Media Over QUIC
Internet-Draft
Intended status: Informational
Expires: 23 April 2026

J. Cenzano-Ferret
A. Frindell
Meta

20 October 2025

MoQ Media Interop
draft-cenzano-moq-media-interop-03

Abstract

This protocol can be used to send and receive video and audio over Media over QUIC Transport [MOQT], using LOC[loc] packaging.

About This Document

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

The latest revision of this draft can be found at https://afrind.github.io/draft-cenzano-media-interop/draft-cenzano-moq-media-interop.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-cenzano-moq-media-interop/.

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

Source for this draft and an issue tracker can be found at https://github.com/afrind/draft-cenzano-media-interop.

Status of This Memo

Table of Contents

1. Introduction

This protocol specifies a simple mechanism for sending media (video and audio) over LOC[loc] for both live-streaming and video conference (VC) style use cases.

moq-mi allows updating encoding parameters in the middle of a track (ex: frame rate, resolution, codec, etc)

The protocol refers to [loc] to define the specific media wire format.

2. Protocol Operation

2.1. Track Names

The publisher selects a namespace of their choosing, and sends an ANNOUNCE message for this namespace.

Within the publisher namespace the publisher will offer media tracks named as videoX and audioX where X will be an integer starting at 0.

So in case the publisher issues 2 audio tracks and 1 video track, the track names available will be video0, audio0, and audio1.

The subscriber will consider all of those tracks belonging to the same namespace as part of the same synchronization group (timestamps aligned to the same timeline).

2.2. Mapping Tracks to MoQT Object Model

For the video track, the publisher begins a new group at the start of each IDR (so object 0 will be always an IDR Keyframe), and each group contains a single subgroup. Each object has the format described in Section 2.4.

For the audio track, the publisher begins a new group with each audio object, and each group contains a single subgroup. Each object has the format described in Section 2.4.

TODO: Datagram forwarding preference could be used, but has problems if audio frame does not fit in a single UDP payload.

2.3. Timestamps

To avoid using fractional numbers and having to deal with rounding errors, timestamps will be expressed with two integers: - timestamp numerator (ex: PTS, DTS, duration) - timebase

To convert a timestamp into seconds you just need to: timestamp(s) = timestamp numerator / timebase

Example:

PTS = 11, timebase = 30

PTS(s) = 11/30 = 0.366666s

2.4. Object Format

MoQ-MI uses MOQT extension headers to provide metadata that identifies and augemts the media information found in the object payload.

2.4.1. Media type header extension (header extension type = 0x0A)

It defines the media type inside object payload (see section IANA in MOQ TODO), and it MUST be present in all objects

Value	Media type
0x0	Video H264 in AVCC
0x1	Audio Opus bitstream
0x2	UTF-8 text
0x3	Audio AAC-LC in MPEG4

Table 1

2.4.2. Video H264 in AVCC metadata header extension (header extension type = 0x15)

It provides video metadata useful to consume the video carried in the payload of the object. The following table specifies the data inside this extesion header.

```
{
  Seq ID (i)
  PTS Timestamp (i)
  DTS Timestamp (i)
  Timebase (i)
  Duration (i)
  Wallclock (i)
}
```

Figure 1: MOQT Media video h264data header extension

It MUST be present in all objects where "media type header extension" is equal to "Video H264 in AVCC"(0x0)

2.4.2.1. Seq ID

Monotonically increasing counter for this media track

2.4.2.2. PTS Timestamp

Indicates PTS in timebase

TODO: Varint does NOT accept easily negative, so it could be challenging to encode at start (priming)

2.4.2.3. DTS Timestamp

Not needed if B frames are NOT used, in that case should be same value as PTS.

TODO: Varint does NOT accept easily negative, so it could be challenging to encode at start (priming)

2.4.2.4. Timebase

Units used in PTS, DTS, and duration.

2.4.2.5. Duration

Duration in timebase. It will be 0 if not set

2.4.2.6. Wall Clock

EPOCH time in ms when this frame started being captured. It will be 0 if not set

2.4.3. Video H264 in AVCC extradata header extension (Header extension type = 0x0D)

Provides extradata needed to start decoding the video stream

It MUST be present in all object 0 (start of group) where "media type header extension" is equal to "Video H264 in AVCC"(0x0) AND there has been an update on the encoding paramets (or very start of the stream)

```
{
  Extradata (..)
}
```

Figure 2: MOQT Media video h264 extradata header extension

2.4.3.1. Extradata

This will be AVCDecoderConfigurationRecord as described in [ISO14496-15:2019] section 5.3.3.1, with field lengthSizeMinusOne = 3 (So length = 4). If any other size length is indicated (in AVCDecoderConfigurationRecord) we should error with “Protocol violation”

2.4.4. Audio Opus bitsream data header extension (Header extension type = 0x0F)

It provides audio metadata useful to consume the audio carried in the payload of the object. Following table specifies the data inside this extesion header.

It MUST be present in all objects where "media type header extension" is equal to "Audio Opus bitsream"(0x1)

```
{
  Seq ID (i)
  PTS Timestamp (i)
  Timebase (i)
  Sample Freq (i)
  Num Channels (i)
  Duration (i)
  Wall Clock (i)
}
```

Figure 3: MOQT Media data audio Opus header extension

2.4.4.1. Seq Id

Monotonically increasing counter for this media track

2.4.4.2. PTS Timestamp

Indicates PTS in timebase

TODO: Varint does NOT accept easily negative, so it could be challenging to encode at start (priming)

2.4.4.3. Timebase

Units used in PTS, DTS, and duration

2.4.4.4. Sample Freq

Sample frequency used in the original signal (before encoding)

2.4.4.5. Num Channels

Number of channels in the original signal (before encoding)

2.4.4.6. Duration

Duration in timebase. It will be 0 if not set

2.4.4.7. Wallclock

EPOCH time in ms when this frame started being captured. It will be 0 if not set

2.4.5. UTF-8 Text header extension (Header extension type = 0x11)

```
{
  Seq ID (i)
}
```

Figure 4: MOQT UTF-8 Text header extension

2.4.5.1. Seq Id

Monotonically increasing counter for this track

2.4.6. Audio AAC-LC in MPEG4 bitstream data header extension (Header extension type = 0x13)

```
{
  Seq ID (i)
  PTS Timestamp (i)
  Timebase (i)
  Sample Freq (i)
  Num Channels (i)
  Duration (i)
  Wall Clock (i)
}
```

Figure 5: MOQT Media audio AAC-LC MPEG4 object header

2.4.6.1. Seq Id

Monotonically increasing counter for this media track

2.4.6.2. PTS Timestamp

Indicates PTS in timebase

TODO: Varint does NOT accept easily negative, so it could be challenging to encode at start (priming)

2.4.6.3. Timebase

Units used in PTS, DTS, and duration

2.4.6.4. Sample Freq

Sample frequency used in the original signal (before encoding)

2.4.6.5. Num Channels

Number of channels in the original signal (before encoding)

2.4.6.6. Duration

Duration in timebase. It will be 0 if not set

2.4.6.7. Wallclock

EPOCH time in ms when this frame started being captured. It will be 0 if not set

3. Examples

3.1. Example of first object for Video H264 in AVCC

3.1.1. MOQT Object subgroup fields in case it carries Video H264 in AVCC, 1st frame sent

```
{
  0x00 (Object ID)(i),
  0x03 (Extension Count)(i),
  0x0A (Header type: Media type header type)(i)
  0x00 (header value: Media type)(i)

  0x15 (Header type: H264 in AVCC metadata)(i)
  0x0D (Header value length)(i)
  0x00 (Header value: Seq ID)(i)
  0x00 (Header value: PTS Timestamp)(i)
  0x00 (Header value: DTS Timestamp)(i)
  0x1E (Header value: Timebase)(i)
  0x01 (Header value: Duration)(i)
  0xC0, 0x00, 0x01, 0x95, 0x45, 0x6C, 0x8B, 0xFF (Header value: Wallclock)(i)

  0x0D (Header type: H264 in AVCC extradata)(i)
  Header value length (i)
  Header value: H264 in AVCC extradata (..)

  Object Payload Length (i),
  Object Payload bytes (..),
}
```

3.1.2. MOQT Object subgroup fields in case it carries Video H264 in AVCC, 2nd or bigger frame with NO encoding settings update

```
{
  0x01 (Object ID)(i),
  0x02 (Extension Count)(i),
  0x0A (Header type: Media type header type)(i)
  0x00 (header value: Media type)(i)

  0x15 (Header type: H264 in AVCC metadata)(i)
  0x0D (Header value length)(i)
  0x01 (Header value: Seq ID)(i)
  0x00 (Header value: PTS Timestamp)(i)
  0x00 (Header value: DTS Timestamp)(i)
  0x1E (Header value: Timebase)(i)
  0x01 (Header value: Duration)(i)
  0xC0, 0x00, 0x01, 0x95, 0x45, 0x6C, 0x3B, 0xE0 (Header value: Wallclock)(i)

  Object Payload Length (i),
  Object Payload bytes (..),
}
```

3.1.3. MOQT Object DATAGRAM fields in case it carries Audio AAC-LC in MPEG4

```
{
  0x00 (Track Alias)(i),
  0x00 (Group ID)(i),
  0x00 (Object ID)(i),
  0x00 (Publisher Priority)(8),
  0x02 (Extension Count)(i),
  0x0A (Header type: Media type header type)(i)
  0x03 (header value: Media type)(i)

  0x13 (Header type: Audio AAC-LC in MPEG4)(i)
  0x15 (Header value length)(i)
  0x00 (Header value: Seq ID)(i)
  0x00 (Header value: PTS Timestamp)(i)
  0x80, 0x00, 0xBB, 0x80 (Header value: Timebase)(i)
  0x80, 0x00, 0xBB, 0x80 (Header value: Sample freq)(i)
  0x02 (Header value: Num channels)(i)
  0x44, 0x00 (Header value: Duration)(i)
  0xC0, 0x00, 0x01, 0x95, 0x45, 0x6C, 0x3B, 0xE0 (Header value: Wallclock)(i)

  Object Payload Length (i),
  Object Payload bytes (..),
}
```

4. Payloads

TODO: This sections needs to be updated with links to LOC

4.1. For object header media type = Video H264 in AVCC (0x00)

Payload MUST be H264 with bitstream AVC1 format as described in [ISO14496-15:2019] section 5.3. Using 4 bytes size field length.

4.2. For object header media type = Audio Opus bitsream (0x01)

Payload MUST be Opus packets, as described in [RFC6716] - section 3

4.3. For object header media type = UTF-8 text (0x02)

Payload MUST be text bytes in UTF-8, as described in [RFC3629]

4.4. For object header media type = Audio AAC-LC in MPEG4 (0x03)

Payload MUST be AAC frame (syntax element raw_data_block()), as described in section 4.4.2.1 of [ISO14496-3:2009].

5. References

[ISO14496-15:2019] "Carriage of network abstraction layer (NAL) unit structured video in the ISO base media file format", ISO ISO14496-15:2019, International Organization for Standardization, October, 2022.

[ISO14496-3:2009] "Information technology — Coding of audio-visual objects", ISO ISO14496-3:2009, International Organization for Standardization, September, 2009.

6. 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.

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

9. 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>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[RFC6716] Valin, JM., Vos, K., and T. Terriberry, "Definition of the Opus Audio Codec", RFC 6716, DOI 10.17487/RFC6716, September 2012, <https://www.rfc-editor.org/rfc/rfc6716>.

[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>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Jordi Cenzano-Ferret
Meta
Email: jcenzano@meta.com

Alan Frindell
Meta
Email: afrind@meta.com