moq
Internet-Draft
Intended status: Informational
Expires: 18 September 2026

L. Curley

17 March 2026

Compressed MP4
draft-lcurley-compressed-mp4-00

Abstract

Fragmented MP4 (fMP4) is widely used for live streaming, but the ISO Base Media File Format (ISOBMFF) box structure imposes significant per-fragment overhead. Each box header requires 8 bytes (4-byte size + 4-byte type), and payload fields use fixed-width integers (u32/u64). For low-latency streaming with single-frame fragments, this overhead can exceed the media payload itself.

This document defines a compression scheme for ISO BMFF that replaces box headers with compact varint-encoded identifiers and sizes, and defines compressed variants of commonly used boxes with varint payload fields. The scheme reduces per-fragment overhead from ~100 bytes to ~20 bytes while preserving the full box hierarchy.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Media Over QUIC Working Group mailing list (moq@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/moq/.

Source for this draft and an issue tracker can be found at https://github.com/kixelated/moq-drafts.

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/.

Copyright Notice

Table of Contents

1. 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.

2. Introduction

Fragmented MP4 (fMP4) is the dominant container format for low-latency live streaming. Each fragment consists of a moof (movie fragment) box followed by an mdat (media data) box. The moof box contains metadata describing the media samples in mdat.

For a typical single-frame video fragment, the overhead looks like this:

Box	Header	Payload	Total
moof	8	0	8
mfhd	8	4	12
traf	8	0	8
tfhd	8	8	16
tfdt	8	12	20
trun	8	16	24
mdat	8	0	8
Total	*56*	*40*	*96*

Table 1

This 96 bytes of overhead is substantial when a single encoded video frame might only be 100-500 bytes at low bitrates or high frame rates. Audio frames are even smaller, often 4-20 bytes for Opus at low bitrates, making the container overhead several times larger than the payload.

This document defines two layers of compression:

1. *Header Compression*: A compression table (cmpd) maps varint IDs to box type names. All boxes after the moov use compressed headers: [varint ID][varint size] instead of [u32 size][4-char type].

2. *Payload Compression*: Compressed variants of common boxes (cmfh, cfhd, cfdt, crun) replace fixed-width payload fields with varints.

Together, these reduce the per-fragment overhead to approximately 20 bytes.

3. Variable-Length Integer Encoding

This document uses the variable-length integer encoding from QUIC [RFC9000], Section 16. The first two bits of the first byte indicate the encoding length:

2MSB	Length	Usable Bits	Range
00	1	6	0-63
01	2	14	0-16383
10	4	30	0-1073741823
11	8	62	0-4611686018427387903

Table 2

In the message formats below, fields marked with (i) use this variable-length integer encoding.

4. Compression Table (cmpd)

The cmpd box is a standard ISO BMFF box placed inside the moov box. It defines a mapping from compact varint IDs to 4-character box type names.

```
cmpd {
  Count (i),
  Compressed Box Entry (..) ...,
}

Compressed Box Entry {
  ID (i),
  Name (32),
}
```

Count: The number of entries in the compression table.

ID: A varint identifier assigned to this box type. IDs SHOULD be assigned starting from 0 to minimize encoding size.

Name: The 4-character ISO BMFF box type name (e.g., moof, mdat, traf).

The cmpd box itself uses a standard ISO BMFF header since it appears inside the moov before compressed encoding takes effect.

A typical compression table for live video streaming:

ID	Name	Description
0	moof	Movie Fragment
1	mdat	Media Data
2	mfhd	Movie Fragment Header
3	traf	Track Fragment
4	tfhd	Track Fragment Header
5	tfdt	Track Fragment Decode Time
6	trun	Track Run

Table 3

With 7 entries using IDs 0-6, each ID fits in a single varint byte. Header compression alone reduces the 56 bytes of box headers (7 boxes x 8 bytes) to 14 bytes (7 boxes x 2 bytes), saving 42 bytes per fragment.

5. Compressed Box Header

The presence of a cmpd box in the moov signals that all top-level boxes following the moov use compressed box headers.

A standard ISO BMFF box header is:

```
Standard Box Header {
  Size (32),
  Type (32),
}
```

This is replaced with:

```
Compressed Box Header {
  ID (i),
  Size (i),
}
```

ID: The varint identifier from the compression table. The receiver looks up the corresponding 4-character box type name in the cmpd table.

Size: A varint containing the size of the box payload in bytes. Unlike standard ISO BMFF where the size field includes the header itself, the compressed size field contains only the payload length. This avoids the need for extended size fields since varints natively handle large values.

The box hierarchy (nesting) is preserved exactly as in standard ISO BMFF. Container boxes (e.g., moof, traf) contain nested boxes whose sizes sum to the parent's payload size. The receiver MUST be able to reconstruct the original uncompressed ISO BMFF structure by reversing the ID-to-name mapping and adjusting size fields.

6. Compressed Box Variants

This section defines compressed variants of commonly used fMP4 boxes. These variants replace fixed-width integer fields with varints, further reducing overhead.

An encoder MAY use the standard box (with a compressed header) OR the compressed variant for any given box. The compression table determines which box type is used.

6.1. cmfh — Compressed Movie Fragment Header

Replaces mfhd (Movie Fragment Header).

```
cmfh {
  Sequence Number (i),
}
```

Sequence Number: The fragment sequence number (varint instead of u32).

Standard mfhd uses 4 bytes for the sequence number. With cmfh, a sequence number under 64 requires only 1 byte.

6.2. cfhd — Compressed Track Fragment Header

Replaces tfhd (Track Fragment Header).

```
cfhd {
  Track ID (i),
  Flags (i),
  Base Data Offset (i),              ; present if flags & 0x01
  Sample Description Index (i),      ; present if flags & 0x02
  Default Sample Duration (i),       ; present if flags & 0x08
  Default Sample Size (i),           ; present if flags & 0x10
  Default Sample Flags (i),          ; present if flags & 0x20
}
```

Track ID: Identifies the track (varint instead of u32).

Flags: A varint encoding the optional field presence flags. The flag values match the standard tfhd tf_flags semantics but are renumbered for compact varint encoding:

Flag	Field
0x01	base-data-offset-present
0x02	sample-description-index-present
0x08	default-sample-duration-present
0x10	default-sample-size-present
0x20	default-sample-flags-present

Table 4

Standard tfhd uses 4 bytes for version/flags and 4 bytes for track ID (minimum 8 bytes). With cfhd, a single-track stream with no optional fields requires as few as 2 bytes.

6.3. cfdt — Compressed Track Fragment Decode Time

Replaces tfdt (Track Fragment Decode Time).

```
cfdt {
  Base Decode Time (i),
}
```

Base Decode Time: The decode timestamp of the first sample in this fragment (varint instead of u32/u64).

Standard tfdt uses 4 bytes for version/flags plus 4 or 8 bytes for the timestamp (8-12 bytes total). With cfdt, small timestamps require as few as 1 byte.

6.4. crun — Compressed Track Run

Replaces trun (Track Run).

```
crun {
  Sample Count (i),
  Flags (i),
  Data Offset (i),                  ; present if flags & 0x01
  First Sample Flags (i),           ; present if flags & 0x04
  Per-Sample Fields (..) ...,
}

Per-Sample Fields {
  Sample Duration (i),              ; present if flags & 0x100
  Sample Size (i),                  ; present if flags & 0x200
  Sample Flags (i),                 ; present if flags & 0x400
  Sample Composition Time Offset (i),; present if flags & 0x800
}
```

Sample Count: The number of samples in this run (varint instead of u32).

Flags: A varint encoding which optional fields are present. The flag values match the standard trun tr_flags semantics:

Flag	Field
0x001	data-offset-present
0x004	first-sample-flags-present
0x100	sample-duration-present
0x200	sample-size-present
0x400	sample-flags-present
0x800	sample-composition-time-offset-present

Table 5

Standard trun uses 4 bytes for version/flags, 4 bytes for sample count, and 4 bytes per optional field. With crun, a single-sample run with only sample-size typically requires 4-5 bytes instead of 16.

7. Example

This section provides a concrete byte-level comparison for a single-frame video fragment.

7.1. Standard fMP4

A typical single-frame fragment with sequence number 42, track ID 1, decode time 3840, and a 200-byte sample:

```
moof (size=80)                          8 bytes
  mfhd (size=16)                        8 bytes
    version=0, flags=0                  4 bytes
    sequence_number=42                  4 bytes
  traf (size=56)                        8 bytes
    tfhd (size=16)                      8 bytes
      version=0, flags=0x020000         4 bytes
      track_id=1                        4 bytes
    tfdt (size=20)                      8 bytes
      version=1, flags=0                4 bytes
      base_decode_time=3840             8 bytes
    trun (size=20)                      8 bytes
      version=0, flags=0x000200         4 bytes
      sample_count=1                    4 bytes
      sample_size=200                   4 bytes
mdat (size=208)                         8 bytes
  <200 bytes of media data>
```

Total overhead: 96 bytes (excluding media data).

7.2. Compressed fMP4

The same fragment using compressed encoding, with the compression table from the example in Section 4:

```
moof (id=0, size=11)                    2 bytes
  cmfh (id=2, size=1)                   2 bytes
    sequence_number=42                  1 byte
  traf (id=3, size=6)                   2 bytes
    cfhd (id=4, size=1)                 2 bytes
      track_id=1                        1 byte
      flags=0                           0 bytes (no optional fields)
    cfdt (id=5, size=2)                 2 bytes
      base_decode_time=3840             2 bytes
    crun (id=6, size=3)                 2 bytes
      sample_count=1                    1 byte
      flags=0x200                       2 bytes
      sample_size=200                   2 bytes
mdat (id=1, size=200)                   2 bytes
  <200 bytes of media data>
```

Total overhead: ~21 bytes (excluding media data).

This represents a *78% reduction* in per-fragment overhead (from 96 bytes to ~21 bytes).

8. Security Considerations

TODO Security

9. IANA Considerations

This document registers the following ISO BMFF box types:

Box Type	Description
cmpd	Compression Table
cmfh	Compressed Movie Fragment Header
cfhd	Compressed Track Fragment Header
cfdt	Compressed Track Fragment Decode Time
crun	Compressed Track Run

Table 6

10. References

10.1. Normative References

[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/rfc/rfc2119>.

[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/rfc/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

10.2. Informative References

[ISOBMFF] "Information technology — Coding of audio-visual objects — Part 12: ISO base media file format", 2022, <https://www.iso.org/standard/83102.html>.

Acknowledgments

This draft was generated with the assistance of AI (Claude).

Author’s Address

Luke Curley
Email: kixelated@gmail.com