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

W. Law
Akamai

19 January 2026

MOQT Streaming Format
draft-ietf-moq-msf-00

Abstract

This document specifies the MOQT Streaming Format, designed to operate on Media Over QUIC Transport.

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

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/msf.

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 23 July 2026.

Table of Contents

1. Introduction 4
2. Conventions and Definitions 4
3. Scope 4
4. Media packaging 6
4.1. LOC packaging 6
4.2. Time-alignment 6
4.3. Content protection and encryption 6
5. Catalog 6
5.1. Catalog Fields 7
5.1.1. MSF version 9
5.1.2. Delta update 9
5.1.3. Add tracks 9
5.1.4. Remove tracks 9
5.1.5. Clone tracks 10
5.1.6. Generated at 10
5.1.7. Is Complete 10
5.1.8. Tracks 10
5.1.9. Tracks object 10
5.1.10. Track namespace 11
5.1.11. Track name 11
5.1.12. Packaging 11
5.1.13. Event timeline type 11
5.1.14. Track role 12
5.1.15. Is Live 13
5.1.16. Target latency 13
5.1.17. Track label 13
5.1.18. Render group 13
5.1.19. Alternate group 14
5.1.20. Initialization data 14
5.1.21. Dependencies 14
5.1.22. Temporal ID 14
5.1.23. Spatial ID 14

5.1.24. Codec 15
5.1.25. Mimetype 15
5.1.26. Framerate 15
5.1.27. Timescale 15
5.1.28. Bitrate 15
5.1.29. Width 15
5.1.30. Height 15
5.1.31. Audio sample rate 16
5.1.32. Channel configuration 16
5.1.33. Display width 16
5.1.34. Display height 16
5.1.35. Language 16
5.1.36. Parent name 16
5.1.37. Track duration 17
5.2. Delta updates 17
5.3. Catalog Examples 18
5.3.1. Time-aligned Audio/Video Tracks with single quality 18
5.3.2. Simulcast video tracks - 3 alternate qualities along with audio 19
5.3.3. SVC video tracks with 2 spatial and 2 temporal qualities 21
5.3.4. Delta update - adding two tracks 23
5.3.5. Delta update removing tracks 24
5.3.6. Time-aligned Audio/Video Tracks with custom field values 24
5.3.7. Time-aligned VOD Audio/Video Tracks 25
5.3.8. Media timeline and Event timeline 26
5.3.9. Terminating a live broadcast 27
6. Media transmission 28
6.1. Group numbering 28
7. Media Timeline track 28
7.1. Media Timeline track payload 28
7.2. Media Timeline Catalog requirements 29
7.3. Media Timeline track updating 29
8. Event Timeline track 29
8.1. Event Timeline data format 30
8.2. Event Timeline Catalog requirements 30
8.3. Event Timeline track updating 31
8.4. Event timeline track examples 31
8.4.1. Event timeline track with wallclock time indexing . . 31
8.4.2. Event timeline track with MOQT Location indexing . . 31
9. Workflow 32
9.1. Initiating a broadcast 32
9.2. Ending a live broadcast 32
10. Security Considerations 32
11. IANA Considerations 32
12. Normative References 33

Acknowledgments 34
Contributors 34
Author's Address 34

1. Introduction

MOQT Streaming Format (MSF) is a media format designed to deliver LOC [LOC] compliant media content over Media Over QUIC Transport (MOQT) [MoQTransport]. MSF works by fragmenting the bitstream into objects that can be independently transmitted. MSF leverages a catalog format to describe the output of the original publisher. MSF specifies how content should be packaged and signaled, defines how the catalog communicates the content, specifies prioritization strategies for real-time and workflows for beginning and terminating broadcasts. MSF also details how end-subscribers may perform adaptive bitrate switching. MSF is targeted at real-time and interactive levels of live latency, as well as VOD content.

This document describes version 1 of the streaming format.

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.

This document uses the conventions detailed in Section 1.3 of [RFC9000] when describing the binary encoding.

3. Scope

The purpose of MSF is to provide an interoperable media streaming format operating over [MoQTransport]. Interoperability implies that:

* An original publisher can package incoming media content into tracks, prepare a catalog and announce the availability of the content to an MOQT relay. Media content refers to audio and video data, as well as ancillary data such as captions, subtitles, accessibility and other timed-text data.

* An MOQT relay can process the announcement as well as cache and propagate the tracks, both to other relays or to the final subscriber.

* A final subscriber can parse the catalog, request tracks, decode and render the received media data.

MSF is intended to provide a format for delivering commercial media content. To that end, the following features are within scope:

* Video codecs - all codecs supported by [LOC]

* Audio codecs - all audio codecs supported by [LOC]

* Catalog track - describes the availability and characteristics of content produced by the original publisher.

* Timeline track - describes the relationship between MOQT Group and Object IDs to media time.

* Token-based authorization and access control

* Captions + Subtitles - support for [WEBVTT] and [IMSC1] transmission

* Latency support across multiple regimes (thresholds are informative only and describe the delay between the original publisher placing the content on the wire and the final subscriber rendering it)

* Real-time - less than 500ms

* Interactive - between 500ms and 2500ms

* Standard - above 2500ms

* VOD latency - content that was previously produced, is no longer live and is available indefinitely.

* Content encryption

* ABR between time-synced tracks - subscribers may switch between tracks at different quality levels in order to maximize visual or audio quality under conditions of throughput variability.

* Capable of delivering interstitial advertising.

* Logs and analytics management - support for the reporting of client-side QoE and relay delivery actions.

Initial versions of MSF will prioritize basic features necessary to exercise interoperability across delivery systems. Later versions will add commercially necessary features.

4. Media packaging

MSF delivers LOC [LOC] packaged media bitstreams.

4.1. LOC packaging

This specification references Low Overhead Container (LOC) [LOC] to define how audio and video content is packaged. With this packaging mode, each EncodedAudioChunk or EncodedVideoChunk sample is placed in a separate MOQT Object. Samples that belong to the same Group of Pictures (GOP) MUST be placed within the same MOQT Group.

When LOC packaging is used for a track, the catalog packaging attribute (Section 5.1.12) MUST be present and it MUST be populated with a value of "loc".

4.2. Time-alignment

MSF Tracks MAY be time-aligned. Those that are, are subject to the following requirements:

* Tracks advertised in the catalog as belonging to a common render group MUST be time-aligned.

* The render duration of the first media object of each equally numbered MOQT Group, after decoding, MUST have overlapping presentation time.

A consequence of this restriction is that an MSF receiver SHOULD be able to cleanly switch between time-aligned media tracks at group boundaries.

4.3. Content protection and encryption

ToDo - content protection for LOC-packaged content.

5. Catalog

A Catalog is an MOQT Track that provides information about the other tracks being produced by a MSF publisher. A Catalog is used by MSF publishers for advertising their output and for subscribers in consuming that output. The payload of the Catalog object is opaque to Relays and can be end-to-end encrypted. The Catalog provides the names and namespaces of the tracks being produced, along with the relationship between tracks, properties of the tracks that consumers may use for selection and any relevant initialization data.

The catalog track MUST have a case-sensitive Track Name of "catalog".

A catalog object MAY be independent of other catalog objects or it MAY represent a delta update of a prior catalog object. The first catalog object published within a new group MUST be independent. A catalog object SHOULD be published only when the availability of tracks changes.

Each catalog update MUST be mapped to an MOQT Object.

5.1. Catalog Fields

A catalog is a JSON [JSON] document, comprised of a series of mandatory and optional fields. At a minimum, a catalog MUST provide all mandatory fields. A producer MAY add additional fields to the ones described in this draft. Custom field names MUST NOT collide with field names described in this draft. The order of field names within the JSON document is not important.

A parser MUST ignore fields it does not understand.

Table 1 provides an overview of all fields defined by this document.

Field	Name	Definition
MSF version	version	Section 5.1.1
Delta update	deltaUpdate	Section 5.1.2
Add tracks	addTracks	Section 5.1.3
Remove tracks	removeTracks	Section 5.1.4
Clone tracks	cloneTracks	Section 5.1.5
Generated at	generatedAt	Section 5.1.6
Is Complete	isComplete	Section 5.1.7
Tracks	tracks	Section 5.1.8
Track namespace	namespace	Section 5.1.10
Track name	name	Section 5.1.11
Packaging	packaging	Section 5.1.12
Event timeline type	eventType	Section 5.1.13

Is Live	isLive	Section 5.1.15
Target latency	targetLatency	Section 5.1.16
Track role	role	Section 5.1.14
Track label	label	Section 5.1.17
Render group	renderGroup	Section 5.1.18
Alternate group	altGroup	Section 5.1.19
Initialization data	initData	Section 5.1.20
Dependencies	depends	Section 5.1.21
Temporal ID	temporalId	Section 5.1.22
Spatial ID	spatialId	Section 5.1.23
Codec	codec	Section 5.1.24
Mime type	mimeType	Section 5.1.25
Framerate	framerate	Section 5.1.26
Timescale	timescale	Section 5.1.27
Bitrate	bitrate	Section 5.1.28
Width	width	Section 5.1.29
Height	height	Section 5.1.30
Audio sample rate	samplerate	Section 5.1.31
Channel configuration	channelConfig	Section 5.1.32
Display width	displayWidth	Section 5.1.33
Display height	displayHeight	Section 5.1.34
Language	lang	Section 5.1.35
Parent name	parentName	Section 5.1.36
Track duration	trackDuration	Section 5.1.37

Table 1

Table 2 defines the allowed locations for these fields within the document

Location	Allowed locations for the field
R	The Root of the JSON object
T	Track object

Table 2

5.1.1. MSF version

Location: R Required: Yes JSON Type: Number

Specifies the version of MSF referenced by this catalog. There is no guarantee that future catalog versions are backwards compatible and field definitions and interpretation may change between versions. A subscriber MUST NOT attempt to parse a catalog version which it does not understand.

5.1.2. Delta update

Location: R Required: Optional JSON Type: Boolean

A Boolean that if true indicates that this catalog object represents a delta (or partial) update. A delta update has a restricted set of fields and special processing rules - see Section 5.2. This value SHOULD NOT be added to a catalog if it is false.

5.1.3. Add tracks

Location: R Required: Optional JSON Type: Array

Indicates a delta processing instruction to add new tracks. The value of this field is an Array of track objects Section 5.1.9.

5.1.4. Remove tracks

Location: R Required: Optional JSON Type: Array

Indicates a delta processing instruction to remove new tracks. The value of this field is an Array of track objects Section 5.1.9. Each track object MUST include a Track Name Section 5.1.11 field, MAY include a Track Namespace Section 5.1.10 field and MUST NOT hold any other fields.

5.1.5. Clone tracks

Location: R Required: Optional JSON Type: Array

Indicates a delta processing instruction to clone new tracks from previously declared tracks. The value of this field is an Array of track objects Section 5.1.9. Each track object MUST include a Parent Name Section 5.1.36 field.

5.1.6. Generated at

Location: R Required: Optional JSON Type: Number

The wallclock time at which this catalog instance was generated, expressed as the number of milliseconds that have elapsed since January 1, 1970 (midnight UTC/GMT). This field SHOULD NOT be included if the isLive field is false.

5.1.7. Is Complete

Location: R Required: Optional JSON Type: Boolean

Issued once a previously live broadcast is complete. This is a commitment that all tracks are complete, no new tracks will be added and no new content will be published. This field MUST NOT be included if it is FALSE. This field MUST NOT be removed from a catalog once it has been added.

5.1.8. Tracks

Location: R Required: Yes JSON Type: Array

An array of track objects Section 5.1.9.

5.1.9. Tracks object

A track object is JSON Object containing a collection of fields whose location is specified ’T’ in Table 2.

5.1.10. Track namespace

Location: T Required: Optional JSON Type: String

The name space under which the track name is defined. See section 2.3 of [MoQTransport]. The track namespace is optional. If it is not declared within a track, then each track MUST inherit the namespace of the catalog track. A namespace declared in a track object overwrites any inherited name space.

5.1.11. Track name

Location: T Required: Yes JSON Type: String

A string defining the name of the track. See section 2.3 of [MoQTransport]. Within the catalog, track names MUST be unique per namespace.

5.1.12. Packaging

Location: T Required: Yes JSON Type: String

A string defining the type of payload encapsulation. Allowed values are strings as defined in Table 3.

Table 3: Allowed packaging values

Name	Value	Reference
LOC	loc	See RFC XXXX
Media Timeline	mediatimeline	See Section 7
Event Timeline	eventtimeline	See Section 8

Table 3

5.1.13. Event timeline type

Location: T Required: Optional JSON Type: String

A String defining the type & structure of the data contained within the data field of the Event timeline track. Types are defined by the application provider and are not centrally registered. Implementers are encouraged to use a unique naming scheme, such as Reverse Domain Name Notation, to avoid naming collisions. This field is required if the Section 5.1.12 value is "eventtimeline". This field MUST NOT be used if the packaging value is not "eventtimeline".

5.1.14. Track role

Location: T Required: Optional JSON Type: String

A string defining the role of content carried by the track. Specified roles are described in Table 4. These role values are case-sensitive.

This role field MAY be used in conjunction with the Mimetype Section 5.1.25 to fully describe the content of the track.

Table 4: Reserved track roles

Role	Description
audiodescription	An audio description for visually impaired users
video	Visual content
audio	Audio content
mediatimeline	An MSF media timeline Section 7
eventtimeline	An MSF event timeline Section 8
caption	A textual representation of the audio track
subtitle	A transcription of the spoken dialogue
signlanguage	A visual track for hearing impaired users.

Table 4

Custom roles MAY be used as long as they do not collide with the specified roles.

5.1.15. Is Live

Location: T Required: Yes JSON Type: Boolean

True if new Objects will be added to the track. False if no new Objects will be added to the track. This is sent under two possible conditions: * the publisher of a previously live track has ended the track. * the track is Video-On-Demand (VOD) and was never live.

5.1.16. Target latency

Location: T Required: Optional JSON Type: Number

The target latency in milliseconds. Target latency is defined as the offset in wallclock time between when content was encoded and when it is displayed to the end user. For example, if a frame of video is encoded at 10:08:32.638 UTC and the target latency is 5000, then that frame should be rendered to the end-user at 10:08:37.638 UTC. This field MUST NOT be included if isLive is FALSE. All tracks belonging to the same render group MUST have identical target latencies. All tracks belonging to the same alternate group MUST have identical target latencies. If this field is absent from the track definition, then the player MAY choose the latency with which it renders the content.

5.1.17. Track label

Location: T Required: Optional JSON Type: String

A string defining a human-readable label for the track. Examples might be "Overhead camera view" or "Deutscher Kommentar". Note that the [JSON] spec requires UTF-8 support by decoders.

5.1.18. Render group

Location: T Required: Optional JSON Type: Number

An integer specifying a group of tracks which are designed to be rendered together. Tracks with the same group number SHOULD be rendered simultaneously, are time-aligned and are designed to accompany one another. A common example would be tying together audio and video tracks.

5.1.19. Alternate group

Location: T Required: Optional JSON Type: Number

An integer specifying a group of tracks which are alternate versions of one-another. Alternate tracks represent the same media content, but differ in their selection properties. Alternate tracks MUST have matching media time sequences. A subscriber typically subscribes to one track from a set of tracks specifying the same alternate group number. A common example would be a set video tracks of the same content offered in alternate bitrates.

5.1.20. Initialization data

Location: T Required: Optional JSON Type: String

A string holding Base64 [BASE64] encoded initialization data for the track.

5.1.21. Dependencies

Location: T Required: Optional JSON Type: Array

Certain tracks may depend on other tracks for decoding. Dependencies holds an array of track names Section 5.1.11 on which the current track is dependent. Since only the track name is signaled, the namespace of the dependencies is assumed to match that of the track declaring the dependencies.

5.1.22. Temporal ID

Location: T Required: Optional JSON Type: Number

A number identifying the temporal layer/sub-layer encoding of the track, starting with 0 for the base layer, and increasing by 1 for the next higher temporal fidelity.

5.1.23. Spatial ID

Location: T Required: Optional JSON Type: Number

A number identifying the spatial layer encoding of the track, starting with 0 for the base layer, and increasing by 1 for the next higher fidelity.

5.1.24. Codec

Location: T Required: Optional JSON Type: String

A string defining the codec used to encode the track. For LOC packaged content, the string codec registrations are defined in Sect 3 and Section 4 of [WEBCODECS-CODEC-REGISTRY].

5.1.25. Mimetype

Location: T Required: Optional JSON Type: String

A string defining the mime type [MIME] of the track.

5.1.26. Framerate

Location: T Required: Optional JSON Type: Number

A number defining the video framerate of the track, expressed as frames per second.

5.1.27. Timescale

Location: T Required: Optional JSON Type: Number

The number of time units that pass per second.

5.1.28. Bitrate

Location: T Required: Optional JSON Type: Number

A number defining the bitrate of track, expressed in bits per second.

5.1.29. Width

Location: T Required: Optional JSON Type: Number

A number expressing the encoded width of the video frames in pixels.

5.1.30. Height

Location: T Required: Optional JSON Type: Number

A number expressing the encoded height of the video frames in pixels.

5.1.31. Audio sample rate

Location: T Required: Optional JSON Type: Number

The number of audio frame samples per second. This property SHOULD only accompany audio codecs.

5.1.32. Channel configuration

Location: T Required: Optional JSON Type: String

A string specifying the audio channel configuration. This property SHOULD only accompany audio codecs. A string is used in order to provide the flexibility to describe complex channel configurations for multi-channel and Next Generation Audio schemas.

5.1.33. Display width

Location: T Required: Optional JSON Type: Number

A number expressing the intended display width of the track content in pixels.

5.1.34. Display height

Location: T Required: Optional JSON Type: Number

A number expressing the intended display height of the track content in pixels.

5.1.35. Language

Location: T Required: Optional JSON Type: String

A string defining the dominant language of the track. The string MUST be one of the standard Tags for Identifying Languages as defined by [LANG].

5.1.36. Parent name

Location: T Required: Optional JSON Type: String

A string defining the parent track name Section 5.1.11 to be cloned. This field MUST only be included inside a Clone tracks Section 5.1.5 object.

5.1.37. Track duration

Location: T Required: Optional JSON Type: Number

The duration of the track expressed in integer milliseconds. This field MUST NOT be included if the isLive Section 5.1.15 field value is true.

5.2. Delta updates

A catalog update might contain incremental changes. This is a useful property if many tracks may be initially declared but then there are small changes to a subset of tracks. The producer can issue a delta update to describe these changes. Changes are described incrementally, meaning that a delta update can itself modify a prior delta update.

A restricted set of operations are allowed with each delta update: * Add a new track that has not previously been declared. * Add a new track by cloning a previously declared track. * Remove a track that has been previously declared.

The following rules are to be followed in constructing and processing delta updates:

* A delta update MUST include the Delta Update Section 5.1.2 field set to true.

* A delta update catalog MUST contain at least one instance of Add tracks Section 5.1.3, Remove tracks Section 5.1.4 or Clone Tracks Section 5.1.5 fields and MAY contain more. It MUST NOT contain an instance of a Tracks Section 5.1.8 field or an MSF version Section 5.1.1 field.

* The Add, Delete and Clone operations are applied sequentially in the order they are declared in the document. Each operation in the sequence is applied to the target document; the resulting document becomes the target of the next operation. Evaluation continues until all operations are successfully applied.

* A Cloned track inherits all the attributes of the track defined by the Parent Name Section 5.1.36, except the Track Name which MUST be new. Attributes redefined in the cloning Object overwrite inherited values.

* The tuple of Track Namespace and Track Name defines a fixed set of Track attributes which MUST NOT be modified after being declared. To modify any attribute, a new track with a different Namespace|Name tuple is created by Adding or Cloning and then the old track is removed.

5.3. Catalog Examples

The following section provides non-normative JSON examples of various catalogs compliant with this draft.

5.3.1. Time-aligned Audio/Video Tracks with single quality

This example shows a catalog for a media producer capable of sending LOC packaged, time-aligned audio and video tracks.

```
{
  "version": 1,
  "generatedAt": 1746104606044,
  "tracks": [
    {
      "name": "1080p-video",
      "namespace": "conference.example.com/conference123/alice",
      "packaging": "loc",
      "isLive": true,
      "targetLatency": 2000,
      "role": "video",
      "renderGroup": 1,
      "codec":"av01.0.08M.10.0.110.09",
      "width":1920,
      "height":1080,
      "framerate":30,
      "bitrate":1500000
    },
    {
      "name": "audio",
      "namespace": "conference.example.com/conference123/alice",
      "packaging": "loc",
      "isLive": true,
      "targetLatency": 2000,
      "role": "audio",
      "renderGroup": 1,
      "codec":"opus",
      "samplerate":48000,
      "channelConfig":"2",
      "bitrate":32000
    }
  ]
}
```

5.3.2. Simulcast video tracks - 3 alternate qualities along with audio

This example shows catalog for a media producer capable of sending 3 time-aligned video tracks for high definition, low definition and medium definition video qualities, along with an audio track. In this example the namespace is absent, which infers that each track must inherit the namespace of the catalog.

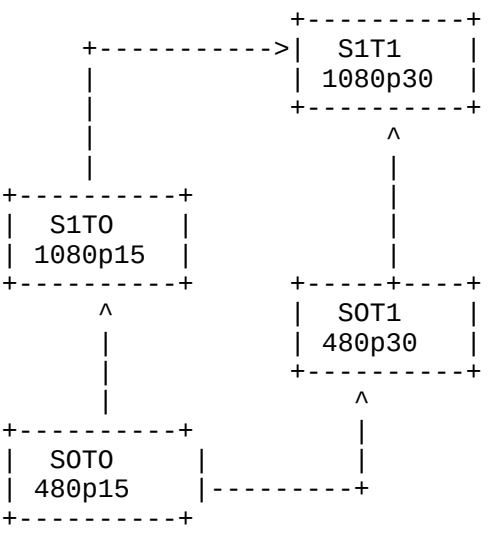
```
{
  "version": 1,
  "generatedAt": 1746104606044,
  "tracks":[
    {
      "name": "hd",
```

```
        "renderGroup": 1,
        "packaging": "loc",
        "isLive": true,
        "targetLatency": 1500,
        "role": "video",
        "codec":"av01",
        "width":1920,
        "height":1080,
        "bitrate":5000000,
        "framerate":30,
        "altGroup":1
      },
      {
        "name": "md",
        "renderGroup": 1,
        "packaging": "loc",
        "isLive": true,
        "targetLatency": 1500,
        "role": "video",
        "codec":"av01",
        "width":720,
        "height":640,
        "bitrate":3000000,
        "framerate":30,
        "altGroup":1
      },
      {
        "name": "sd",
        "renderGroup": 1,
        "packaging": "loc",
        "isLive": true,
        "targetLatency": 1500,
        "role": "video",
        "codec":"av01",
        "width":192,
        "height":144,
        "bitrate":500000,
        "framerate":30,
        "altGroup":1
      },
      {
        "name": "audio",
        "renderGroup": 1,
        "packaging": "loc",
        "isLive": true,
        "targetLatency": 1500,
        "role": "audio",
        "codec":"opus",
```

```
        "samplerate":48000,
        "channelConfig":"2",
        "bitrate":32000
      }
     ]
  }
```

5.3.3. SVC video tracks with 2 spatial and 2 temporal qualities

This example shows catalog for a media producer capable of sending scalable video codec with 2 spatial and 2 temporal layers with a dependency relation as shown below:



The corresponding catalog uses "depends" attribute to express the track relationships.

```
{
  "version": 1,
  "generatedAt": 1746104606044,
  "tracks":[
    {
      "name": "480p15",
      "namespace": "conference.example.com/conference123/alice",
      "renderGroup": 1,
      "packaging": "loc",
      "isLive": true,
      "role": "video",
      "codec":"av01.0.01M.10.0.110.09",
      "width":640,
```

```
          "height":480,
          "bitrate":3000000,
          "framerate":15
        },
        {
          "name": "480p30",
          "namespace": "conference.example.com/conference123/alice",
          "renderGroup": 1,
          "packaging": "loc",
          "isLive": true,
          "role": "video",
          "codec":"av01.0.04M.10.0.110.09",
          "width":640,
          "height":480,
          "bitrate":3000000,
          "framerate":30,
          "depends": ["480p15"]
        },
        {
          "name": "1080p15",
          "namespace": "conference.example.com/conference123/alice",
          "renderGroup": 1,
          "packaging": "loc",
          "isLive": true,
          "role": "video",
          "codec":"av01.0.05M.10.0.110.09",
          "width":1920,
          "height":1080,
          "bitrate":3000000,
          "framerate":15,
          "depends":["480p15"]
        },

        {
          "name": "1080p30",
          "namespace": "conference.example.com/conference123/alice",
          "renderGroup": 1,
          "packaging": "loc",
          "isLive": true,
          "role": "video",
          "codec":"av01.0.08M.10.0.110.09",
          "width":1920,
          "height":1080,
          "bitrate":5000000,
          "framerate":30,
          "depends": ["480p30", "1080p15"]
        },
        {
```

```
        "name": "audio",
        "namespace": "conference.example.com/conference123/alice",
        "renderGroup": 1,
        "packaging": "loc",
        "isLive": true,
        "role": "audio",
        "codec":"opus",
        "samplerate":48000,
        "channelConfig":"2",
        "bitrate":32000
      }
    ]
  }
```

5.3.4. Delta update - adding two tracks

This example shows the catalog delta update for a media producer adding two tracks to an established video conference. One track is newly declared, the other is cloned from a previous track.

```
{
  "deltaUpdate": true,
  "generatedAt": 1746104606044,
  "addTracks": [
      {
        "name": "slides",
        "isLive": true,
        "role": "video",
        "codec": "av01.0.08M.10.0.110.09",
        "width": 1920,
        "height": 1080,
        "framerate": 15,
        "bitrate": 750000,
        "renderGroup": 1
      }
   ],
   "cloneTracks": [
      {
        "parentName": "video-1080",
        "name": "video-720",
        "width":1280,
        "height":720,
        "bitrate":600000
      }
   ]
}
```

5.3.5. Delta update removing tracks

This example shows a delta update for a media producer removing two tracks from an established video conference.

```
{
  "deltaUpdate": true,
  "generatedAt": 1746104606044,
  "removeTracks": [{"name": "video"},{"name": "slides"}]
}
```

5.3.6. Time-aligned Audio/Video Tracks with custom field values

This example shows catalog for a media producer capable of sending LOC packaged, time-aligned audio and video tracks along with custom fields in each track description.

```
{
  "version": 1,
  "generatedAt": 1746104606044,
  "tracks": [
    {
      "name": "1080p-video",
      "namespace": "conference.example.com/conference123/alice",
      "packaging": "loc",
      "isLive": true,
      "role": "video",
      "renderGroup": 1,
      "codec":"av01.0.08M.10.0.110.09",
      "width":1920,
      "height":1080,
      "framerate":30,
      "bitrate":1500000,
      "com.example-billing-code": 3201,
      "com.example-tier": "premium",
      "com.example-debug": "h349835bfkjfg82394d945034jsdfn349fns"
    },
    {
      "name": "audio",
      "namespace": "conference.example.com/conference123/alice",
      "packaging": "loc",
      "isLive": true,
      "role": "audio",
      "renderGroup": 1,
      "codec":"opus",
      "samplerate":48000,
      "channelConfig":"2",
      "bitrate":32000
    }
  ]
}
```

5.3.7. Time-aligned VOD Audio/Video Tracks

This example shows catalog for a media producer offering VOD (video on-demand) non-live content. The content is LOC packaged, and includes time-aligned audio and video tracks.

```
{
  "version": 1,
  "tracks": [
    {
      "name": "video",
      "namespace": "movies.example.com/assets/boy-meets-girl-season3/episode5",
      "packaging": "loc",
      "isLive": false,
      "trackDuration": 8072340,
      "renderGroup": 1,
      "codec":"av01.0.08M.10.0.110.09",
      "width":1920,
      "height":1080,
      "framerate":30,
      "bitrate":1500000
    },
    {
      "name": "audio",
      "namespace": "movies.example.com/assets/boy-meets-girl-season3/episode5",
      "packaging": "loc",
      "isLive": false,
      "trackDuration": 8072340,
      "renderGroup": 1,
      "codec":"opus",
      "samplerate":48000,
      "channelConfig":"2",
      "bitrate":32000
    }
  ]
}
```

5.3.8. Media timeline and Event timeline

This example shows a catalog for a media producer capable of sending LOC packaged, time-aligned audio and video tracks, along with a Media Timeline which describes the history of those tracks and an Event Timeline providing synchronized data.

```
{
  "version": 1,
  "generatedAt": 1746104606044,
  "tracks": [
    {
      "name": "history",
      "namespace": "conference.example.com/conference123/alice",
      "packaging": "mediatimeline",
      "mimetype": "application/json",
      "depends": ["1080p-video","audio"]
```

```
      },
      {
        "name": "identified-objects",
        "namespace": "another-provider/time-synchronized-data",
        "packaging": "eventtimeline",
        "eventType": "com.ai-extraction/appID/v3",
        "mimetype": "application/json",
        "depends": ["1080p-video"]
      },
      {
        "name": "1080p-video",
        "namespace": "conference.example.com/conference123/alice",
        "packaging": "loc",
        "isLive": true,
        "targetLatency": 2000,
        "role": "video",
        "renderGroup": 1,
        "codec":"av01.0.08M.10.0.110.09",
        "width":1920,
        "height":1080,
        "framerate":30,
        "bitrate":1500000
      },
      {
        "name": "audio",
        "namespace": "conference.example.com/conference123/alice",
        "packaging": "loc",
        "isLive": true,
        "targetLatency": 2000,
        "role": "audio",
        "renderGroup": 1,
        "codec":"opus",
        "samplerate":48000,
        "channelConfig":"2",
        "bitrate":32000
      }
    ]
  }
```

5.3.9. Terminating a live broadcast

This example shows catalog for a media producer terminating a previously live broadcast containing a video and an audio track.

```
{
  "version": 1,
  "generatedAt": 1746104606044,
  "isComplete": true,
  "tracks": []
}
```

6. Media transmission

The MOQT Groups and MOQT Objects need to be mapped to MOQT Streams. Irrespective of the Section 4 in place, each MOQT Object MUST be mapped to a new MOQT Stream.

6.1. Group numbering

The Group ID of the first Group published in a track at application startup MUST be a unique integer that will not repeat in the future. One approach to achieve this is to set the initial Group ID to the creation time of the first Object in the group, represented as the number of milliseconds since the Unix epoch, rounded to the nearest millisecond. This ensures that republishing the same track in the future, such as after a loss of connectivity or an encoder restart, will not result in smaller or duplicate Group IDs for the same track name. Note that this method does not prevent duplication if more than 1000 groups are published per second.

Each subsequent Group ID MUST increase by 1.

If a publisher is able to maintain state across a republish, it MUST signal the gap in Group IDs using the MOQT Prior Group ID Gap Extension header.

7. Media Timeline track

The media timeline track provides data about the previously published groups and their relationship to wallclock time and media time. Media timeline tracks allow players to seek to precise points behind the live head in a live broadcast, or for random access in a VOD asset. Media timeline tracks are optional. Multiple media timeline tracks can exist inside a catalog.

7.1. Media Timeline track payload

A media timeline track is a JSON [JSON] document. This document MAY be compressed using GZIP [GZIP]. The document contains an array of records. Each record consists of an array of three required items, whose ordinal position defines their type:

* The first item holds the media presentation timestamp, expressed as a JSON Number. This value MUST match the media presentation timestamp, rounded to the nearest millisecond, of the first media sample in the referenced Object

* The second item holds the MOQT Location of the entry, defined as a tuple of the MOQT Group ID and MOQT Object ID, and expressed as a JSON Array of Numbers, where the first number is the Group ID and the second number is the Object ID.

* The third time holds the wallclock time at which the media was encoded, defined as the number of milliseconds that have elapsed since January 1, 1970 (midnight UTC/GMT) and expressed as a JSON Number. For VOD assets, or if the wallclock time is not known, the value SHOULD be 0.

An example media timeline is shown below:

```
[
  [0, [0,0], 1759924158381],
  [2002, [1,0], 1759924160383],
  [4004, [2,0], 1759924162385],
  [6006, [3,0], 1759924164387],
  [8008, [4,0], 1759924166389]
]
```

7.2. Media Timeline Catalog requirements

A media timeline track MUST carry a ’type’ identifier in the Catalog with a value of "mediatimeline". A media timeline track MUST carry a ’depends’ attribute which contains an array of all track names to which the media timeline track applies. The mime-type of a media timeline track MUST be specified as "application/json".

7.3. Media Timeline track updating

The publisher MUST publish an independent media timeline in the first MOQT Object of each MOQT Group of a media timeline track. The publisher MAY publish incremental updates in the second and subsequent Objects within each Group. Incremental updates only contain media timeline records since the last media timeline Object.

8. Event Timeline track

The event timeline track provides a mechanism to associate ad-hoc event metadata with the broadcast. Use-case examples include live sports score data, GPS coordinates of race cars, SAP-types for media segments or active speaker notifications in web conferences.

To allow the client to bind this event metadata with the broadcast content described by the media timeline track, each event record MUST contain a reference to one of Media PTS, wallclock time or MOQT Location.

Event timeline tracks are optional. Multiple event timeline tracks can exist inside a catalog. The type & structure of the data contained within each event timeline track is declared in the catalog, to facilitate client selection and parsing.

8.1. Event Timeline data format

An event timeline track is a JSON [JSON] document. This document MAY be compressed using GZIP [GZIP]. The document contains an array of records. Each record consists of a JSON Object containing the following required fields:

* An index reference, which MUST be either 't' for wallclock time, 'l' for Location or 'm' for Media PTS. Only one of these index values may be used within each record. Event timelines SHOULD use the same index reference type for each record. The definitions for wallclock time, Location and Media PTS are identical to those defined for media timeline payload Section 7.1. Wallclock time and media PTS values are JSON Number, while Location value is an Array of Numbers, where the first item represents the MOQT GroupID and the second item the MOQT Object ID.

* A 'data' Object, whose structure is defined by the Section 5.1.13 value declared for this track in the Catalog.

8.2. Event Timeline Catalog requirements

An event timeline track MUST carry:

* a Section 5.1.12 attribute with a value of "eventtimeline".

* a Section 5.1.21 attribute which contains an array of all track names to which the event timeline track applies.

* a Section 5.1.25 attribute with a value of "application/json".

* an Section 5.1.13 attribute declaring the type & structure of data contained in the event timeline track.

8.3. Event Timeline track updating

The publisher MUST publish an independent event timeline in the first MOQT Object of each MOQT Group of an event timeline track. The publisher MAY publish incremental updates in the second and subsequent Objects within each Group. Incremental updates only contain event timeline records since the last event timeline Object.

8.4. Event timeline track examples

8.4.1. Event timeline track with wallclock time indexing

This example shows how sports scores and game information might be defined in a live sports broadcast.

```
[
    {
        "t": 1756885678361,
        "data": {
            "status": "in_progress",
            "period": 1,
            "clock": "12:00",
            "homeScore": 0,
            "awayScore": 0,
            "lastPlay": "Game Start"
        }
    },
    {
        "t": 1756885981542,
        "data": {
            "status": "in_progress",
            "period": 1,
            "clock": "09:25",
            "homeScore": 2,
            "awayScore": 0,
            "lastPlay": "Team A: #23 makes 2-pt jump shot"
        }
    }
]
```

8.4.2. Event timeline track with MOQT Location indexing

This example shows drone GPS coordinates synched with the start of each Group.

```
[
    {
        "l": [0,0],
        "data": [47.1812,8.4592]
    },
    {
        "l": [1,0],
        "data": [47.1662,8.5155]
    }
]
```

9. Workflow

9.1. Initiating a broadcast

An MSF publisher MUST publish a catalog track object before publishing any media track objects.

9.2. Ending a live broadcast

After publishing a catalog and defining tracks carrying live content, an original publisher can deliver a deterministic signal to all subscribers that the broadcast is complete by taking the following steps:

* Send a SUBSCRIBE_DONE (See MOQT Sect 8.1.2) message for all active tracks using status code 0x2 Track Ended.

* If the live stream is being converted instantly to a VOD asset, then publish an independent (non-delta) catalog update which, for each track, sets isLive Section 5.1.15 to FALSE and adds a track duration Section 5.1.37 field.

* If the live stream is being terminated permanently without conversion to VOD, then publish an independent catalog update which signals isComplete Section 5.1.7 as TRUE and which contains an empty Tracks Section 5.1.8 field.

10. Security Considerations

ToDo

11. IANA Considerations

This document creates a new entry in the "MoQ Streaming Format" Registry (see [MoQTransport] Sect 8). The type value is 0x001, the name is "MOQT Streaming Format" and the RFC is XXX.

12. Normative References

[BASE64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[GZIP] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <https://www.rfc-editor.org/rfc/rfc1952>.

[IMSC1] "W3C, TTML Profiles for Internet Media Subtitles and Captions 1.0 (IMSC1)", April 2016, <https://www.w3.org/TR/ttml-imsc1/>.

[JSON] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[LANG] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/rfc/rfc5646>.

[LOC] Zanaty, M., Nandakumar, S., and P. Thatcher, "Low Overhead Media Container", Work in Progress, Internet-Draft, draft-mzanaty-moq-loc-05, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-mzanaty-moq-loc-05>.

[MIME] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[MoQTransport] Nandakumar, S., Vasiliev, V., Swett, I., and A. Frindell, "Media over QUIC Transport", Work in Progress, Internet-Draft, draft-ietf-moq-transport-11, 28 April 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-moq-transport-11>.

[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>.

[RFC4180] Shafranovich, Y., "Common Format and MIME Type for Comma-Separated Values (CSV) Files", RFC 4180, DOI 10.17487/RFC4180, October 2005, <https://www.rfc-editor.org/rfc/rfc4180>.

[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>.

[WEBCODECS-CODEC-REGISTRY] "WebCodecs Codec Registry", September 2024, <https://www.w3.org/TR/webcodecs-codec-registry/>.

[WEBVTT] "World Wide Web Consortium (W3C), WebVTT: The Web Video Text Tracks Format", April 2019, <https://www.w3.org/TR/webvtt1/>.

Acknowledgments

* the MoQ Workgroup and mailing lists.

Contributors

The following persons where the co-authors of the individual draft (draft-law-moq-warpstreamingformat) this document is based on:

* Luke Curley
* Victor Vasiliev
* Suhas Nandakumar
* Kirill Pugin
* Will Law

Author’s Address

Will Law
Akamai
Email: wilaw@akamai.com