]        1ffffbc [25]
      (  1) |11111111|11111111|11011110|1 [25]        1ffffbd [25]
      (  2) |11111111|11111111|11011111|0 [25]        1ffffbe [25]
      (  3) |11111111|11111111|11011111|1 [25]        1ffffbf [25]
      (  4) |11111111|11111111|11100000|0 [25]        1ffffc0 [25]
      (  5) |11111111|11111111|11100000|1 [25]        1ffffc1 [25]
      (  6) |11111111|11111111|11100001|0 [25]        1ffffc2 [25]
      (  7) |11111111|11111111|11100001|1 [25]        1ffffc3 [25]
      (  8) |11111111|11111111|11100010|0 [25]        1ffffc4 [25]
      (  9) |11111111|11111111|11100010|1 [25]        1ffffc5 [25]
      ( 10) |11111111|11111111|11100011|0 [25]        1ffffc6 [25]
      ( 11) |11111111|11111111|11100011|1 [25]        1ffffc7 [25]
      ( 12) |11111111|11111111|11100100|0 [25]        1ffffc8 [25]
      ( 13) |11111111|11111111|11100100|1 [25]        1ffffc9 [25]
      ( 14) |11111111|11111111|11100101|0 [25]        1ffffca [25]
      ( 15) |11111111|11111111|11100101|1 [25]        1ffffcb [25]
      ( 16) |11111111|11111111|11100110|0 [25]        1ffffcc [25]
      ( 17) |11111111|11111111|11100110|1 [25]        1ffffcd [25]
      ( 18) |11111111|11111111|11100111|0 [25]        1ffffce [25]
      ( 19) |11111111|11111111|11100111|1 [25]        1ffffcf [25]
      ( 20) |11111111|11111111|11101000|0 [25]        1ffffd0 [25]
      ( 21) |11111111|11111111|11101000|1 [25]        1ffffd1 [25]
      ( 22) |11111111|11111111|11101001|0 [25]        1ffffd2 [25]
      ( 23) |11111111|11111111|11101001|1 [25]        1ffffd3 [25]
      ( 24) |11111111|11111111|11101010|0 [25]        1ffffd4 [25]
      ( 25) |11111111|11111111|11101010|1 [25]        1ffffd5 [25]
      ( 26) |11111111|11111111|11101011|0 [25]        1ffffd6 [25]
      ( 27) |11111111|11111111|11101011|1 [25]        1ffffd7 [25]
      ( 28) |11111111|11111111|11101100|0 [25]        1ffffd8 [25]
      ( 29) |11111111|11111111|11101100|1 [25]        1ffffd9 [25]
      ( 30) |11111111|11111111|11101101|0 [25]        1ffffda [25]
      ( 31) |11111111|11111111|11101101|1 [25]        1ffffdb [25]
' ' ( 32) |0000 [4]                                          0 [4]
'!' ( 33) |11111111|1010 [12]                             ffa [12]
```



```
'"' ( 34) |1101010 [7]                                     6a [7]
'#' ( 35) |11111111|11010 [13]                          1ffa [13]
'$' ( 36) |11111111|111100 [14]                         3ffc [14]
'%' ( 37) |11110110|0 [9]                                 1ec [9]
'&' ( 38) |11111110|00 [10]                               3f8 [10]
''' ( 39) |11111111|11011 [13]                          1ffb [13]
'(' ( 40) |11110110|1 [9]                                 1ed [9]
```

```
')' ( 41) |11110111|0 [9]                                1ee [9]
'*' ( 42) |11111111|1011 [12]                           ffb [12]
'+' ( 43) |11111111|010 [11]                            7fa [11]
',' ( 44) |100010 [6]                                     22 [6]
'-' ( 45) |100011 [6]                                     23 [6]
'.' ( 46) |100100 [6]                                     24 [6]
'/' ( 47) |1101011 [7]                                    6b [7]
'0' ( 48) |0001 [4]                                        1 [4]
'1' ( 49) |0010 [4]                                        2 [4]
'2' ( 50) |0011 [4]                                        3 [4]
'3' ( 51) |01000 [5]                                       8 [5]
'4' ( 52) |01001 [5]                                       9 [5]
'5' ( 53) |01010 [5]                                       a [5]
'6' ( 54) |100101 [6]                                     25 [6]
'7' ( 55) |100110 [6]                                     26 [6]
'8' ( 56) |01011 [5]                                       b [5]
'9' ( 57) |01100 [5]                                       c [5]
':' ( 58) |01101 [5]                                       d [5]
';' ( 59) |11110111|1 [9]                                1ef [9]
'<' ( 60) |11111111|11111010| [16]                      fffa [16]
'=' ( 61) |1101100 [7]                                    6c [7]
'>' ( 62) |11111111|11100 [13]                          1ffc [13]
'?' ( 63) |11111111|1100 [12]                           ffc [12]
'@' ( 64) |11111111|11111011| [16]                      fffb [16]
'A' ( 65) |1101101 [7]                                    6d [7]
'B' ( 66) |11101010| [8]                                  ea [8]
'C' ( 67) |11101011| [8]                                  eb [8]
'D' ( 68) |11101100| [8]                                  ec [8]
'E' ( 69) |11101101| [8]                                  ed [8]
'F' ( 70) |11101110| [8]                                  ee [8]
'G' ( 71) |100111 [6]                                     27 [6]
'H' ( 72) |11111000|0 [9]                                1f0 [9]
'I' ( 73) |11101111| [8]                                  ef [8]
'J' ( 74) |11110000| [8]                                  f0 [8]
'K' ( 75) |11111110|01 [10]                             3f9 [10]
'L' ( 76) |11111000|1 [9]                                1f1 [9]
'M' ( 77) |101000 [6]                                     28 [6]
'N' ( 78) |11110001| [8]                                  f1 [8]
'O' ( 79) |11110010| [8]                                  f2 [8]
'P' ( 80) |11111001|0 [9]                                1f2 [9]
'Q' ( 81) |11111110|10 [10]                             3fa [10]
```

```
'R' ( 82) |11111001|1 [9]                                1f3 [9]
'S' ( 83) |101001 [6]                                    29 [6]
'T' ( 84) |01110 [5]                                      e [5]
'U' ( 85) |11111010|0 [9]                                1f4 [9]
'V' ( 86) |11111010|1 [9]                                1f5 [9]
'W' ( 87) |11110011| [8]                                  f3 [8]
'X' ( 88) |11111110|11 [10]                             3fb [10]
'Y' ( 89) |11111011|0 [9]                                1f6 [9]
'Z' ( 90) |11111111|00 [10]                             3fc [10]
'[' ( 91) |11111111|011 [11]                            7fb [11]
'\' ( 92) |11111111|11101 [13]                         1ffd [13]
']' ( 93) |11111111|100 [11]                            7fc [11]
'^' ( 94) |11111111|1111100 [15]                       7ffc [15]
'_' ( 95) |11111011|1 [9]                                1f7 [9]
'`' ( 96) |11111111|11111111|0 [17]                    1fffe [17]
'a' ( 97) |01111 [5]                                      f [5]
'b' ( 98) |1101110 [7]                                   6e [7]
'c' ( 99) |101010 [6]                                    2a [6]
'd' (100) |101011 [6]                                    2b [6]
'e' (101) |10000 [5]                                     10 [5]
'f' (102) |1101111 [7]                                   6f [7]
'g' (103) |1110000 [7]                                   70 [7]
'h' (104) |1110001 [7]                                   71 [7]
'i' (105) |101100 [6]                                    2c [6]
'j' (106) |11111100|0 [9]                                1f8 [9]
'k' (107) |11111100|1 [9]                                1f9 [9]
'l' (108) |1110010 [7]                                   72 [7]
'm' (109) |101101 [6]                                    2d [6]
'n' (110) |101110 [6]                                    2e [6]
'o' (111) |101111 [6]                                    2f [6]
'p' (112) |110000 [6]                                    30 [6]
'q' (113) |11111101|0 [9]                                1fa [9]
'r' (114) |110001 [6]                                    31 [6]
's' (115) |110010 [6]                                    32 [6]
't' (116) |110011 [6]                                    33 [6]
'u' (117) |110100 [6]                                    34 [6]
'v' (118) |1110011 [7]                                   73 [7]
'w' (119) |11110100| [8]                                  f4 [8]
'x' (120) |1110100 [7]                                   74 [7]
'y' (121) |11110101| [8]                                  f5 [8]
'z' (122) |11111101|1 [9]                                1fb [9]
'{' (123) |11111111|11111100| [16]                      fffc [16]
'|' (124) |11111111|111101 [14]                        3ffd [14]
'}' (125) |11111111|11111101| [16]                      fffd [16]
'~' (126) |11111111|11111110| [16]                      fffe [16]
    (127) |11111111|11111111|11101110|0 [25]        1ffffdc [25]
    (128) |11111111|11111111|11101110|1 [25]        1ffffdd [25]
    (129) |11111111|11111111|11101111|0 [25]        1ffffde [25]
```

```
(130) |11111111|11111111|11101111|1 [25]        1fffdf [25]
(131) |11111111|11111111|11110000|0 [25]        1ffffe0 [25]
(132) |11111111|11111111|11110000|1 [25]        1ffffe1 [25]
(133) |11111111|11111111|11110001|0 [25]        1ffffe2 [25]
(134) |11111111|11111111|11110001|1 [25]        1ffffe3 [25]
(135) |11111111|11111111|11110010|0 [25]        1ffffe4 [25]
(136) |11111111|11111111|11110010|1 [25]        1ffffe5 [25]
(137) |11111111|11111111|11110011|0 [25]        1ffffe6 [25]
(138) |11111111|11111111|11110011|1 [25]        1ffffe7 [25]
(139) |11111111|11111111|11110100|0 [25]        1ffffe8 [25]
(140) |11111111|11111111|11110100|1 [25]        1ffffe9 [25]
(141) |11111111|11111111|11110101|0 [25]        1ffffea [25]
(142) |11111111|11111111|11110101|1 [25]        1ffffeb [25]
(143) |11111111|11111111|11110110|0 [25]        1ffffec [25]
(144) |11111111|11111111|11110110|1 [25]        1ffffed [25]
(145) |11111111|11111111|11110111|0 [25]        1ffffee [25]
(146) |11111111|11111111|11110111|1 [25]        1ffffef [25]
(147) |11111111|11111111|11111000|0 [25]        1fffff0 [25]
(148) |11111111|11111111|11111000|1 [25]        1fffff1 [25]
(149) |11111111|11111111|11111001|0 [25]        1fffff2 [25]
(150) |11111111|11111111|11111001|1 [25]        1fffff3 [25]
(151) |11111111|11111111|11111010|0 [25]        1fffff4 [25]
(152) |11111111|11111111|11111010|1 [25]        1fffff5 [25]
(153) |11111111|11111111|11111011|0 [25]        1fffff6 [25]
(154) |11111111|11111111|11111011|1 [25]        1fffff7 [25]
(155) |11111111|11111111|11111100|0 [25]        1fffff8 [25]
(156) |11111111|11111111|11111100|1 [25]        1fffff9 [25]
(157) |11111111|11111111|11111101|0 [25]        1fffffa [25]
(158) |11111111|11111111|11111101|1 [25]        1fffffb [25]
(159) |11111111|11111111|11111110|0 [25]        1fffffc [25]
(160) |11111111|11111111|11111110|1 [25]        1fffffd [25]
(161) |11111111|11111111|11111111|0 [25]        1fffffe [25]
(162) |11111111|11111111|11111111|1 [25]        1ffffff [25]
(163) |11111111|11111111|10000000| [24]          ffff80 [24]
(164) |11111111|11111111|10000001| [24]          ffff81 [24]
(165) |11111111|11111111|10000010| [24]          ffff82 [24]
(166) |11111111|11111111|10000011| [24]          ffff83 [24]
(167) |11111111|11111111|10000100| [24]          ffff84 [24]
(168) |11111111|11111111|10000101| [24]          ffff85 [24]
(169) |11111111|11111111|10000110| [24]          ffff86 [24]
(170) |11111111|11111111|10000111| [24]          ffff87 [24]
(171) |11111111|11111111|10001000| [24]          ffff88 [24]
(172) |11111111|11111111|10001001| [24]          ffff89 [24]
```

```
(173) |11111111|11111111|10001010| [24]          ffff8a [24]
(174) |11111111|11111111|10001011| [24]          ffff8b [24]
(175) |11111111|11111111|10001100| [24]          ffff8c [24]
(176) |11111111|11111111|10001101| [24]          ffff8d [24]
(177) |11111111|11111111|10001110| [24]          ffff8e [24]
```



```
(178) |11111111|11111111|10001111| [24]          ffff8f [24]
(179) |11111111|11111111|10010000| [24]          ffff90 [24]
(180) |11111111|11111111|10010001| [24]          ffff91 [24]
(181) |11111111|11111111|10010010| [24]          ffff92 [24]
(182) |11111111|11111111|10010011| [24]          ffff93 [24]
(183) |11111111|11111111|10010100| [24]          ffff94 [24]
(184) |11111111|11111111|10010101| [24]          ffff95 [24]
(185) |11111111|11111111|10010110| [24]          ffff96 [24]
(186) |11111111|11111111|10010111| [24]          ffff97 [24]
(187) |11111111|11111111|10011000| [24]          ffff98 [24]
(188) |11111111|11111111|10011001| [24]          ffff99 [24]
(189) |11111111|11111111|10011010| [24]          ffff9a [24]
(190) |11111111|11111111|10011011| [24]          ffff9b [24]
(191) |11111111|11111111|10011100| [24]          ffff9c [24]
(192) |11111111|11111111|10011101| [24]          ffff9d [24]
(193) |11111111|11111111|10011110| [24]          ffff9e [24]
(194) |11111111|11111111|10011111| [24]          ffff9f [24]
(195) |11111111|11111111|10100000| [24]          ffffa0 [24]
(196) |11111111|11111111|10100001| [24]          ffffa1 [24]
(197) |11111111|11111111|10100010| [24]          ffffa2 [24]
(198) |11111111|11111111|10100011| [24]          ffffa3 [24]
(199) |11111111|11111111|10100100| [24]          ffffa4 [24]
(200) |11111111|11111111|10100101| [24]          ffffa5 [24]
(201) |11111111|11111111|10100110| [24]          ffffa6 [24]
(202) |11111111|11111111|10100111| [24]          ffffa7 [24]
(203) |11111111|11111111|10101000| [24]          ffffa8 [24]
(204) |11111111|11111111|10101001| [24]          ffffa9 [24]
(205) |11111111|11111111|10101010| [24]          ffffaa [24]
(206) |11111111|11111111|10101011| [24]          ffffab [24]
(207) |11111111|11111111|10101100| [24]          ffffac [24]
(208) |11111111|11111111|10101101| [24]          ffffad [24]
(209) |11111111|11111111|10101110| [24]          ffffae [24]
(210) |11111111|11111111|10101111| [24]          ffffaf [24]
(211) |11111111|11111111|10110000| [24]          ffffb0 [24]
(212) |11111111|11111111|10110001| [24]          ffffb1 [24]
(213) |11111111|11111111|10110010| [24]          ffffb2 [24]
```

```
(214) |11111111|11111111|10110011| [24]          ffffb3 [24]
(215) |11111111|11111111|10110100| [24]          ffffb4 [24]
(216) |11111111|11111111|10110101| [24]          ffffb5 [24]
(217) |11111111|11111111|10110110| [24]          ffffb6 [24]
(218) |11111111|11111111|10110111| [24]          ffffb7 [24]
(219) |11111111|11111111|10111000| [24]          ffffb8 [24]
(220) |11111111|11111111|10111001| [24]          ffffb9 [24]
(221) |11111111|11111111|10111010| [24]          ffffba [24]
(222) |11111111|11111111|10111011| [24]          ffffbb [24]
(223) |11111111|11111111|10111100| [24]          ffffbc [24]
(224) |11111111|11111111|10111101| [24]          ffffbd [24]
(225) |11111111|11111111|10111110| [24]          ffffbe [24]
```



```
(226) |11111111|11111111|10111111| [24]          ffffbf [24]
(227) |11111111|11111111|11000000| [24]          ffffc0 [24]
(228) |11111111|11111111|11000001| [24]          ffffc1 [24]
(229) |11111111|11111111|11000010| [24]          ffffc2 [24]
(230) |11111111|11111111|11000011| [24]          ffffc3 [24]
(231) |11111111|11111111|11000100| [24]          ffffc4 [24]
(232) |11111111|11111111|11000101| [24]          ffffc5 [24]
(233) |11111111|11111111|11000110| [24]          ffffc6 [24]
(234) |11111111|11111111|11000111| [24]          ffffc7 [24]
(235) |11111111|11111111|11001000| [24]          ffffc8 [24]
(236) |11111111|11111111|11001001| [24]          ffffc9 [24]
(237) |11111111|11111111|11001010| [24]          ffffca [24]
(238) |11111111|11111111|11001011| [24]          ffffcb [24]
(239) |11111111|11111111|11001100| [24]          ffffcc [24]
(240) |11111111|11111111|11001101| [24]          ffffcd [24]
(241) |11111111|11111111|11001110| [24]          ffffce [24]
(242) |11111111|11111111|11001111| [24]          ffffcf [24]
(243) |11111111|11111111|11010000| [24]          ffffd0 [24]
(244) |11111111|11111111|11010001| [24]          ffffd1 [24]
(245) |11111111|11111111|11010010| [24]          ffffd2 [24]
(246) |11111111|11111111|11010011| [24]          ffffd3 [24]
(247) |11111111|11111111|11010100| [24]          ffffd4 [24]
(248) |11111111|11111111|11010101| [24]          ffffd5 [24]
(249) |11111111|11111111|11010110| [24]          ffffd6 [24]
(250) |11111111|11111111|11010111| [24]          ffffd7 [24]
(251) |11111111|11111111|11011000| [24]          ffffd8 [24]
(252) |11111111|11111111|11011001| [24]          ffffd9 [24]
(253) |11111111|11111111|11011010| [24]          ffffda [24]
(254) |11111111|11111111|11011011| [24]          ffffdb [24]
```

```
      (255) |11111111|11111111|11011100| [24]          ffffdc [24]
  EOS (256) |11111111|11111111|11011101| [24]          ffffdd [24]
```

Appendix E. Examples

A number of examples are worked through here, for both requests and responses, and with and without huffman coding.

E.1. Request Decoding Example With Huffman

```
# Header set to be encoded
:method: GET
:scheme: http
:path: /
:authority: www.foo.com
```



```
# Hexdump of encoded data which will be decoded
02 84 f7 77 78 ff 07 83 ce 31 77 06 81 0f 04 88 | ...wx....1w.....
db 6d 89 8b 5a 44 b7 4f                         | .m..ZD.O

# Decoded opcodes
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '02'
    name_index:
        encoded: "02"
        decoded: 2
    value_data_length:
        encoded: "84"
        decoded: 4
    value_data:
        is_huffman_encoded: 1
        encoded: "f77778ff"
        decoded: "GET"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '07'
```

```
        name_index:
            encoded: "07"
            decoded: 7
        value_data_length:
            encoded: "83"
            decoded: 3
        value_data:
            is_huffman_encoded: 1
            encoded: "ce3177"
            decoded: "http"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '06'
        name_index:
            encoded: "06"
            decoded: 6
        value_data_length:
            encoded: "81"
            decoded: 1
        value_data:
            is_huffman_encoded: 1
            encoded: "0f"
            decoded: "/"

    LITERAL_INCREMENTAL_OPCODE:
```



```
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '04'
        name_index:
            encoded: "04"
            decoded: 4
        value_data_length:
            encoded: "88"
            decoded: 8
        value_data:
            is_huffman_encoded: 1
            encoded: "db6d898b5a44b74f"
            decoded: "www.foo.com"


    # Decoded header set
```

```
   :authority: www.foo.com
   :method: GET
   :path: /
   :scheme: http

   ##########################

   # Header set to be encoded
   :method: GET
   :scheme: https
   :path: /
   :authority: www.bar.com
   cache-control: no-cache

   # Hexdump of encoded data which will be decoded
   03 84 ce 31 74 3f 02 88 db 6d 89 7a 1e 44 b7 4f | ...1t?...m.z.D.O
   1d 86 63 65 4a 13 98 ff 83 85                   | ..ceJ.....

   # Decoded opcodes
   LITERAL_INCREMENTAL_OPCODE:
           opcodeLengthInBits: 2
           discoveredFromPeekingAtByte: '03'
       name_index:
           encoded: "03"
           decoded: 3
       value_data_length:
           encoded: "84"
           decoded: 4
       value_data:
           is_huffman_encoded: 1
           encoded: "ce31743f"
           decoded: "https"
```



```
   LITERAL_INCREMENTAL_OPCODE:
           opcodeLengthInBits: 2
           discoveredFromPeekingAtByte: '02'
       name_index:
           encoded: "02"
           decoded: 2
       value_data_length:
           encoded: "88"
```

```
            decoded: 8
        value_data:
            is_huffman_encoded: 1
            encoded: "db6d897a1e44b74f"
            decoded: "www.bar.com"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '1d'
        name_index:
            encoded: "1d"
            decoded: 29
        value_data_length:
            encoded: "86"
            decoded: 6
        value_data:
            is_huffman_encoded: 1
            encoded: "63654a1398ff"
            decoded: "no-cache"

    INDEX_OPCODE:
            opcodeLengthInBits: 1
            discoveredFromPeekingAtByte: '83'
        entry_index:
            encoded: "83"
            decoded: 3

    INDEX_OPCODE:
            opcodeLengthInBits: 1
            discoveredFromPeekingAtByte: '85'
        entry_index:
            encoded: "85"
            decoded: 5


    # Decoded header set
    :authority: www.bar.com
    :method: GET
    :path: /
    :scheme: https
```



```
    cache-control: no-cache
```

```
   ########################

   # Header set to be encoded
   :method: GET
   :scheme: https
   :path: /custom-path.css
   :authority: www.bar.com
   custom-key: custom-value

   # Hexdump of encoded data which will be decoded
   05 8b 04 eb 08 b7 49 5c 88 e6 44 c2 1f 00 88 4e | ......I\..D....N
   b0 8b 74 97 90 fa 7f 89 4e b0 8b 74 97 9a 17 a8 | ..t.....N..t....
   ff 82 86                                        | ...

   # Decoded opcodes
   LITERAL_INCREMENTAL_OPCODE:
           opcodeLengthInBits: 2
           discoveredFromPeekingAtByte: '05'
       name_index:
           encoded: "05"
           decoded: 5
       value_data_length:
           encoded: "8b"
           decoded: 11
       value_data:
           is_huffman_encoded: 1
           encoded: "04eb08b7495c88e644c21f"
           decoded: "/custom-path.css"

   LITERAL_INCREMENTAL_OPCODE:
           opcodeLengthInBits: 2
           discoveredFromPeekingAtByte: '00'
       name_index:
           encoded: "00"
           decoded: 0
       name_data_length:
           encoded: "88"
           decoded: 8
       name_data:
           is_huffman_encoded: 1
           encoded: "4eb08b749790fa7f"
           decoded: "custom-key"
       value_data_length:
           encoded: "89"
           decoded: 9
       value_data:
```

```
           is_huffman_encoded: 1
           encoded: "4eb08b74979a17a8ff"
           decoded: "custom-value"

   INDEX_OPCODE:
           opcodeLengthInBits: 1
           discoveredFromPeekingAtByte: '82'
       entry_index:
           encoded: "82"
           decoded: 2

   INDEX_OPCODE:
           opcodeLengthInBits: 1
           discoveredFromPeekingAtByte: '86'
       entry_index:
           encoded: "86"
           decoded: 6


   # Decoded header set
   :authority: www.bar.com
   :method: GET
   :path: /custom-path.css
   :scheme: https
   custom-key: custom-value

   ##########################
```

E.2. Request Decoding Example Without Huffman

```
   # Header set to be encoded
   :method: GET
   :scheme: http
   :path: /
   :authority: www.foo.com

   # Hexdump of encoded data which will be decoded
   02 03 47 45 54 07 04 68 74 74 70 06 01 2f 04 0b | ..GET..http../..
   77 77 77 2e 66 6f 6f 2e 63 6f 6d                | www.foo.com

   # Decoded opcodes
   LITERAL_INCREMENTAL_OPCODE:
           opcodeLengthInBits: 2
```

```
            discoveredFromPeekingAtByte: '02'
        name_index:
```



```
            encoded: "02"
            decoded: 2
        value_data_length:
            encoded: "03"
            decoded: 3
        value_data:
            is_huffman_encoded: 0
            encoded: "474554"
            decoded: "GET"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '07'
        name_index:
            encoded: "07"
            decoded: 7
        value_data_length:
            encoded: "04"
            decoded: 4
        value_data:
            is_huffman_encoded: 0
            encoded: "68747470"
            decoded: "http"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '06'
        name_index:
            encoded: "06"
            decoded: 6
        value_data_length:
            encoded: "01"
            decoded: 1
        value_data:
            is_huffman_encoded: 0
            encoded: "2f"
            decoded: "/"

    LITERAL_INCREMENTAL_OPCODE:
```

```
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '04'
        name_index:
            encoded: "04"
            decoded: 4
        value_data_length:
            encoded: "0b"
            decoded: 11
        value_data:
```



```
            is_huffman_encoded: 0
            encoded: "7777772e666f6f2e636f6d"
            decoded: "www.foo.com"


   # Decoded header set
   :authority: www.foo.com
   :method: GET
   :path: /
   :scheme: http

   ##########################

   # Header set to be encoded
   :method: GET
   :scheme: https
   :path: /
   :authority: www.bar.com
   cache-control: no-cache

   # Hexdump of encoded data which will be decoded
   03 05 68 74 74 70 73 02 0b 77 77 77 2e 62 61 72 | ..https..www.bar
   2e 63 6f 6d 1d 08 6e 6f 2d 63 61 63 68 65 83 85 | .com..no-cache..

   # Decoded opcodes
   LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '03'
        name_index:
            encoded: "03"
            decoded: 3
        value_data_length:
```

```
            encoded: "05"
            decoded: 5
        value_data:
            is_huffman_encoded: 0
            encoded: "6874747073"
            decoded: "https"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '02'
        name_index:
            encoded: "02"
            decoded: 2
        value_data_length:
            encoded: "0b"
            decoded: 11
```



```
        value_data:
            is_huffman_encoded: 0
            encoded: "7777772e6261722e636f6d"
            decoded: "www.bar.com"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '1d'
        name_index:
            encoded: "1d"
            decoded: 29
        value_data_length:
            encoded: "08"
            decoded: 8
        value_data:
            is_huffman_encoded: 0
            encoded: "6e6f2d6361636865"
            decoded: "no-cache"

    INDEX_OPCODE:
            opcodeLengthInBits: 1
            discoveredFromPeekingAtByte: '83'
        entry_index:
            encoded: "83"
            decoded: 3
```

```
   INDEX_OPCODE:
           opcodeLengthInBits: 1
           discoveredFromPeekingAtByte: '85'
       entry_index:
           encoded: "85"
           decoded: 5


   # Decoded header set
   :authority: www.bar.com
   :method: GET
   :path: /
   :scheme: https
   cache-control: no-cache

   ########################

   # Header set to be encoded
   :method: GET
   :scheme: https
   :path: /custom-path.css
   :authority: www.bar.com
```



```
   custom-key: custom-value

   # Hexdump of encoded data which will be decoded
   05 10 2f 63 75 73 74 6f 6d 2d 70 61 74 68 2e 63 | ../custom-path.c
   73 73 00 0a 63 75 73 74 6f 6d 2d 6b 65 79 0c 63 | ss..custom-key.c
   75 73 74 6f 6d 2d 76 61 6c 75 65 82 86          | ustom-value..

   # Decoded opcodes
   LITERAL_INCREMENTAL_OPCODE:
           opcodeLengthInBits: 2
           discoveredFromPeekingAtByte: '05'
       name_index:
           encoded: "05"
           decoded: 5
       value_data_length:
           encoded: "10"
           decoded: 16
       value_data:
```

```
            is_huffman_encoded: 0
            encoded: "2f637573746f6d2d706174682e637373"
            decoded: "/custom-path.css"

    LITERAL_INCREMENTAL_OPCODE:
            opcodeLengthInBits: 2
            discoveredFromPeekingAtByte: '00'
        name_index:
            encoded: "00"
            decoded: 0
        name_data_length:
            encoded: "0a"
            decoded: 10
        name_data:
            is_huffman_encoded: 0
            encoded: "637573746f6d2d6b6579"
            decoded: "custom-key"
        value_data_length:
            encoded: "0c"
            decoded: 12
        value_data:
            is_huffman_encoded: 0
            encoded: "637573746f6d2d76616c7565"
            decoded: "custom-value"

    INDEX_OPCODE:
            opcodeLengthInBits: 1
            discoveredFromPeekingAtByte: '82'
        entry_index:
            encoded: "82"
```



```
            decoded: 2

    INDEX_OPCODE:
            opcodeLengthInBits: 1
            discoveredFromPeekingAtByte: '86'
        entry_index:
            encoded: "86"
            decoded: 6


    # Decoded header set
```

```
   :authority: www.bar.com
   :method: GET
   :path: /custom-path.css
   :scheme: https
   custom-key: custom-value

   ########################
```

E.3. Response Decoding Example With Huffman

```
# Header set to be encoded
:status: 302
cache-control: private
date: Mon, 21 OCt 2013 20:13:21 GMT
location: : https://www.bar.com

# Hexdump of encoded data which will be decoded
08 82 40 9f 18 86 c3 1b 39 bf 38 7f 22 92 a2 fb | ..@.....9.8."...
a2 03 20 f2 eb cc 0c 49 00 62 d2 43 4c 82 7a 1d | .. ....I.b.CL.z.
2f 91 68 71 cf 3c 32 6e bd 7e 9e 9e 92 6e 7e 32 | /.hq.<2n.~...n~2
55 7d bf                                        | U}.

# Decoded opcodes
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '08'
    name_index:
        encoded: "08"
        decoded: 8
    value_data_length:
        encoded: "82"
        decoded: 2
    value_data:
        is_huffman_encoded: 1
```



```
        encoded: "409f"
        decoded: "302"

LITERAL_INCREMENTAL_OPCODE:
```

```
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '18'
    name_index:
        encoded: "18"
        decoded: 24
    value_data_length:
        encoded: "86"
        decoded: 6
    value_data:
        is_huffman_encoded: 1
        encoded: "c31b39bf387f"
        decoded: "private"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '22'
    name_index:
        encoded: "22"
        decoded: 34
    value_data_length:
        encoded: "92"
        decoded: 18
    value_data:
        is_huffman_encoded: 1
        encoded: "a2fba20320f2ebcc0c490062d2434c827a1d"
        decoded: "Mon, 21 OCt 2013 20:13:21 GMT"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '2f'
    name_index:
        encoded: "2f"
        decoded: 47
    value_data_length:
        encoded: "91"
        decoded: 17
    value_data:
        is_huffman_encoded: 1
        encoded: "6871cf3c326ebd7e9e9e926e7e32557dbf"
        decoded: ": https://www.bar.com"


# Decoded header set
:status: 302
```

```
cache-control: private
date: Mon, 21 OCt 2013 20:13:21 GMT
location: : https://www.bar.com

#########################

# Header set to be encoded
:status: 200
cache-control: private
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
content-encoding: gzip
set-cookie: foo=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAALASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ1234 max-age=3600; version=1

# Hexdump of encoded data which will be decoded
04 82 31 1f 03 92 a2 fb a2 03 20 f2 eb cc 0c 49 | ..1....... ....I
00 62 d2 43 4c c2 7a 1d 03 90 e3 9e 78 64 dd 7a | .b.CL.z.....xd.z
fd 3d 3d 24 dc fc 64 aa fb 7f 20 84 e1 fb b3 0f | .==$..d... .....
3d ff ee 02 df 7d fb 36 ed db b7 6e dd bb 76 ed | =....}.6...n..v.
db b7 6e dd bb 76 ed db b7 6e dd bb 76 ed db b7 | ..n..v...n..v...
6e dd bb 76 ed db b7 6e dd bb 76 ed db b7 6e dd | n..v...n..v...n.
bb 76 ed db b7 6e dd bb 76 ed db b7 6e dd bb 76 | .v...n..v...n..v
ed db b7 6e dd bb 7e 3b 69 ec f0 fe 7e 1f d7 f3 | ...n..~;i...~...
d5 fe 7f 7e 5f d7 9f 6f 97 cb bf e9 b7 fb fe bc | ...~_..o........
fb 7c bb fe 9b 7f bf 8f 87 f3 f0 fe bc fc bb 7b | .|.............{
fe 9b 7e 3f d7 9f 6f 97 7f d3 6f f7 f1 fe bb 7e | ..~?..o...o....~
9b 7f bf 8f c7 f9 f0 db 4f 67 f5 e7 db e5 f4 ef | ........Og......
db fd f8 91 a1 3f 1d b4 f6 78 7f 3f 0f eb f9 ea | .....?...x.?....
ff 3f bf 2f eb cf b7 cb e5 df f4 db fd ff 5e 7d | .?./..........^}
be 5d ff 4d bf df c7 c3 f9 f8 7f 5e 7e 5d bd ff | .].M.......^~]..
4d bf 1f eb cf b7 cb bf e9 b7 fb f8 ff 5d bf 4d | M............].M
bf df c7 e3 fc f8 6d a7 b3 fa f3 ed f2 fa 77 ed | ......m.......w.
fe fc 48 d0 9f 8e da 7b 3c 3f 9f 87 f5 fc f5 7f | ..H....{<?......
9f df 97 f5 e7 db e5 f2 ef fa 6d fe ff af 3e df | ..........m...>.
2e ff a6 df ef e3 e1 fc fc 3f af 3f 2e de ff a6 | .........?.?....
df 8f f5 e7 db e5 df f4 db fd fc 7f ae df a6 df | ................
ef e3 f1 fe 7c 36 d3 d9 fd 79 f6 f9 7d 3b f6 ff | ....|6...y..};..
7e 24 68 4f c7 6d 3d 9e 1f cf c3 fa fe 7a bf cf | ~$hO.m=......z..
ef cb fa f3 ed f2 f9 77 fd 36 ff 7f d7 9f 6f 97 | .......w.6....o.
7f d3 6f f7 f1 f0 fe 7e 1f d7 9f 97 6f 7f d3 6f | ..o....~....o..o
c7 fa f3 ed f2 ef fa 6d fe fe 3f d7 6f d3 6f f7 | .......m..?.o.o.
```

```
f1 f8 ff 3e 1b 69 ec fe bc fb 7c be 9d fb 7f bf | ...>.i....|.....
```



```
12 34 27 fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f | .4'..?....?....?
cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf | ....?....?....?.
f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 | ...?....?....?..
fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc | ..?....?....?...
ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff | .?....?....?....
3f cf f3 fc ff 3f cf f3 fc ff 3f cf f0 8d 09 0b | ?....?....?.....
5f d2 37 f0 86 c4 4a 23 ef 0e 70 c7 2b 2f bb 61 | _.7...J#..p.+/.a
7f 85 86 88                                     | ....

# Decoded opcodes
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '04'
    name_index:
        encoded: "04"
        decoded: 4
    value_data_length:
        encoded: "82"
        decoded: 2
    value_data:
        is_huffman_encoded: 1
        encoded: "311f"
        decoded: "200"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '03'
    name_index:
        encoded: "03"
        decoded: 3
    value_data_length:
        encoded: "92"
        decoded: 18
    value_data:
        is_huffman_encoded: 1
        encoded: "a2fba20320f2ebcc0c490062d2434cc27a1d"
        decoded: "Mon, 21 OCt 2013 20:13:22 GMT"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
```

```
        discoveredFromPeekingAtByte: '03'
    name_index:
        encoded: "03"
        decoded: 3
    value_data_length:
        encoded: "90"
        decoded: 16
    value_data:
```



```
        is_huffman_encoded: 1
        encoded: "e39e7864dd7afd3d3d24dcfc64aafb7f"
        decoded: "https://www.bar.com"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '20'
    name_index:
        encoded: "20"
        decoded: 32
    value_data_length:
        encoded: "84"
        decoded: 4
    value_data:
        is_huffman_encoded: 1
        encoded: "e1fbb30f"
        decoded: "gzip"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '3d'
    name_index:
        encoded: "3d"
        decoded: 61
    value_data_length:
        encoded: "ffee02"
        decoded: 493
    value_data:
        is_huffman_encoded: 1
        encoded: "df7dfb36eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76edd\
bb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddb\
b76eddbb7e3b69ecf0fe7e1fd7f3d5fe7f7e5fd79f6f97cbbfe9b7fbfebcfb7cbbfe9b7\
fbf8f87f3f0febcfcbb7bfe9b7e3fd79f6f977fd36ff7f1febb7e9b7fbf8fc7f9f0db4f\
```

```
67f5e7dbe5f4efdbfdf891a13f1db4f6787f3f0febf9eaff3fbf2febcfb7cbe5dff4dbf\
dff5e7dbe5dff4dbfdfc7c3f9f87f5e7e5dbdff4dbf1febcfb7cbbfe9b7fbf8ff5dbf4d\
bfdfc7e3fcf86da7b3faf3edf2fa77edfefc48d09f8eda7b3c3f9f87f5fcf57f9fdf97f\
5e7dbe5f2effa6dfeffaf3edf2effa6dfefe3e1fcfc3faf3f2edeffa6df8ff5e7dbe5df\
f4dbfdfc7faedfa6dfefe3f1fe7c36d3d9fd79f6f97d3bf6ff7e24684fc76d3d9e1fcfc\
3fafe7abfcfefcbfaf3edf2f977fd36ff7fd79f6f977fd36ff7f1f0fe7e1fd79f976f7f\
d36fc7faf3edf2effa6dfefe3fd76fd36ff7f1f8ff3e1b69ecfebcfb7cbe9dfb7fbf123\
427fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcf\
f3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff\
3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff08d090b5fd237f086c44a23ef0e70c7\
2b2fbb617f"
        decoded: "foo=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAALASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIU\
AXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQ\
WEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOE\
```



```
IUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJK\
HQWOEIUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ1234max-age=3600; version=1\
"

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '85'
    entry_index:
        encoded: "85"
        decoded: 5

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '86'
    entry_index:
        encoded: "86"
        decoded: 6

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '88'
    entry_index:
        encoded: "88"
        decoded: 8
```

```
# Decoded header set
:status: 200
cache-control: private
content-encoding: gzip
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
set-cookie: foo=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAALASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ1234 max-age=3600; version=1

##########################

# Header set to be encoded
:status: 200
cache-control: private
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
```



```
content-encoding: gzip
set-cookie: foo=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZLASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1234 max-age=3600; version=1

# Hexdump of encoded data which will be decoded
01 ff ee 02 df 7d fb 3f cf f3 fc ff 3f cf f3 fc | .....}.?....?...
ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff | .?....?....?....
3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f | ?....?....?....?
cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf | ....?....?....?.
f3 fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 | ...?....?....?..
fc ff 3f cf f3 fc ff 3f cf f3 fc ff 3f cf f3 fc | ..?....?....?...
ff 3e 3b 69 ec f0 fe 7e 1f d7 f3 d5 fe 7f 7e 5f | .>;i...~......~_
d7 9f 6f 97 cb bf e9 b7 fb fe bc fb 7c bb fe 9b | ..o.........|...
7f bf 8f 87 f3 f0 fe bc fc bb 7b fe 9b 7e 3f d7 | ..........{..~?.
```

```
9f 6f 97 7f d3 6f f7 f1 fe bb 7e 9b 7f bf 8f c7 | .o...o....~.....
f9 f0 db 4f 67 f5 e7 db e5 f4 ef db fd f8 91 a1 | ...Og...........
3f 1d b4 f6 78 7f 3f 0f eb f9 ea ff 3f bf 2f eb | ?...x.?.....?./.
cf b7 cb e5 df f4 db fd ff 5e 7d be 5d ff 4d bf | .........^}.].M.
df c7 c3 f9 f8 7f 5e 7e 5d bd ff 4d bf 1f eb cf | ......^~]..M....
b7 cb bf e9 b7 fb f8 ff 5d bf 4d bf df c7 e3 fc | ........].M.....
f8 6d a7 b3 fa f3 ed f2 fa 77 ed fe fc 48 d0 9f | .m.......w...H..
8e da 7b 3c 3f 9f 87 f5 fc f5 7f 9f df 97 f5 e7 | ..{<?...........
db e5 f2 ef fa 6d fe ff af 3e df 2e ff a6 df ef | .....m...>......
e3 e1 fc fc 3f af 3f 2e de ff a6 df 8f f5 e7 db | ....?.?.........
e5 df f4 db fd fc 7f ae df a6 df ef e3 f1 fe 7c | ...............|
36 d3 d9 fd 79 f6 f9 7d 3b f6 ff 7e 24 68 4f c7 | 6...y..};..~$hO.
6d 3d 9e 1f cf c3 fa fe 7a bf cf ef cb fa f3 ed | m=......z.......
f2 f9 77 fd 36 ff 7f d7 9f 6f 97 7f d3 6f f7 f1 | ..w.6....o...o..
f0 fe 7e 1f d7 9f 97 6f 7f d3 6f c7 fa f3 ed f2 | ..~....o..o.....
ef fa 6d fe fe 3f d7 6f d3 6f f7 f1 f8 ff 3e 1b | ..m..?.o.o....>.
69 ec fe bc fb 7c be 9d fb 7f bf 12 34 27 6e dd | i....|......4'n.
bb 76 ed db b7 6e dd bb 76 ed db b7 6e dd bb 76 | .v...n..v...n..v
ed db b7 6e dd bb 76 ed db b7 6e dd bb 76 ed db | ...n..v...n..v..
b7 6e dd bb 76 ed db b7 6e dd bb 76 ed db b7 6e | .n..v...n..v...n
dd bb 76 ed db b7 6e dd bb 76 ed db b4 8d 09 0b | ..v...n..v......
5f d2 37 f0 86 c4 4a 23 ef 0e 70 c7 2b 2f bb 61 | _.7...J#..p.+/.a
7f 81                                           | ..
```

```
# Decoded opcodes
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '01'
    name_index:
```



```
        encoded: "01"
        decoded: 1
    value_data_length:
        encoded: "ffee02"
        decoded: 493
    value_data:
        is_huffman_encoded: 1
        encoded: "df7dfb3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3f\
cff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fc\
ff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3fcff3e3b69ecf\
0fe7e1fd7f3d5fe7f7e5fd79f6f97cbbfe9b7fbfebcfb7cbbfe9b7fbf8f87f3f0febcfc\
bb7bfe9b7e3fd79f6f977fd36ff7f1febb7e9b7fbf8fc7f9f0db4f67f5e7dbe5f4efdbf\
```

```
df891a13f1db4f6787f3f0febf9eaff3fbf2febcfb7cbe5dff4dbfdff5e7dbe5dff4dbf\
dfc7c3f9f87f5e7e5dbdff4dbf1febcfb7cbbfe9b7fbf8ff5dbf4dbfdfc7e3fcf86da7b\
3faf3edf2fa77edfefc48d09f8eda7b3c3f9f87f5fcf57f9fdf97f5e7dbe5f2effa6dfe\
ffaf3edf2effa6dfefe3e1fcfc3faf3f2edeffa6df8ff5e7dbe5dff4dbfdfc7faedfa6d\
fefe3f1fe7c36d3d9fd79f6f97d3bf6ff7e24684fc76d3d9e1fcfc3fafe7abfcfefcbfa\
f3edf2f977fd36ff7fd79f6f977fd36ff7f1f0fe7e1fd79f976f7fd36fc7faf3edf2eff\
a6dfefe3fd76fd36ff7f1f8ff3e1b69ecfebcfb7cbe9dfb7fbf1234276eddbb76eddbb7\
6eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76eddbb76\
eddbb76eddbb76eddbb76eddbb76eddbb76eddbb48d090b5fd237f086c44a23ef0e70c7\
2b2fbb617f"
        decoded: "foo=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZLASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIU\
AXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQ\
WEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOE\
IUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJK\
HQWOEIUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234AAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1234max-age=3600; version=1\
"

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '81'
    entry_index:
        encoded: "81"
        decoded: 1


# Decoded header set
:status: 200
cache-control: private
content-encoding: gzip
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
set-cookie: foo=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZLASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
```



```
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1234 max-age=3600; version=1
```

```
#########################
```

E.4. Response Decoding Example Without Huffman

```
# Header set to be encoded
:status: 302
cache-control: private
date: Mon, 21 OCt 2013 20:13:21 GMT
location: : https://www.bar.com

# Hexdump of encoded data which will be decoded
08 03 33 30 32 18 07 70 72 69 76 61 74 65 22 1d | ..302..private".
4d 6f 6e 2c 20 32 31 20 4f 43 74 20 32 30 31 33 | Mon, 21 OCt 2013
20 32 30 3a 31 33 3a 32 31 20 47 4d 54 2f 15 3a |  20:13:21 GMT/.:
20 68 74 74 70 73 3a 2f 2f 77 77 77 2e 62 61 72 |  https://www.bar
2e 63 6f 6d                                     | .com

# Decoded opcodes
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '08'
    name_index:
        encoded: "08"
        decoded: 8
    value_data_length:
        encoded: "03"
        decoded: 3
    value_data:
        is_huffman_encoded: 0
        encoded: "333032"
        decoded: "302"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '18'
    name_index:
        encoded: "18"
        decoded: 24
    value_data_length:
        encoded: "07"
```

```
        decoded: 7
    value_data:
        is_huffman_encoded: 0
        encoded: "70726976617465"
        decoded: "private"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '22'
    name_index:
        encoded: "22"
        decoded: 34
    value_data_length:
        encoded: "1d"
        decoded: 29
    value_data:
        is_huffman_encoded: 0
        encoded: "4d6f6e2c203231204f4374203230313320323033a31333a3231204\
74d54"
        decoded: "Mon, 21 OCt 2013 20:13:21 GMT"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '2f'
    name_index:
        encoded: "2f"
        decoded: 47
    value_data_length:
        encoded: "15"
        decoded: 21
    value_data:
        is_huffman_encoded: 0
        encoded: "3a2068747470733a2f2f7777772e6261722e636f6d"
        decoded: ": https://www.bar.com"


# Decoded header set
:status: 302
cache-control: private
date: Mon, 21 OCt 2013 20:13:21 GMT
location: : https://www.bar.com

##########################

# Header set to be encoded
:status: 200
cache-control: private
date: Mon, 21 OCt 2013 20:13:22 GMT
```

```
location: https://www.bar.com
content-encoding: gzip
set-cookie: foo=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAALASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ1234 max-age=3600; version=1

# Hexdump of encoded data which will be decoded
04 03 32 30 30 03 1d 4d 6f 6e 2c 20 32 31 20 4f | ..200..Mon, 21 O
43 74 20 32 30 31 33 20 32 30 3a 31 33 3a 32 32 | Ct 2013 20:13:22
20 47 4d 54 03 13 68 74 74 70 73 3a 2f 2f 77 77 |  GMT..https://ww
77 2e 62 61 72 2e 63 6f 6d 20 04 67 7a 69 70 3d | w.bar.com .gzip=
7f e1 02 66 6f 6f 3d 41 41 41 41 41 41 41 41 41 | ...foo=AAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 4c | AAAAAAAAAAAAAAAL
41 53 44 4a 4b 48 51 4b 42 5a 58 4f 51 57 45 4f | ASDJKHQKBZXOQWEO
50 49 55 41 58 51 57 45 4f 49 55 41 58 4c 4a 4b | PIUAXQWEOIUAXLJK
48 51 57 4f 45 49 55 41 4c 51 57 45 4f 49 55 41 | HQWOEIUALQWEOIUA
58 4c 51 45 55 41 58 4c 4c 4b 4a 41 53 44 51 57 | XLQEUAXLLKJASDQW
45 4f 55 49 41 58 4e 31 32 33 34 4c 41 53 44 4a | EOUIAXN1234LASDJ
4b 48 51 4b 42 5a 58 4f 51 57 45 4f 50 49 55 41 | KHQKBZXOQWEOPIUA
58 51 57 45 4f 49 55 41 58 4c 4a 4b 48 51 57 4f | XQWEOIUAXLJKHQWO
45 49 55 41 4c 51 57 45 4f 49 55 41 58 4c 51 45 | EIUALQWEOIUAXLQE
55 41 58 4c 4c 4b 4a 41 53 44 51 57 45 4f 55 49 | UAXLLKJASDQWEOUI
41 58 4e 31 32 33 34 4c 41 53 44 4a 4b 48 51 4b | AXN1234LASDJKHQK
42 5a 58 4f 51 57 45 4f 50 49 55 41 58 51 57 45 | BZXOQWEOPIUAXQWE
4f 49 55 41 58 4c 4a 4b 48 51 57 4f 45 49 55 41 | OIUAXLJKHQWOEIUA
4c 51 57 45 4f 49 55 41 58 4c 51 45 55 41 58 4c | LQWEOIUAXLQEUAXL
4c 4b 4a 41 53 44 51 57 45 4f 55 49 41 58 4e 31 | LKJASDQWEOUIAXN1
32 33 34 4c 41 53 44 4a 4b 48 51 4b 42 5a 58 4f | 234LASDJKHQKBZXO
51 57 45 4f 50 49 55 41 58 51 57 45 4f 49 55 41 | QWEOPIUAXQWEOIUA
58 4c 4a 4b 48 51 57 4f 45 49 55 41 4c 51 57 45 | XLJKHQWOEIUALQWE
4f 49 55 41 58 4c 51 45 55 41 58 4c 4c 4b 4a 41 | OIUAXLQEUAXLLKJA
53 44 51 57 45 4f 55 49 41 58 4e 31 32 33 34 5a | SDQWEOUIAXN1234Z
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
```

```
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 31 32 33 34 20 6d 61 78 2d | ZZZZZZZ1234 max-
61 67 65 3d 33 36 30 30 3b 20 76 65 72 73 69 6f | age=3600; versio
6e 3d 31 85 86 88                               | n=1...

# Decoded opcodes
```



```
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '04'
    name_index:
        encoded: "04"
        decoded: 4
    value_data_length:
        encoded: "03"
        decoded: 3
    value_data:
        is_huffman_encoded: 0
        encoded: "323030"
        decoded: "200"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '03'
    name_index:
        encoded: "03"
        decoded: 3
    value_data_length:
        encoded: "1d"
        decoded: 29
    value_data:
        is_huffman_encoded: 0
        encoded: "4d6f6e2c203231204f4374203230313320323033a31333a3232204\
74d54"
        decoded: "Mon, 21 OCt 2013 20:13:22 GMT"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '03'
    name_index:
        encoded: "03"
        decoded: 3
```

```
    value_data_length:
        encoded: "13"
        decoded: 19
    value_data:
        is_huffman_encoded: 0
        encoded: "68747470733a2f2f7777772e6261722e636f6d"
        decoded: "https://www.bar.com"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '20'
    name_index:
        encoded: "20"
```



```
        decoded: 32
    value_data_length:
        encoded: "04"
        decoded: 4
    value_data:
        is_huffman_encoded: 0
        encoded: "677a6970"
        decoded: "gzip"

LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '3d'
    name_index:
        encoded: "3d"
        decoded: 61
    value_data_length:
        encoded: "7fe102"
        decoded: 480
    value_data:
        is_huffman_encoded: 0
        encoded: "666f6f3d4141414141414141414141414141414141414141414\
14141414141414141414141414141414141414141414141414141414141414141\
41414141414141414141414141414c4153444a4b48514b425a584f5157454f504955415\
85157454f495541584c4a4b4851574f454955414c5157454f495541584c51455541584c\
4c4b4a4153445157454f554941584e313233344c4153444a4b48514b425a584f5157454\
f504955415851574f495541584c4a4b4851574f454955414c5157454f495541584c51\
455541584c4c4b4a4153445157454f554941584e313233344c4153444a4b48514b425a5\
84f5157454f5049554158515745 4f495541584c4a4b4851574f454955414c5157454f49\
```

```
5541584c51455541584c4c4b4a4153445157454f554941584e313233344c4153444a4b4\
8514b425a584f5157454f504955415851574f495541584c4a4b4851574f454955414c\
5157454f495541584c51455541584c4c4b4a4153445157454f554941584e313233345a5\
a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a\
5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a3\
1323334206d61782d6167653d333630303b2076657273696f6e3d31"
        decoded: "foo=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAALASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIU\
AXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQ\
WEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOE\
IUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJK\
HQWOEIUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234ZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ1234max-age=3600; version=1\
"

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '85'
    entry_index:
        encoded: "85"
```



```
        decoded: 5

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '86'
    entry_index:
        encoded: "86"
        decoded: 6

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '88'
    entry_index:
        encoded: "88"
        decoded: 8


# Decoded header set
:status: 200
cache-control: private
content-encoding: gzip
```

```
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
set-cookie: foo=AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAALASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ1234 max-age=3600; version=1

##########################

# Header set to be encoded
:status: 200
cache-control: private
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
content-encoding: gzip
set-cookie: foo=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZLASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1234 max-age=3600; version=1

# Hexdump of encoded data which will be decoded
```



```
01 7f e1 02 66 6f 6f 3d 5a 5a 5a 5a 5a 5a 5a 5a | ....foo=ZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a 5a | ZZZZZZZZZZZZZZZZ
4c 41 53 44 4a 4b 48 51 4b 42 5a 58 4f 51 57 45 | LASDJKHQKBZXOQWE
4f 50 49 55 41 58 51 57 45 4f 49 55 41 58 4c 4a | OPIUAXQWEOIUAXLJ
4b 48 51 57 4f 45 49 55 41 4c 51 57 45 4f 49 55 | KHQWOEIUALQWEOIU
41 58 4c 51 45 55 41 58 4c 4c 4b 4a 41 53 44 51 | AXLQEUAXLLKJASDQ
57 45 4f 55 49 41 58 4e 31 32 33 34 4c 41 53 44 | WEOUIAXN1234LASD
4a 4b 48 51 4b 42 5a 58 4f 51 57 45 4f 50 49 55 | JKHQKBZXOQWEOPIU
41 58 51 57 45 4f 49 55 41 58 4c 4a 4b 48 51 57 | AXQWEOIUAXLJKHQW
4f 45 49 55 41 4c 51 57 45 4f 49 55 41 58 4c 51 | OEIUALQWEOIUAXLQ
45 55 41 58 4c 4c 4b 4a 41 53 44 51 57 45 4f 55 | EUAXLLKJASDQWEOU
```

```
49 41 58 4e 31 32 33 34 4c 41 53 44 4a 4b 48 51 | IAXN1234LASDJKHQ
4b 42 5a 58 4f 51 57 45 4f 50 49 55 41 58 51 57 | KBZXOQWEOPIUAXQW
45 4f 49 55 41 58 4c 4a 4b 48 51 57 4f 45 49 55 | EOIUAXLJKHQWOEIU
41 4c 51 57 45 4f 49 55 41 58 4c 51 45 55 41 58 | ALQWEOIUAXLQEUAX
4c 4c 4b 4a 41 53 44 51 57 45 4f 55 49 41 58 4e | LLKJASDQWEOUIAXN
31 32 33 34 4c 41 53 44 4a 4b 48 51 4b 42 5a 58 | 1234LASDJKHQKBZX
4f 51 57 45 4f 50 49 55 41 58 51 57 45 4f 49 55 | OQWEOPIUAXQWEOIU
41 58 4c 4a 4b 48 51 57 4f 45 49 55 41 4c 51 57 | AXLJKHQWOEIUALQW
45 4f 49 55 41 58 4c 51 45 55 41 58 4c 4c 4b 4a | EOIUAXLQEUAXLLKJ
41 53 44 51 57 45 4f 55 49 41 58 4e 31 32 33 34 | ASDQWEOUIAXN1234
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 | AAAAAAAAAAAAAAAA
41 41 41 41 41 41 41 31 32 33 34 20 6d 61 78 | AAAAAAA1234 max
2d 61 67 65 3d 33 36 30 30 3b 20 76 65 72 73 69 | -age=3600; versi
6f 6e 3d 31 81                                  | on=1.
```

```
# Decoded opcodes
LITERAL_INCREMENTAL_OPCODE:
        opcodeLengthInBits: 2
        discoveredFromPeekingAtByte: '01'
    name_index:
        encoded: "01"
        decoded: 1
    value_data_length:
        encoded: "7fe102"
        decoded: 480
    value_data:
        is_huffman_encoded: 0
        encoded: "666f6f3d5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5\
a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a5a\
5a5a5a5a5a5a5a5a5a5a5a5a5a5a4c4153444a4b48514b425a584f5157454f504955415\
85157454f495541584c4a4b4851574f454955414c5157454f495541584c51455541584c\
```



```
4c4b4a4153445157454f554941584e313233344c4153444a4b48514b425a584f5157454\
f5049554158515745 4f495541584c4a4b4851574f454955414c5157454f495541584c51\
455541584c4c4b4a4153445157454f554941584e313233344c4153444a4b48514b425a5\
84f5157454f5049554158515745 4f495541584c4a4b4851574f454955414c5157454f49\
5541584c51455541584c4c4b4a4153445157454f554941584e313233344c4153444a4b4\
8514b425a584f5157454f5049554158515745 4f495541584c4a4b4851574f454955414c\
5157454f495541584c51455541584c4c4b4a4153445157454f554941584e31323334414\
```

```
1414141414141414141414141414141414141414141414141414141414141414141\
4141414141414141414141414141414141414141414141414141414141414141413\
1323334206d61782d6167653d333630303b2076657273696f6e3d31"
        decoded: "foo=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZZZZZZZLASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIU\
AXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQ\
WEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOE\
IUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJK\
HQWOEIUALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234AAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1234max-age=3600; version=1\
"

INDEX_OPCODE:
        opcodeLengthInBits: 1
        discoveredFromPeekingAtByte: '81'
    entry_index:
        encoded: "81"
        decoded: 1


# Decoded header set
:status: 200
cache-control: private
content-encoding: gzip
date: Mon, 21 OCt 2013 20:13:22 GMT
location: https://www.bar.com
set-cookie: foo=ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ\
ZZZZZZZZZZZZZZZZZLASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUAXLQEU\
AXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQWEOIUA\
XLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEIUALQW\
EOIUAXLQEUAXLLKJASDQWEOUIAXN1234LASDJKHQKBZXOQWEOPIUAXQWEOIUAXLJKHQWOEI\
UALQWEOIUAXLQEUAXLLKJASDQWEOUIAXN1234AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1234 max-age=3600; version=1

##########################
```

Authors' Addresses

Roberto Peon
Google, Inc

EMail: fenix@google.com

Herve Ruellan
Canon CRF

EMail: herve.ruellan@crf.canon.fr