Media Over QUIC
Internet-Draft
Intended status: Standards Track
Expires: 16 September 2026

M. Zanaty
S. Nandakumar
Cisco
P. Thatcher
Microsoft

15 March 2026

Low Overhead Media Container
draft-ietf-moq-loc-02

Abstract

This specification describes a Low Overhead Media Container (LOC) format for encoded and encrypted audio and video media data to be used primarily for interactive Media over QUIC Transport (MOQT). It may be used in the MOQT Streaming Format (MSF) specification, which defines a catalog format for publishers to declare and describe their LOC tracks and for subscribers to consume them. Examples are also provided for building media applications using LOC and MOQT.

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://moq-wg.github.io/loc/draft-ietf-moq-loc.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-moq-loc/.

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/moq-wg/loc.

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. Introduction

This specification describes a low-overhead media container (LOC) format for encoded and encrypted audio and video media data.

"Low-overhead" refers to minimal extra encapsulation as well as minimal application overhead when interfacing with WebCodecs [WebCodecs].

The container format description is specified for all audio and video codecs defined in the WebCodecs Codec Registry [WEBCODECS-CODEC-REGISTRY]. The audio and video payload bitstream is identical to the "internal data" inside an EncodedAudioChunk and EncodedVideoChunk, respectively, specified in the registry.

(Note: Do we need to support timed text tracks such as Web Video Text Tracks (WebVTT) ?)

In addition to the media payloads, critical metadata called properties are also specified for audio and video payloads.

A primary motivation is to align with media formats used in WebCodecs to minimize extra encapsulation and application overhead when interfacing with WebCodecs. Other container formats like CMAF or RTP would require more extensive application overhead in format conversions, as well as larger encapsultion overhead which may burden some use cases like low bitrate audio scenarios.

This specification can also be used by applications outside the context of WebCodecs or a web browser. While the media payloads are defined by referring to the "internal data" of an EncodedAudioChunk or EncodedVideoChunk in the WebCodecs Codec Registry, this "internal data" is the elementary bitstream format of codecs without any encapsulation. Referring to the WebCodecs Codec Registry avoids duplicating it in an identical IANA registry.

* Section 2 defines the core media payload formats.

* Section 2.3 defines the metadata, called properties, associated with audio and video payloads.

* Section 3 defines the usage of end-to-end encrypted LOC payloads.

* Section 4 provides examples with details for building audio and video applications using LOC over MOQ.

1.1. Requirements Notation and Conventions

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.

1.2. Terminology

Track, Group, Subgroup, Object, and their corresponding identifiers (ID or alias) are defined in [MoQTransport] and used here to refer to those aspects of the MOQT Object Model.

2. Payload Format

The WebCodecs Codec Registry defines the contents of an EncodedAudioChunk and EncodedVideoChunk for the audio and video codec formats in the registry. The "internal data" in these chunks is used directly in this specification as the "LOC Payload" bitstream. This "internal data" is the elementary bitstream format of each codec without any encapsulation.

2.1. Video Payload Format

For video formats with multiple bitstream formats in the WebCodecs Registry, such as H.264/AVC or H.265/HEVC, the LOC Payload can use either the "canonical" format ("avc" or "hevc") often used in storage containers like MP4 / ISO BMFF, or the "annexB" format used in some non-MP4 applications. These formats differ in how they carry initialization and configuration information called parameter sets as well as how parts of a video frame are delimited with length prefixes or start codes.

2.1.1. Parameter Sets In Payload

Parameter sets can be sent in the bitstream payload before key frames, similar to "annexB" formats. Newer "canonical" formats such as "avc3" and "hev1" codec strings also support parameter sets in the bitstream payload or outside it in "extradata" metadata headers.

2.1.2. Parameter Sets in Headers

Parameter sets can be sent in headers before key frames, as described in the Video Config LOC Property Section 2.3.2.1, similar to the original "canonical" formats such as "avc1" and "hvc1" codec strings. The Video Config contents are the "extradata" bytes defined by the corresponding codec specification, which map to the WebCodecs VideoDecoderConfig description property in the EncodedVideoChunkMetadata.

2.1.3. Length Prefixes in Payload

A 4-byte length prefix can be sent before each NAL Unit, similar to "canonical" ("avc" or "hevc") formats. A length value of 1 SHOULD be interpreted as a start code rather than a length. The length is in network byte order, i.e. big endian, and SHOULD be 4 bytes long to disambiguate from start code prefixes. A length prefix less than 4 bytes long, which is uncommon, MAY be specified in the Video Config Section 2.3.2.1.

2.1.4. Start Code Prefixes in Payload

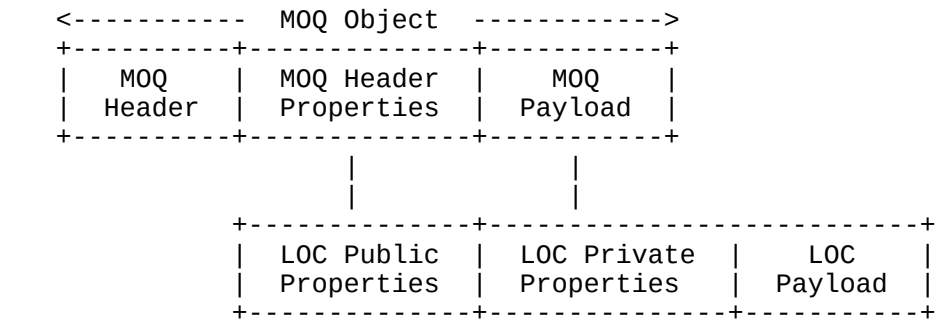
A 4-byte start code can be sent before each NAL Unit, similar to "annexB" formats. The start code value is 1 in network byte order, i.e. big endian, and SHOULD be 4 bytes long to disambiguate from length prefixes. A 3-byte start code, which is uncommon, MAY be used if the track never uses length prefixes or any Config Section 2.3.2.1.

2.2. MOQ Object Mapping

An application object when transported as a [MoQTransport] object is composed of a MOQ Object Header, with optional Properties, and a Payload. Media objects encoded using the container format defined in this specification populate the MOQ Object Properties with the LOC Public Properties, and populate the MOQ Object Payload with LOC Private Properties followed by the LOC Payload, as shown below.

The LOC Payload is the "internal data" of an EncodedAudioChunk or EncodedVideoChunk.

The LOC Public and Private Properties carry optional metadata related to the Payload, where Public Properties are visible to relays while Private Properties can be encypted end to end wi



LOC Public Properties = some MOQ Object Properties
LOC Private Properties + LOC Payload = all MOQ Object Payload
LOC Payload = "internal data" of EncodedAudio/VideoChunk

2.3. LOC Properties

The LOC Public and Private Properties carry optional metadata for the corresponding LOC Payload. The LOC Public Properties are contained within the MOQ Object Properties. This metadata provides necessary information for end subscribers, relays and other intermediaries to perform their operations without accessing the media payload. For example, media switches can use this metadata to perform their media switching decisions without accessing the payload which may be encrypted end-to-end (from original publisher to end subscribers). The LOC Private Properties are contained within the MOQ Object Payload, and are not intended to be processed by relays.

The following sections define specific metadata as LOC Public and Private Properties and register them in the IANA registry for MOQ Object Properties.

Other specifications can define other metadata as LOC Public and Private Properties and register them in the same registry. Each property must specify the following information in the IANA registry.

* Name: Short name for the metadata (not sent on the wire)

* Description: Detailed description (not sent on the wire)

* ID: Identifier assigned by the registry (vi64)

* Length: Length of metadata Value in bytes (vi64 if ID is odd, omitted if ID is even)

* Value: Value of metadata (vi64 if ID is even, Length bytes if ID is odd)

2.3.1. Common Properties

2.3.1.1. Timestamp

* Name: Timestamp

* Description: Timestamp of the encoded media frame encoded as vi64. The unit of the timestamp is determined by the Timescale property Section 2.3.1.2. If no timescale property is present, the timestamp is interpreted as wall-clock time in microseconds since the Unix epoch.

* ID: 0x06

* Length: Omitted (ID is even)

* Value: vi64 (1-8 bytes)

2.3.1.2. Timescale

* Name: Timescale

* Description: The number of Timestamp units per second, encoded as vi64. This property defines the unit for interpreting timestamp values in the Timestamp property. Common values include 1000000 for microseconds, 48000 for audio at 48kHz sample rate, 90000 for video at 90kHz clock rate. When this property is present, the Timestamp represents media time rather than wall-clock time. The epoch or anchor point for the timestamp is application-defined. If this property is not present, timestamps default to microseconds since Unix epoch.

* ID: 0x08

* Length: Omitted (ID is even)

* Value: vi64 (1-9 bytes)

2.3.2. Video Header Data

2.3.2.1. Video Config

* Name: Video Config

* Description: Video codec configuration "extradata", as defined by the corresponding codec specification, which maps to the WebCodecs VideoDecoderConfig description property in the EncodedVideoChunkMetadata.

* ID: 13 (IANA, please assign from the MOQ Properties Registry)

* Length: Varies

* Value: Varies

2.3.2.2. Video Frame Marking

* Name: Video Frame Marking

* Description: Flags for video frames which are independent, discardable, or base layer sync points, as well as temporal and spatial layer identification, as defined in [RFC9626], encoded in the least significant bits of a vi64.

* ID: 4 (IANA, please assign from the MOQ Properties Registry)

* Length: Varies (1-4 bytes)

* Value: Varies

2.3.3. Audio Header Data

2.3.3.1. Audio Level

* Name: Audio Level

* Description: The magnitude of the audio level of the corresponding audio frame as well as a voice activity indicator as defined in section 3 of [RFC6464], encoded in the least significant 8 bits of a vi64.

* ID: 6 (IANA, please assign from the MOQ Properties Registry)

* Length: Varies (1-2 bytes)

* Value: Varies

3. Payload Encryption

When end to end encryption is supported, the encoded payload is encrypted with symmetric keys derived from key establishment mechanisms, such as [MOQ-MLS], and the payload itself is protected using mechanisms defined in [SecureObjects].

3.1. Secure Objects Integration

[SecureObjects] defines a comprehensive framework for end-to-end encryption of MOQT objects. When using Secure Objects with LOC, the following considerations apply:

3.1.1. Key Identification

The Secure Object Key ID property (type 0x2 in [SecureObjects]) MUST be included as an immutable property to identify the keying material used for encryption. This property is authenticated but not encrypted, allowing relays to forward objects without decryption while ensuring subscribers can identify the correct decryption key.

3.1.2. Immutable Properties

LOC Properties that should be immutable but visible to relays SHOULD be encoded as Immutable Properties as defined in [MoQTransport] to ensure they cannot be modified by relays and are included in the authenticated associated data (AAD) during encryption. This specification does not define any Immutable Properties, but other specifications may define some for use with LOC.

3.1.3. Private Properties for Sensitive Metadata

Some LOC metadata may be sensitive and should not be visible to relays. [SecureObjects] defines a Private properties mechanism (type 0xA) that allows metadata to be encrypted alongside the payload.

The following LOC properties MAY be carried as Private properties when end-to-end confidentiality is required:

* Timestamp and Timescale - reveals timing information about the source

* Audio Level - reveals voice activity and audio characteristics

* Video Frame Marking - reveals encoding structure details

* Video Config - reveals encoding configuration details

When using Private properties:

1. The LOC property is encoded as a key-value pair within the Private properties payload

2. The Private properties are concatenated with the media payload before encryption

3. Upon decryption, the receiver extracts the Private properties and reconstructs the LOC metadata

This approach allows sensitive metadata to remain confidential from relays while still being available to authorized end subscribers.

3.1.4. Cipher Suite Requirements

Implementations using LOC with Secure Objects MUST support the AES_128_GCM_SHA256_128 cipher suite (0x0004). Other cipher suites defined in [SecureObjects] MAY be used based on application requirements for authentication tag size versus bandwidth overhead.

3.1.5. AAD Construction

The authenticated associated data (AAD) for AEAD encryption includes:

* Key ID

* Group ID and Object ID

* Track namespace and name

* Serialized immutable properties

This binding ensures objects cannot be replayed across different tracks or contexts.

4. Examples

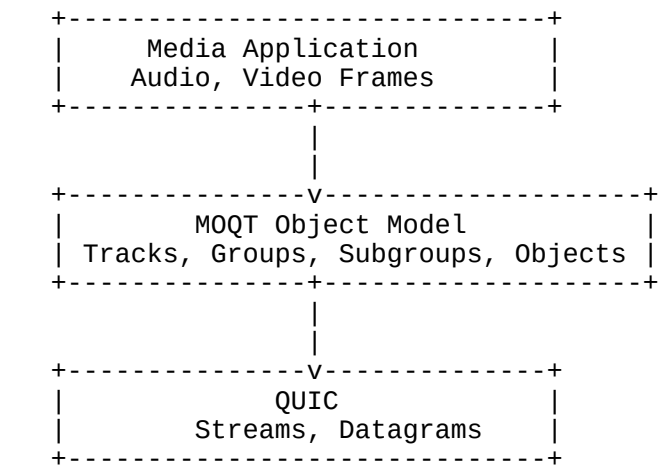
This section provides examples with details for building audio and video applications using MOQ and LOC; more specifically, it provides information on:

* Using a MSF catalog [MoQCatalog] to describe track information,

* Packaging media into LOC streaming format, and

* Mapping application media objects to the MOQT object model and transport.

The figure below shows the conceptual model for mapping media application data to the MOQT object model and underlying QUIC transport.



4.1. Application with one audio track

An example is shown below for an Opus mono channel audio track at 48Khz.

```
codec: "opus"
bitrate: 24000
samplerate: 480000
channelConfig: "mono"
lang: "en"
```

When ready for publishing, each encoded audio chunk, say 10ms, represents a MOQT Object. In this setup, there is one MOQT Object per MOQT Group, where the GroupID in the object header is increment by one for each encoded audio chunk and the ObjectID is defaulted to value 0.

These objects can be sent as QUIC streams or datagrams. When mapped to QUIC datagrams, each object must fit entirely within a QUIC datagram, and when mapped to QUIC Streams, each such unitary group is sent over an individual unidirectional QUIC stream since there is just one SubGroup per each MOQT Group.

4.2. Application with one single quality video track

An example is shown below for an H.264 video track with 1280x720p resolution and 30 fps frame rate at 1 Mbps bitrate.

codec: "avc3.42E01E"
bitrate: 1000000
framerate: 30
width: 1280
height: 720

When ready for publishing, each encoded video chunk is considered as input to MOQT Object payload. If encrypted, the output of encryption will serve as the object's payload. The GroupID is incremented by 1 at IDR Frame boundaries. The ObjectID is increment by 1 for each encoded video frame, starting at 0 and resetting to 0 at the start of a new group. The first encoded video frame, MOQT Object with ObjectID 0, shall be the Independent (IDR) frame and the rest of the encoded video frames corresponds to dependent (delta) frames, organized in the decode order.

When mapping to QUIC for sending, one unidirectional QUIC stream is setup to deliver all the encoded video chunks within a MOQT group.

When decoding at the 'End Consumer', the objects from each of the QUIC streams are fed in the GroupID then ObjectID order to the decoder for the track.

4.3. Application with single video track with temporal layers

An example is shown below for an H.264 video track with 1280x720p resolution and 2 temporal layers at 30 fps and 60 fps frame rate.

codec: "avc3.42E01F"
bitrate: 1500000
framerate: 60
width: 1280
height: 720

When ready for publishing, each encoded video chunk is considered as input to MOQT Object payload. If encrypted, the output of encryption will serve as the object's payload. The GroupID is incremented by 1 at Independent (IDR) frame boundaries. Each MOQT group shall contain 2 SubGroups corresponding to the 2 temporal layers as shown below:

Layer:0/30fps Subgroup: 0 ObjectID: even
Layer:1/60fps Subgroup: 1 ObjectID: odd

Within the MOQT group, ObjectID is increment by 1 for each encoded video frame, starting at 0 and resetting to 0 at the start of a new group. The first encoded video frame, MOQT Object with ObjectID 0, shall be the Indepedent (IDR) frame and the rest of the encoded video frames corresponds to dependent (delta) frames, organized in the decode order. When mapping to QUIC for sending, one unidirectional QUIC stream is used per SubGroup, thus resulting in 2 QUIC streams per MOQT group.

When decoding at the 'End Consumer' for a given MOQT group, the objects must be fed in the GroupID then ObjectID order. This implies that the consumer media application needs to order objects across the SubGroup QUIC streams.

4.4. Application with mutiple dependent video tracks

An example is shown below for an H.264 video track with 2 spatial qualities at 360p and 720p each at 30 fps

Video Track 1
codec: "avc3.42E01E"
bitrate: 500000
framerate: 30
width: 640
height: 360

Video Track 2
codec: "svc1.56401F"
bitrate: 1000000
framerate: 30
width: 1280
height: 720

When ready for publishing, the mapping to the MOQT object model and to underlying QUIC, follows the same procedures as described in Section 4.2 for each video track.

When decoding at the ’End Consumer’ for a given MOQT group, the objects must be fed in the GroupID then ObjectID order in the ascending quality track order.

For the example in the section, this would imply following pattern when decoding group 5.

```
Track 1 Group 5 Object 0
Track 2 Group 5 Object 0
Track 1 Group 5 Object 1
Track 2 Group 5 Object 1
....
```

4.5. Application with mutiple dependent video tracks with dyadic framerate levels.

An example is shown below for an H.264 video track with 2 spatial qualities at 360p and 720p, however, the framerate between tracks vary dyadically.

```
Video Track 1
codec: "avc3.42E01E"
bitrate: 500000
framerate: 30
width: 640
height: 360

Video Track 2
codec: "svc1.56E01F"
bitrate: 1000000
framerate: 60
width: 1280
height: 720
```

When ready for publishing, the mapping to the MOQT object model and to underlying QUIC, follows the same procedures as described in Section 4.2 for each video track.

When decoding at the ’End Consumer’ for a given MOQT group, the objects from across the tracks must be fed in the timestamp order to the decoder, if no frame reordering is present in the encoding.

If the encoding uses frame reordering, or if timestamp cannot be obtained, the object to choose next shall follow the below formula.

```
Object Decode Order = ObjectID * multiplier + offset

multiplier = 2^(maxlayer-max(0,layer-1))
offset = 2^(maxlayer-layer) MOD multiplier
```

4.6. Application with multiple simulcast qualities video tracks

An example is shown below for an H.264 video track with 2 simulcast spatial qualities at 360p and 720p each at 30 fps.

```
Video Track 1
codec: "avc3.42E01E"
bitrate: 500000
framerate: 30
width: 640
height: 360

Video Track 2
codec: "avc3.42E01F"
bitrate: 1000000
framerate: 30
width: 1280
height: 720
```

When ready for publishing, the mapping to the MOQT object model and to underlying QUIC, follows the same procedures as described in Section 4.2 for each video track.

When decoding at the ’End Consumer’, the objects from the QUIC stream are fed in the GroupID then ObjectID order to the decoders setup for the corresponding video tracks.

5. Security and Privacy Considerations

The metadata in LOC Properties is visible to relays, since the MOQ Object Properties are often not encrypted end-to-end (from original publisher to end subscribers) in common schemes. In some cases, this may be an intentional design intent for proper relay operation. In other cases, this may be unintentional or undesirable leaking of the metadata to relays. Each metadata that is defined should consider the security and privacy aspects of granting relays visibility to the metadata.

5.1. Protecting Sensitive Metadata

The metadata defined and registered in this specification (Timestamp, Timescale, Config, Video Frame Marking, and Audio Level) may be sensitive metadata that should be encrypted end-to-end. Applications requiring confidentiality of this metadata SHOULD use the Private properties mechanism described in Section 3.1.3 to encrypt sensitive metadata alongside the payload.

When using [SecureObjects] for end-to-end encryption:

* Sensitive metadata (timestamps, audio levels, frame marking) can be encrypted using Private properties

* Immutable properties are authenticated but visible to relays

* Media switches that need metadata access require appropriate key material

5.2. Immutable Property Considerations

Properties marked as immutable via the IMMUTABLE_PROPERTIES mechanism in [MoQTransport] provide integrity protection - relays cannot modify these values without detection. However, immutable properties are NOT encrypted and remain visible to relays. Applications must carefully consider which properties require:

* Confidentiality (use Private properties)

* Integrity without confidentiality (use Immutable properties)

* Neither (standard mutable properties suitable for relay operation)

5.3. Relay Trust Model

Different deployment scenarios have different trust models for relays:

* Untrusted relays: Use Private properties for all sensitive metadata

* Semi-trusted relays (media switches): May have access to metadata keys but not payload keys, enabling switching decisions without content access

* Trusted relays: May have full key access for transcoding or processing

Applications should select the appropriate protection mechanisms based on their relay trust model and privacy requirements.

5.4. Deletion Detection

When using end-to-end encryption, relays cannot modify encrypted payloads but could selectively delete or withhold objects. [MoQTransport] defines PRIOR_GROUP_ID_GAP and PRIOR_OBJECT_ID_GAP properties that publishers can include to indicate intentional gaps in sequences. Subscribers can use these to distinguish between publisher-intended gaps and potential relay deletion. [SecureObjects] provides additional mechanisms for detecting such attacks.

6. IANA Considerations

6.1. MOQ Properties Registry

This document registers the following entries in the "MOQ property Headers" registry established by [MoQTransport]:

Type	Name	Scope	Specification
0x06	TIMESTAMP	Object	Section 2.3.1.1
0x08	TIMESCALE	Track or Object	Section 2.3.1.2

Table 1

6.2. MoQ Streaming Format Registry

This document creates a new entry in the "MoQ Streaming Format" Registry (see [MoQTransport] Sect 8). The type value is 0x002, the name is "LOC Streaming Format" and the RFC is XXX.

7. References

7.1. Normative References

[MoQTransport]
Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-17, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-17>.

[WebCodecs]
"WebCodecs", July 2023,
<https://www.w3.org/TR/webcodecs/>.

[WEBCODECS-CODEC-REGISTRY]
"WebCodecs Codec Registry", July 2023,
<https://www.w3.org/TR/webcodecs-codec-registry/>.

[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>.

[RFC9626] Zanaty, M., Berger, E., and S. Nandakumar, "Video Frame Marking RTP Header Extension", RFC 9626, DOI 10.17487/RFC9626, March 2025, <https://www.rfc-editor.org/rfc/rfc9626>.

[RFC6464] Lennox, J., Ed., Ivov, E., and E. Marocco, "A Real-time Transport Protocol (RTP) Header Extension for Client-to-Mixer Audio Level Indication", RFC 6464, DOI 10.17487/RFC6464, December 2011, <https://www.rfc-editor.org/rfc/rfc6464>.

7.2. Informative References

[MoQCatalog]
Law, W., "MOQT Streaming Format", Work in Progress, Internet-Draft, draft-ietf-moq-msf-00, 19 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-msf-00>.

[SecureObjects]
Jennings, C. F., Nandakumar, S., and R. Barnes, "End-to-End Secure Objects for Media over QUIC Transport", Work in Progress, Internet-Draft, draft-jennings-moq-secure-objects-04, 8 February 2026, <https://datatracker.ietf.org/doc/html/draft-jennings-moq-secure-objects-04>.

[MOQ-MLS] Jennings, C. F., Nandakumar, S., and R. Barnes, "End-to-end Security for Media over QUIC", Work in Progress, Internet-Draft, draft-jennings-moq-e2ee-mls-03, 30 June 2025, <https://datatracker.ietf.org/doc/html/draft-jennings-moq-e2ee-mls-03>.

Appendix A. Acknowledgements

Thanks to Cullen Jennings for suggestions and review.

Authors' Addresses

Mo Zanaty
Cisco
Email: mzanaty@cisco.com

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Peter Thatcher
Microsoft
Email: pthatcher@microsoft.com