QUIC
Internet-Draft
Intended status: Standards Track
Expires: November 24, 2018

C. Krasic
Google, Inc
M. Bishop
Akamai Technologies
A. Frindell, Ed.
Facebook

May 23, 2018

QPACK: Header Compression for HTTP over QUIC
draft-ietf-quic-qpack-00

Abstract

This specification defines QPACK, a compression format for efficiently representing HTTP header fields, to be used in HTTP over QUIC. This is a variation of HPACK header compression that seeks to reduce head-of-line blocking.

Note to Readers

Discussion of this draft takes place on the QUIC working group mailing list (quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=quic [1].

Working Group information can be found at https://github.com/quicwg [2]; source code and issues list for this draft can be found at https://github.com/quicwg/base-drafts/labels/-qpack [3].

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 https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 24, 2018.

Table of Contents

1. Introduction

The QUIC transport protocol was designed from the outset to support HTTP semantics, and its design subsumes many of the features of HTTP/2. QUIC's stream multiplexing comes into some conflict with header compression. A key goal of the design of QUIC is to improve stream multiplexing relative to HTTP/2 by eliminating HoL (head of line) blocking, which can occur in HTTP/2. HoL blocking can happen because all HTTP/2 streams are multiplexed onto a single TCP connection with its in-order semantics. QUIC can maintain independence between streams because it implements core transport functionality in a fully stream-aware manner. However, the HTTP/QUIC mapping is still subject to HoL blocking if HPACK is used directly. HPACK exploits multiplexing for greater compression, shrinking the representation of headers that have appeared earlier on the same connection. In the context of QUIC, this imposes a vulnerability to HoL blocking (see Section 1.1).

QUIC is described in [QUIC-TRANSPORT]. The HTTP/QUIC mapping is described in [QUIC-HTTP]. For a full description of HTTP/2, see [RFC7540]. The description of HPACK is [RFC7541], with important terminology in Section 1.3.

QPACK modifies HPACK to allow correctness in the presence of out-of-order delivery, with flexibility for implementations to balance between resilience against HoL blocking and optimal compression ratio. The design goals are to closely approach the compression ratio of HPACK with substantially less head-of-line blocking under the same loss conditions.

QPACK is intended to be a relatively non-intrusive extension to HPACK; an implementation should be easily shared within stacks supporting both HTTP/2 over (TLS+)TCP and HTTP/QUIC.

1.1. Head-of-Line Blocking in HPACK

HPACK enables several types of header representations, one of which also adds the header to a dynamic table of header values. These values are then available for reuse in subsequent header blocks simply by referencing the entry number in the table.

If the packet containing a header is lost, that stream cannot complete header processing until the packet is retransmitted. This is unavoidable. However, other streams which rely on the state created by that packet _also_ cannot make progress. This is the problem which QUIC solves in general, but which is reintroduced by HPACK when the loss includes a HEADERS frame.

1.2. Avoiding Head-of-Line Blocking in HTTP/QUIC

Because QUIC does not guarantee order between data on different streams, a header block might reference an entry in the dynamic table that has not yet been received.

Each header block contains a Largest Reference (see Section 3.5.1) which identifies the table state necessary for decoding. If the greatest absolute index in the dynamic table is less than the value of the Largest Reference, the stream is considered "blocked." While blocked, header field data should remain in the blocked stream's flow control window. When the Largest Reference is zero, the frame contains no references to the dynamic table and can always be processed immediately. A stream becomes unblocked when the greatest absolute index in the dynamic table becomes greater than or equal to the Largest Reference for all header blocks the decoder has started reading from the stream.

A decoder can permit the possibility of blocked streams by setting SETTINGS_QPACK_BLOCKED_STREAMS to a non-zero value. This setting specifies an upper bound on the number of streams which can be blocked.

An encoder can decide whether to risk having a stream become blocked. If permitted by the value of SETTINGS_QPACK_BLOCKED_STREAMS, compression efficiency can be improved by referencing dynamic table entries that are still in transit, but if there is loss or reordering the stream can become blocked at the decoder. An encoder avoids the risk of blocking by only referencing dynamic table entries which have been acknowledged, but this means using literals. Since literals make the header block larger, this can result in the encoder becoming blocked on congestion or flow control limits.

An encoder MUST limit the number of streams which could become blocked to the value of SETTINGS_QPACK_BLOCKED_STREAMS at all times. Note that the decoder might not actually become blocked on every stream which risks becoming blocked. If the decoder encounters more blocked streams than it promised to support, it SHOULD treat this as a stream error of type HTTP_QPACK_DECOMPRESSION_FAILED.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Definitions of terms that are used in this document:

Header: A name-value pair sent as part of an HTTP message.

Header set: The full collection of headers associated with an HTTP message.

Header block: The compressed representation of a header set.

Encoder: An implementation which transforms a header set into a header block.

Decoder: An implementation which transforms a header block into a header set.

QPACK is a name, not an acronym.

2.1. Notational Conventions

Diagrams use the format described in Section 3.1 of [RFC2360], with the following additional conventions:

x (A) Indicates that x is A bits long

x (A+) Indicates that x uses the prefixed integer encoding defined in Section 5.1 of [RFC7541], beginning with an A-bit prefix.

x ... Indicates that x is variable-length and extends to the end of the region.

3. Wire Format

QPACK instructions occur in three locations, each of which uses a separate instruction space:

- Table updates are carried by a unidirectional stream from encoder to decoder. Instructions on this stream modify the dynamic table state without generating output to any particular request.

- Acknowledgements of table modifications and header processing are carried by a unidirectional stream from decoder to encoder.

- Finally, the contents of HEADERS and PUSH_PROMISE frames on request streams reference the QPACK table state.

This section describes the instructions which are possible on each stream type.

All table updates occur on the control stream. Request streams only carry header blocks that do not modify the state of the table.

3.1. Primitives

The prefixed integer from Section 5.1 of [RFC7541] is used heavily throughout this document. The string literal, defined by Section 5.2 of [RFC7541], is used with the following modification.

HPACK defines string literals to begin on a byte boundary. They begin with a single flag (indicating whether the string is Huffman-coded), followed by the Length encoded as a 7-bit prefix integer, and finally Length octets of data.

QPACK permits strings to begin other than on a byte boundary. An "N-bit prefix string literal" begins with the same Huffman flag, followed by the length encoded as an (N-1)-bit prefix integer. The remainder of the string literal is unmodified.

A string literal without a prefix length noted is an 8-bit prefix string literal and follows the definitions in [RFC7541] without modification.

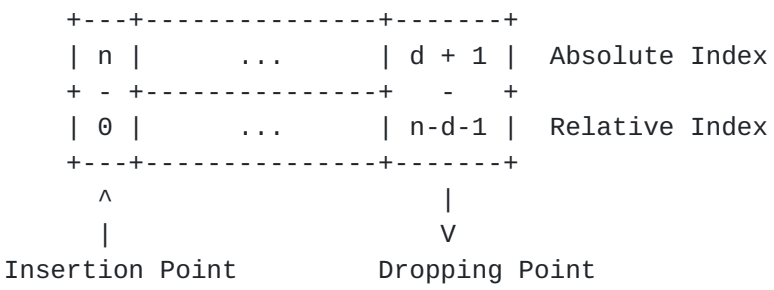
3.2. Indexing

Entries in the QPACK static and dynamic tables are addressed separately.

Entries in the static table have the same indices at all times. The static table is defined in Appendix A of [RFC7541]. Note that because HPACK did not use zero-based references, there is no value at index zero of the static table.

Entries are inserted into the dynamic table over time. Each entry possesses both an absolute index which is fixed for the lifetime of that entry and a relative index which changes over time based on the context of the reference. The first entry inserted has an absolute index of "1"; indices increase sequentially with each insertion.

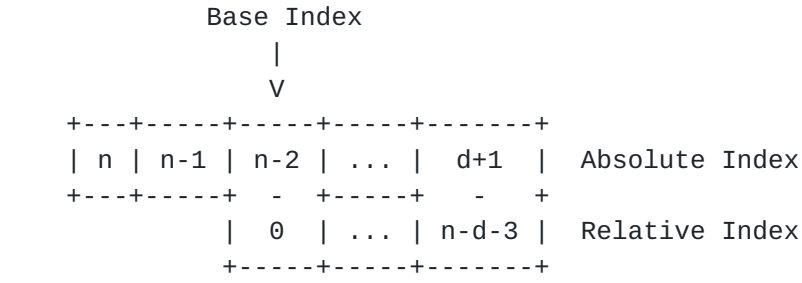
On the control stream, a relative index of "0" always refers to the most recently inserted value in the dynamic table. Note that this means the entry referenced by a given relative index can change while interpreting a HEADERS frame as new entries are inserted.



n = count of entries inserted
d = count of entries dropped

Example Dynamic Table Indexing - Control Stream

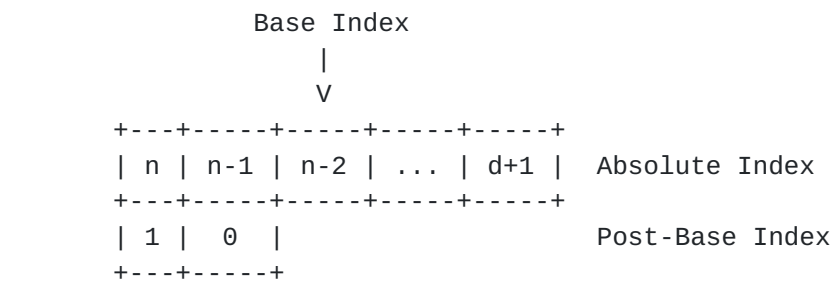
Because frames from request streams can be delivered out of order with instructions on the control stream, relative indices are relative to the Base Index at the beginning of the header block (see Section 3.5.1). The Base Index is the absolute index of the entry which has the relative index of zero when interpreting the frame. The relative indices of entries do not change while interpreting headers on a request or push stream.



n = count of entries inserted
d = count of entries dropped

Example Dynamic Table Indexing - Request Stream

Entries with an absolute index greater than a frame's Base Index can be referenced using specific Post-Base instructions. The relative indices of Post-Base references count up from Base Index.



n = count of entries inserted
d = count of entries dropped

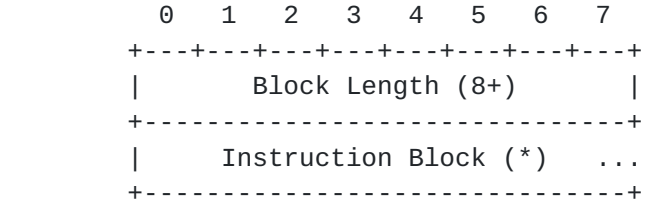
Dynamic Table Indexing - Post-Base References

If the decoder encounters a reference to an entry which has already been dropped from the table or which is greater than the declared Largest Reference, this MUST be treated as a stream error of type "HTTP_QPACK_DECOMPRESSION_FAILED" error code. If this reference occurs on the control stream, this MUST be treated as a session error.

3.3. QPACK Encoder Stream

Table updates can add a table entry, possibly using existing entries to avoid transmitting redundant information. The name can be transmitted as a reference to an existing entry in the static or the dynamic table or as a string literal. For entries which already exist in the dynamic table, the full entry can also be used by reference, creating a duplicate entry.

Each set of encoder instructions is prefaced by its length, encoded as a variable length integer with an 8-bit prefix. Instructions MUST NOT span more than one block.



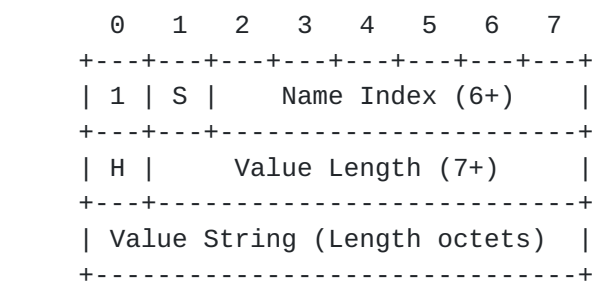
Encoder instruction block

3.3.1. Insert With Name Reference

An addition to the header table where the header field name matches the header field name of an entry stored in the static table or the dynamic table starts with the '1' one-bit pattern. The "S" bit

indicates whether the reference is to the static (S=1) or dynamic (S=0) table. The header field name is represented using the relative index of that entry, which is represented as an integer with a 6-bit prefix (see Section 5.1 of [RFC7541]).

The header name reference is followed by the header field value represented as a string literal (see Section 5.2 of [RFC7541]).

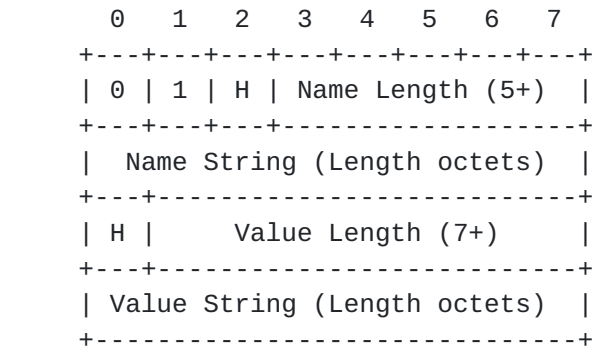


Insert Header Field -- Indexed Name

3.3.2. Insert Without Name Reference

An addition to the header table where both the header field name and the header field value are represented as string literals (see Section 3.1) starts with the '01' two-bit pattern.

The name is represented as a 6-bit prefix string literal, while the value is represented as an 8-bit prefix string literal.



Insert Header Field -- New Name

3.3.3. Duplicate

Duplication of an existing entry in the dynamic table starts with the '000' three-bit pattern. The relative index of the existing entry is represented as an integer with a 5-bit prefix.

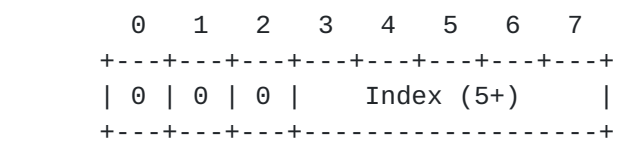


Figure 1: Duplicate

The existing entry is re-inserted into the dynamic table without resending either the name or the value. This is useful to mitigate the eviction of older entries which are frequently referenced, both to avoid the need to resend the header and to avoid the entry in the table blocking the ability to insert new headers.

3.3.4. Dynamic Table Size Update

An encoder informs the decoder of a change to the size of the dynamic table using an instruction which begins with the '001' three-bit pattern. The new maximum table size is represented as an integer with a 5-bit prefix (see Section 5.1 of [RFC7541]).

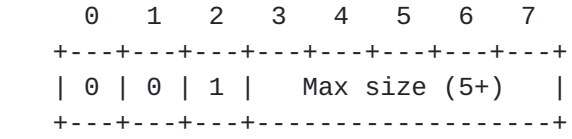


Figure 2: Maximum Dynamic Table Size Change

The new maximum size MUST be lower than or equal to the limit determined by the protocol using QPACK. A value that exceeds this limit MUST be treated as a decoding error. In HTTP/QUIC, this limit is the value of the SETTINGS_HEADER_TABLE_SIZE parameter (see [QUIC-HTTP]) received from the decoder.

Reducing the maximum size of the dynamic table can cause entries to be evicted (see Section 4.3 of [RFC7541]). This MUST NOT cause the eviction of entries with outstanding references (see Section 4.3).

3.4. QPACK Decoder Stream

The decoder stream carries information used to ensure consistency of the dynamic table. Information is sent from the QPACK decoder to the QPACK encoder; that is, the server informs the client about the processing of the client's header blocks and table updates, and the client informs the server about the processing of the server's header blocks and table updates.

3.4.1. Table State Synchronize

After processing a set of instructions on the encoder stream, the decoder will emit a Table State Synchronize instruction on the decoder stream. The instruction begins with the '1' one-bit pattern. The instruction specifies the total number of dynamic table inserts and duplications since the last Table State Synchronize, encoded as a 7-bit prefix integer. The encoder uses this value to determine which table entries are vulnerable to head-of-line blocking. A decoder MAY coalesce multiple synchronization updates into a single update.

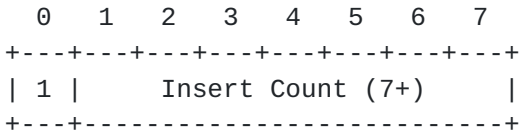


Figure 3: Table Size Synchronize

3.4.2. Header Acknowledgement

After processing a header block on a request or push stream, the decoder emits a Header Acknowledgement instruction on the decoder stream. The instruction begins with the '0' one-bit pattern and includes the request stream's stream ID, encoded as a 7-bit prefix integer. It is used by the peer's QPACK encoder to know when it is safe to evict an entry.

The same Stream ID can be identified multiple times, as multiple header blocks can be sent on a single stream in the case of intermediate responses, trailers, and pushed requests. Since header frames on each stream are received and processed in order, this gives the encoder precise feedback on which header blocks within a stream have been fully processed.

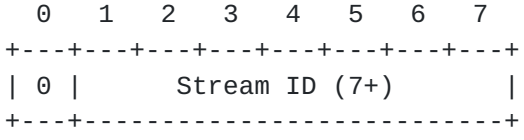


Figure 4: Header Acknowledgement

3.5. Request and Push Streams

HEADERS and PUSH_PROMISE frames on request and push streams reference the dynamic table in a particular state without modifying it. Frames on these streams emit the headers for an HTTP request or response.

3.5.1. Header Data Prefix

Header data is prefixed with two integers, "Largest Reference" and "Base Index".

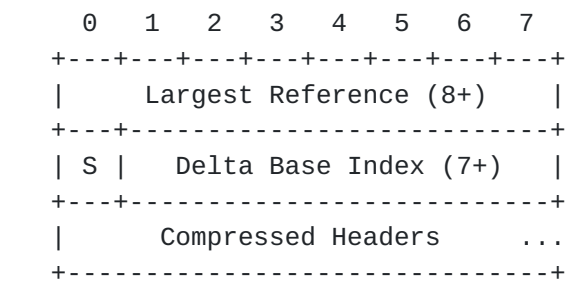


Figure 5: Frame Payload

"Largest Reference" identifies the largest absolute dynamic index referenced in the block. Blocking decoders use the Largest Reference to determine when it is safe to process the rest of the block.

"Base Index" is used to resolve references in the dynamic table as described in Section 3.2. To save space, Base Index is encoded relative to Largest Reference using a one-bit sign flag.

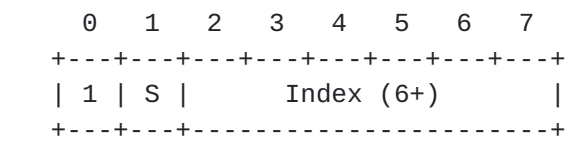
baseIndex = largestReference + deltaBaseIndex

If the encoder inserted entries to the table while the encoding the block, Largest Reference will be greater than Base Index, so deltaBaseIndex will be negative and encoded with S=1. If the block did not reference the most recent entry in the table and did not insert any new entries, Largest Reference will be less than Base Index, so deltaBaseIndex will be positive and encoded with S=0. When Largest Reference and Base Index are equal, deltaBaseIndex is 0 and encoded with S=0.

3.5.2. Instructions

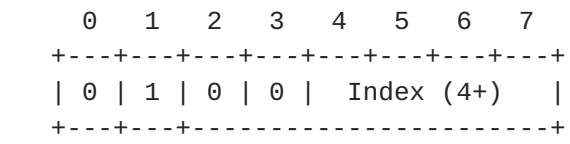
3.5.2.1. Indexed Header Field

An indexed header field representation identifies an entry in either the static table or the dynamic table and causes that header field to be added to the decoded header list, as described in Section 3.2 of [RFC7541].



Indexed Header Field

If the entry is in the static table, or in the dynamic table with an absolute index less than or equal to Base Index, this representation starts with the '1' 1-bit pattern, followed by the "S" bit indicating whether the reference is into the static (S=1) or dynamic (S=0) table. Finally, the relative index of the matching header field is represented as an integer with a 6-bit prefix (see Section 5.1 of [RFC7541]).



Indexed Header Field

If the entry is in the dynamic table with an absolute index greater than Base Index, the representation starts with the '0100' 4-bit pattern, followed by the post-base index (see Section 3.2) of the matching header field, represented as an integer with a 4-bit prefix (see Section 5.1 of [RFC7541]).

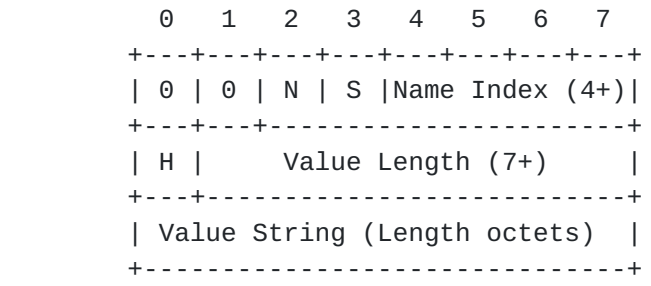
3.5.2.2. Literal Header Field With Name Reference

A literal header field with a name reference represents a header where the header field name matches the header field name of an entry stored in the static table or the dynamic table.

If the entry is in the static table, or in the dynamic table with an absolute index less than or equal to Base Index, this representation starts with the '00' two-bit pattern. If the entry is in the dynamic table with an absolute index greater than Base Index, the representation starts with the '0101' four-bit pattern.

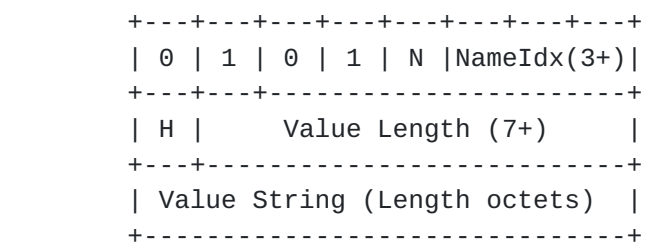
The following bit, 'N', indicates whether an intermediary is permitted to add this header to the dynamic header table on subsequent hops. When the 'N' bit is set, the encoded header MUST always be encoded with a literal representation. In particular, when a peer sends a header field that it received represented as a literal header field with the 'N' bit set, it MUST use a literal representation to forward this header field. This bit is intended

for protecting header field values that are not to be put at risk by compressing them (see Section 7.1 of [RFC7541] for more details).



Literal Header Field With Name Reference

For entries in the static table or in the dynamic table with an absolute index less than or equal to Base Index, the header field name is represented using the relative index of that entry, which is represented as an integer with a 4-bit prefix (see Section 5.1 of [RFC7541]). The "S" bit indicates whether the reference is to the static (S=1) or dynamic (S=0) table.



Literal Header Field With Post-Base Name Reference

For entries in the dynamic table with an absolute index greater than Base Index, the header field name is represented using the post-base index of that entry (see Section 3.2) encoded as an integer with a 3-bit prefix.

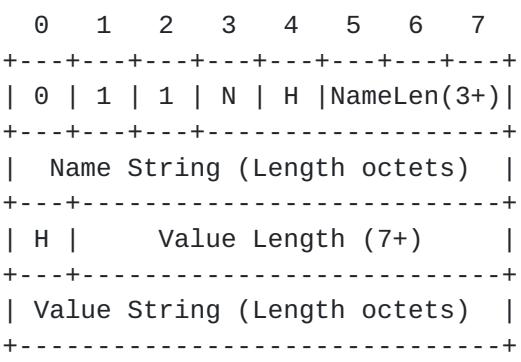
3.5.2.3. Literal Header Field Without Name Reference

An addition to the header table where both the header field name and the header field value are represented as string literals (see Section 3.1) starts with the '011' three-bit pattern.

The fourth bit, 'N', indicates whether an intermediary is permitted to add this header to the dynamic header table on subsequent hops. When the 'N' bit is set, the encoded header MUST always be encoded with a literal representation. In particular, when a peer sends a header field that it received represented as a literal header field

with the 'N' bit set, it MUST use a literal representation to forward this header field. This bit is intended for protecting header field values that are not to be put at risk by compressing them (see Section 7.1 of [RFC7541] for more details).

The name is represented as a 4-bit prefix string literal, while the value is represented as an 8-bit prefix string literal.



Literal Header Field Without Name Reference

4. Encoding Strategies

4.1. Single pass encoding

An encoder making a single pass over a list of headers must choose Base Index before knowing Largest Reference. When trying to reference a header inserted to the table after encoding has begun, the entry is encoded with different instructions that tell the decoder to use an absolute index greater than the Base Index.

4.2. Preventing Eviction Races

Due to out-of-order arrival, QPACK's eviction algorithm requires changes (relative to HPACK) to avoid the possibility that an indexed representation is decoded after the referenced entry has already been evicted. QPACK employs a two-phase eviction algorithm, in which the encoder will not evict entries that have outstanding (unacknowledged) references.

4.3. Reference Tracking

An encoder MUST ensure that a header block which references a dynamic table entry is not received by the decoder after the referenced entry has already been evicted. An encoder also respects the limit set by the decoder on the number of streams that are allowed to become blocked. Even if the decoder is willing to tolerate blocked streams, the encoder might choose to avoid them in certain cases.

In order to enable this, the encoder will need to track outstanding (unacknowledged) header blocks and table updates using feedback received from the decoder.

4.3.1. Blocked Eviction

The encoder MUST NOT permit an entry to be evicted while a reference to that entry remains unacknowledged. If a new header to be inserted into the dynamic table would cause the eviction of such an entry, the encoder MUST NOT emit the insert instruction until the reference has been processed by the decoder and acknowledged.

The encoder can emit a literal representation for the new header in order to avoid encoding delays, and MAY insert the header into the table later if desired.

To ensure that the blocked eviction case is rare, references to the oldest entries in the dynamic table SHOULD be avoided. When one of the oldest entries in the table is still actively used for references, the encoder SHOULD emit an Duplicate representation instead (see Section 3.3.3).

4.3.2. Blocked Decoding

For header blocks encoded in non-blocking mode, the encoder needs to forego indexed representations that refer to table updates which have not yet been acknowledged with Section 3.4. Since all table updates are processed in sequence on the control stream, an index into the dynamic table is sufficient to track which entries have been acknowledged.

To track blocked streams, the necessary Base Index value for each stream can be used. Whenever the decoder processes a table update, it can begin decoding any blocked streams that now have their dependencies satisfied.

4.4. Speculative table updates

Implementations can _speculatively_ send header frames on the HTTP Control Streams which are not needed for any current HTTP request or response. Such headers could be used strategically to improve performance. For instance, the encoder might decide to _refresh_ by sending Duplicate representations for popular header fields (Section 3.3.3), ensuring they have small indices and hence minimal size on the wire.

4.5. Sample One Pass Encoding Algorithm

Pseudo-code for single pass encoding, excluding handling of duplicates, non-blocking mode, and reference tracking.

```
baseIndex = dynamicTable.baseIndex
largestReference = 0
for header in headers:
  staticIdx = staticTable.getIndex(header)
  if staticIdx:
    encodeIndexReference(streamBuffer, staticIdx)
    continue

  dynamicIdx = dynamicTable.getIndex(header)
  if !dynamicIdx:
    # No matching entry.  Either insert+index or encode literal
    nameIdx = getNameIndex(header)
    if shouldIndex(header) and dynamicTable.canIndex(header):
      encodeLiteralWithIncrementalIndex(controlBuffer, nameIdx,
                                        header)
      dynamicTable.add(header)
      dynamicIdx = dynamicTable.baseIndex

  if !dynamicIdx:
    # Couldn't index it, literal
    if nameIdx <= staticTable.size:
      encodeLiteral(streamBuffer, nameIndex, header)
    else:
      # encode literal, possibly with nameIdx above baseIndex
      encodeDynamicLiteral(streamBuffer, nameIndex, baseIndex,
                           header)
      largestReference = max(largestReference,
                             dynamicTable.toAbsolute(nameIdx))
  else:
    # Dynamic index reference
    assert(dynamicIdx)
    largestReference = max(largestReference, dynamicIdx)
    # Encode dynamicIdx, possibly with dynamicIdx above baseIndex
    encodeDynamicIndexReference(streamBuffer, dynamicIdx,
                                baseIndex)

# encode the prefix
encodeInteger(prefixBuffer, 0x00, largestReference, 8)
delta = largestReference - baseIndex
sign = delta > 0 ? 0x80 : 0
encodeInteger(prefixBuffer, sign, delta, 7)

return controlBuffer, prefixBuffer + streamBuffer
```

5. Security Considerations

TBD.

6. IANA Considerations

None.

7. References

7.1. Normative References

[QUIC-HTTP] Bishop, M., "Hypertext Transfer Protocol (HTTP) over QUIC", draft-ietf-quic-http-12 (work in progress), April 2018.

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

[RFC7541] Peon, R. and H. Ruellan, "HPACK: Header Compression for HTTP/2", RFC 7541, DOI 10.17487/RFC7541, May 2015, <https://www.rfc-editor.org/info/rfc7541>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[QUIC-TRANSPORT] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-11 (work in progress), April 2018.

[RFC2360] Scott, G., "Guide for Internet Standards Writers", BCP 22, RFC 2360, DOI 10.17487/RFC2360, June 1998, <https://www.rfc-editor.org/info/rfc2360>.

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

7.3. URIs

[1] https://mailarchive.ietf.org/arch/search/?email_list=quic

[2] https://github.com/quicwg

[3] https://github.com/quicwg/base-drafts/labels/-qpack

Acknowledgments

This draft draws heavily on the text of [RFC7541]. The indirect input of those authors is gratefully acknowledged, as well as ideas from:

- Ryan Hamilton
- Patrick McManus
- Kazuho Oku
- Biren Roy
- Ian Swett
- Dmitri Tikhonov

Change Log

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

B.1. Since draft-ietf-quic-qcram-00

- Separate instruction sets for table updates and header blocks (#1235, #1142, #1141)
- Reworked indexing scheme (#1176, #1145, #1136, #1130, #1125, #1314)
- Added mechanisms that support one-pass encoding (#1138, #1320)
- Added a setting to control the number of blocked decoders (#238, #1140, #1143)
- Moved table updates and acknowledgments to dedicated streams (#1121, #1122, #1238)

Authors' Addresses

Charles 'Buck' Krasic
Google, Inc

Email: ckrasic@google.com

Mike Bishop
Akamai Technologies

Email: mbishop@evequefou.be

Alan Frindell (editor)
Facebook

Email: afrind@fb.com