ATSC 3.0 - MMT Design Enhancement

MMT and SCTE35 Support
Encoder to Signaler and Scheduler
Enhancement Design

Draft

Document Revision: v1.8

Author: Jason Justman - jjustman@sinclairdigital.com

Original Draft Attendees: Attendees: OneMedia 3.0, SBG, ATEME, Enensys
Original Draft Genesis Location: Rennes, FR

Technical Overview:

The systems interface observed between ATSC 3.0 encoder, signaler and scheduler have been proven for baseline operational functionality by the delivery of a completed MPD and MPU payload.

After technical analysis of this method, encoder constitution of the ISOBMFF fragment is the correct model for ROUTE delivery and restricts the design objective of MMTP in regards to optimal MFU transmission latency in non-uniform GOP scenarios, including SHVC, where base and enhancement layers may be delivered at spatial, temporal, and closed or open long-gop for optimal codec efficiency.

The utilization of a complete MPU (e.g. a ISOBMFF box including trun data) from an encoder to a packager, rather than fragmented MFU emission for MMTP delivery into the Signaler and Scheduler prevents adoption of two mission critical use cases in alignment for ATSC 3.0.

Problem Statement:

1. In order to support end-to-end SCTE35 triggering in both ROUTE and MMT models, an analysis of encoder output was performed to determine the correct placement of this payload. Multiplexing in the MPD handoff for ROUTE is a valid use case and model, but limits the ability to emit an MMT Signaling Message to represent a transport-specific signal in SCTE-35 without a requirement of an application aware signaling message. This maintains the technical design requirement and spirit of SCTE35.

2. MMTP with MPU re-assembly is a function of the decoder, not the encoder to scheduler interface. In order to be in alignment of the spirit of MMT - fragmented delivery between the encoder, signaler and scheduler needs to be a requirement to differentiate core transport of MMT vs. ROUTE

Business Objectives:

In order to facilitate successful Next-Gen Broadcasting operational models, it is imperative that the ad-technology problem in ATSC 3.0 be solved. Heritage broadcasters are facing increasing challenges from the loss of retransmission revenue by changes in MVPD contractual relationships, limited solutions for addressable and interactive advertising in traditional ATSC 1.0 models, and a move to 'actuals' impression-based metrics from Digital monetization technology.

The process of unifying advertising opportunities for Broadcasters across Heritage ATSC 1.0, Next-generation ATSC 3.0, Digital/OTT platforms and Cable/MSO/MVPD retransmission begins at a common signaling format standard. This signaling format can provide a common signalizing format, and adoption of multiple fulfillment use cases aligning with each distribution mechanism, to reach the right audience, with the right content, on any device.

Current ATSC 3.0 working groups are developing a DASH-specific mechanism based upon XML, called XLink. This new format for ad break triggering is successful for 'segmented' content transmission, for example VOD content of DASH, but has severe ramifications for its flexibility. In the XLink solution, the ad-break signaling indicator is moved from the traditional transport layer into a more complex and constrained application layer. This shift, and the loss of a singular advertising the fails to address Next-Gen Live Linear, Non-connected devices, and Re-transmission use cases for revenue growth.

The objective of this design proposal is to outline a core foundation for Next Generation Broadcasting starts with SCTE-35. By adopting an industry-proven signaling technology for ATSC 3.0 MMT, this design enables the compatibility required and flexibility for innovation needed to finally address, and solve, the AdTech problem in Next Generation Broadcast.

Technical Overview:

This document outlines a common, in-band workflow for addressing baseline interoperability for ATSC 3.0 MMT content supply chain, with system component touchpoints between the Encoder, Packager, Signaler, and Scheduler for solution design.

By relying on in-band signaling and existing and extending ISOBMFF box definitions, MMT Signaling Information, and alignment into a baseline A/344 runtime application, the objective in this document is to demonstrate end-to-end success and findings in preparation of adoption of common message identifiers. For more information, please review the SCTE website and latest standards publication at < https://www.scte.org/SCTEDocs/Standards/ANSI_SCTE%2035%202019r1.pdf >

Business Use Case:

The potential for broadcasters to unify ad break management and decisioning across distribution platforms is a long-term goal and objective, but in order to gain validated learnings from this unified technology, the following 5 Business Use cases are outlined for demonstration.

In each use-case, these models are supported by existing attributes in the SCTE-35 standard. By adopting pre-defined and recommended "Descriptor" types from the SCTE-35 recommended practice, confusion and integration can be scoped to achieve a successful end-to-end solution with common industry standard terminology.

This technology, when combined with programmatic data relevant to each use case in the SCTE-35 payload, can enable one message to reach multiple technology, connectivity, and interactivity use cases.

1. Traditional Linear Insertion over ATSC 3.0 MMT - Local, Zonal - In this use-case, the receiving device would perform the 'splice' by switching between MMT flows, e.g. by using the A/344 runtime RMP.

   This solution on the client, along with optimized encoding of a secondary (or tertiary via LDM) linear stream can be invoked via a single SCTE-35 command with the destination MMT_service_id as a 'program splice' event.

2. MVPD Providers - Inventory Carve/Share/Scrape at Regional/Local/Zonal/Addressable: In this use case, an agreed upon SCTE-35 'tier', along with encryption, can be used to provide a transport-level signaling mechanism for ad-break availability. Broadcasters can then align effective relationships of inventory that would otherwise be unavailable to MVPD distributors, including multiple tiers and scrapes of opportunities to ensure both parties can optimize yield.

   By keeping in the transport-level, this solution can then be leveraged by a trans-multiplex (transmux) - converting the HEVC and AC4/MPEG-H video and audio flows into a QAM/IP compatible MPEG-TS for direct multiplexing of essence content into the MVPD's network.

3. Pre-positioned/NRT Linear Insertions - Device - Non-Internet Connected:
   In this use case, Live Linear programming delivered via MMT and the corresponding A/344 runtime app can switch video playback to a locally-decisioned and pre-positioned creative from the SCTE-35 trigger. Multiple descriptors, along with multiple triggers in the same ad break (using tiers) can be used to trigger selections of proper ad selected copy and ad replacement opportunities.

   This semi-personalized delivery would allow for a brand-based advertiser to deliver a demographic targeted pre-emption (e.g. Coca-Cola to afternoon day-part watchers, lifestyle program watchers, etc.), and if needed, selectivity/pre-emptibility in the A/344.

   Likewise, Zonal insertions can also be adopted in the pre-positioned model by emitting a signaling flow with the SCTE-35 trigger on a LDM emission - or the collection of end-user data - e.g. Zip code captured via A/344 application from Value-Add interactive service (e.g. Weather Forecast/Alerting) into device-specific retention via local storage or cookie drop, and final ad decisioning in a non-broadband connected environment via selection of a SCTE-35 segmentation_descriptor indicating replacement if zip-code matches pre-emptiable creative copy.

4. Fully addressable VAST digital ad insertion. For devices that are internet connected, the SCTE-35 trigger be multiplexed early as a "preroll" using mux and pts_execute time, as needed and contain a URL resource to a VAST ad decision server for decisioning and creative download.

   In order to provide a linear-like experience, the mux time to prepare for the ad break must be long-enough for ad decisioning and download/streaming of the replaced content. By increasing the "preroll" length, a broadcast-quality experience for addressable digial ad insertion can be provided, without negatively impacting broadcast latency. Non-broadcast quality experiences (e.g. buffering/spinning wheel) that interrupt the viewing experience are not acceptable, and if the ad decision/creative payload is not completely downloaded (or cached) by the splice execute time, the ad is not replaced. Feedback information to the ad network over the internet backchannel can give broadcasters insight to any ads that were not run because of network buffering or delays.

   This ad decisioning flow would provide a fully-addressable and reportable impression, along with enriched metadata for ad decisioning - including contextual metadata of content where the ad is placed, by leveraging industry standard EIDR or TMSid as program descriptors in the SCTE-35 message.

Additionally, impression beacons for non-replaced (e.g. rolled-thru) network or barter insertions can be reported by internet-connected devices with proper metadata provided via the SCTE-35 message. By injecting a SCTE-35 time-signal command at the linear time the ad begins and finishes with ad beacons (containing AdID), a subset of audience impression data for traditional eGRP/Share of Voice linear buyers for demographic data enrichment can be developed for media buyers.

IAB standard supported companion ads, overlays, snipes/bugs/L-bar/J-bar with linear content. By providing a VAST resource, with a non-linear creative overlay (no video emission) - a low overhead solution can be delivered to provide enhanced graphic or textual information along with a traditional linear creative spot placement.

5. Fully addressable VAST digital ad insertion with pre-positioned creative copy in partnership with an ad-network. In this model, only the ad decisioning call and response is required to be fulfilled by the internet backchannel, and any other linear content would be pre-positioned via NRT/carousel. By providing a unified identifier of the creative copy in the VAST payload response (e.g. Ad-ID), along with a traditional URL, devices can use a non-internet delivered creative, reducing the time and latency for effective digital ad insertion at scale.

   By adopting these 5 base use-cases, with the technology foundation described in this document, Next Gen Broadcast AdTech delivers on reducing per-impression transmission cost of DAI, increasing value with increased insight for advertisers, and enhancing the advertising experience by delivering the right content, on any device, to the right audience.

Architecture:
This document assumes the following architecture based US ATSC 3.0 System Design:

TS -> Encoder -> WEBDAV push (chunked) to Packager/Signaler -> Scheduler -> STLTP

ATSC 3.0 Exciter -> Tuner -> Demodulator -> A/331 receiver

Recommended Design Enhancements in a hypothetical Encoder/Packager flow for MMT sender operation with MFU emission:

Encoder:
    WEBDAV push:

```
while(true) {

    //start of MPU Fragmented Push
    webdav_connect_with_chunked_http_and_init_box(decode_parameters)

    while(!gop_closed) {
        sem_wait(sample_complete)  //single mfu sample is available from encoder buffer
        webdav_chunked_http_push_sample(mfu_sample_block)

    }
    webdav_close_with_chunked_http_and_init_box(moof_block)
}
```

Cadence: "Derivation from MFU specification"
where mpu_fragment_type: (packet_id ^ mpu_sample_number)

0x00
0x01
0x02
0x02
....
(end of MPU

Cadence: "American ATSC 3.0 MMT"
where mpu_fragment_type: (packet_id ^ mpu_sample_number)

0x00
0x02
....
0x02
0x01
(end of MPU)

See Appendix for examples of Informative vs. Normative ISO23008-1 2017 Specification regarding ambiguity.

Functional Implementation of Encoder to Signaler interface:

webdav_connect_with_chunked_http_and_init_box: http payload:

```
HTTP/1.1 200 OK
Content-Type: binary/octet
Transfer-Encoding: chunked

36\r\n
    ftyp...\r\n
```

(0x00 0x00 0x00 0x24...)

(connection open: `http_output_fd`)

webdav_chunked_http_push_sample(mfu_sample_block_t) http payload:

(continued open connection)

```
char[9] mfu_sample_size = ''
snprintf(&mfu_sample_size, "%u", mfu_sample_block_t->size);
fwrite (http_output_fd, strlen(mfu_sample_size, 1,
&mfu_sample_size)
fwrite(http_output_fd, 2, 1, "\r\n")
fwrite(http_output_fd, mfu_sample_block->size, 1,
mfu_sample_block->data)
fwrite(http_output_fd, 2, 1, "\r\n")
```

(0x00 0x00 0x00 0x01...)

wherein:

```
mfu_sample_block {
   uint8_t* data;
   uint32_t size;
} mfu sample_block_t
```

Encoder sample hint track payload enhancement for NTP Clock correlation along with MPEGTS PCR reference for inclusion of SCTE-35:

```
muli->addChild(mjsd);
if(scte_35_payload) {
   muli->addChild(mjgp);
}
```

8.3.1 Definition

Each media sample will be assigned to one or more MFUs. Each sample of the MMT hint track will generate one or more MFUs. The hint sample may omit certain bytes of an MFU if deemed redundant, such as the length field of a NAL unit in the case of AVC or HEVC video bitstreams.

8.3.2 Syntax

```
aligned(8) class MMTHSample {

   unsigned int(32) sequence_number;
   if (is_timed) {
      signed int(8) trackrefindex;
      unsigned int(32) movie_fragment_sequence_number
      unsigned int(32) samplenumber;
      unsigned int(8)  priority;
      unsigned int(8)  dependency_counter;
      unsigned int(32) offset;
      unsigned int(32) length;
      multiLayerInfo();
   } else {
      unsigned int(16) item_ID;
   }
}


aligned(8) class multiLayerInfo extends Box("muli") {
   bit(1) multilayer_flag;
   bit(7) reserved0;

   if (multilayer_flag==1) {
      bit(3) dependency_id;
      bit(1) depth_flag;
      bit(4) reserved1;
      bit(3) temporal_id;
      bit(1) reserved2;
      bit(4) quality_id;
      bit(6) priority_id;
      bit(10) view_id;
   }
   else{
      bit(6) layer_id;
      bit(3) temporal_id;
      bit(7) reserved3;
   }
}
```

Proposed Box Type for NTP emission from encoder to packager/signaler on MFU basis:

Box Type: `mjsd`
Container: muli
Mandatory: No
Quantity: Zero or Exactly one per sample

```
aligned(8) class mjae_sample_timestamp_descriptor extends Box("mjsd") {

   unsigned int(64) sample_presentation_time;
   unsigned int(64) sample_decode_time;

}

sample_presentation_time

indicates the presentation time of the designated MFU sample by
the 64-bit NTP time stamp format

sample_decode_time

indicates the decode time of the in the designated MFU sample by
the 64-bit NTP timestamp format
```

Proposed Box Type for MPEG-TS PCR and SCTE-35 binary signal payload emission from encoder to packager/signaler:

```
Box Type: `mjgp`
Container: mjsd
Mandatory: No
Quantity: zero or more


aligned(8) class mjae_scte35_binary_payload extends Box ("mjgp"){

   unsigned bit(7)    reserved '1111111'
   unsigned bit(33)   pts_timestamp;

   unsigned int(16)   signal_data_size;
   bit(8)             signal_data[];
}

pts_timestamp

indicates the MPEG-TS clock reference (90,000Hz) of the SCTE35_signal muxed
in relation to the sample_presentation_time in the parent box

It is always assumed that the PTS_timestamp is the mux timestamp of the SPTS
pid (MPEG-TS: single program transport stream) containing the message.

signal_data_size

unsigned 16 bit integer representing the number of bytes in the following
signal_data payload

signal_data

binary payload of the raw SCTE35 splice command
```

Citation:

ISO/IEC 14496-12:2005(E)
Boxes with an unrecognized type shall be ignored and skipped.

Proposed Packager/Signaler requirements for inclusion of NTP and SCTE35 message into MMT emission:

Objectives:

1.) Transport from the Encoder to the Packager of the relevant sample_presentation_time via the mjsd box (see pp. 6-10)

2.) Transport from the Encoder to the Packager of any applicable SCTE-35 messages that would be contained in the Encoder Input Mux, and for the Packager to re-emit this as a new MMT Signaling Information message to provide the SCTE-35 message as a transport-level signal

3.) Packager to Signaler emission of a representative A/331 mmt_atsc3_message emission (providing functionality ala the MMT inband_event_descriptor A/337:2018, delivered in the payload of a mmt_atsc3_message), with the event_value_bytes representing the mjgp box payload (enables A/344 application runtime client-side splicing of SCTE35 messages)

4.) (Optional) Packager/Signaler may remove the encoder-provided mjsd/mjgp boxes for resultant output stream for device compatability, provided the mjgp box has been properly translated into an MMT Signalling Information SCTE35_message, and A337 SCTE35_message.

Objective #2:

Signaler requirements for translation of NTP (mjsd) and SCTE35_Message (mjgp) box into Signaling Information (Transport-layer carriage):

```
for each asset {
    for each sample {

        processMMTHSample();
        multiLayerBox = processMultiLayerInfoBox();

        //multiLayerBox will contain 0 or more children, comprised of
        //0..1 msjd box and 0..n mjgp boxes

        if(mjsdBox = multiLayerBox->getChild("mjsd")) {

            uint64_t ntp_timestamp = msjd->sample_presentation_time();

            if(new_mpu_sequence) {
            ////pts and dts capture for MP_Table message with 0x0001 MPU timestamp descriptor

            MPU_timestamp_descriptor_t* MPU_timestamp_descriptor = calloc(1, sizeof(MPU_timestamp_descriptor_t));

            MPU_timestamp_descriptor->mpu_presentation_time = ntp_timestamp;
            //inject into MP_table per this <asset_id, mpu_sequence_number>
            inject_MP_table(asset_id, mpu_sequence_number, MPU_timestamp_descriptor);
        }

        //process any SCTE35_signals into correlating new signaling message type and descriptor

        SCTE35_Signal_Message_t* SCTE35_Signal_Message = calloc(1, sizeof(SCTE35_Signal_Message_t);

        child=0;
        while(mjgpBox = mjsdBox->getChild("mjgp", child++) { //each instance of the scte35_signal

            //add SCTE35_Signal_Descriptor to SCTE35_Signal_message,
            SCTE35_Signal_Message_add_SCTE35_Signal_Descriptor(SCTE35_Signal_Message, ntp_timestamp, mjgpBox-
    >pts_timestamp, &mjgpBox->signal_data, mjgpBox->size-8);

        }

        if(SCTE35_Signal_Message->number_of_SCTE35_Signal_Descriptor) {
            inject_SCTE35_Signal_Message(SCTE35_Signal_Message);
        }
    }
}
```

SCTE35 Message Identifier:

```
0xF337 : SCTE35_Signal message

SCTE35_Signal_Message {
  message_id       16
      version      8
      length       32

      number_of_SCTE35_Signal_Descriptor   N1   8
      SCTE35_Signal_Descriptor()           var  (per length)
}
```

SCTE35 Signal Descriptor:

```
0xF33F: SCTE35_Signal_Descriptor tag

SCTE35_Signal_Descriptor {

      descriptor_tag         16
      descriptor_length      16
      ntp_timestamp          64
      reserved               7 '1111111'
      pts_timestamp          33
      signal_data_length     N1 16
        for(i=0; i < N1; i++) {
          signal_data[i];
       }
}


ntp_timestamp: indicates the presentation time of the designated MFU sample
in which the original SCTE35_signal was muxed nearest, 64-bit NTP time stamp
format

pts_timestamp: indicates the MPEG-TS clock reference (90,000Hz) of the
SCTE35_signal muxed in relation to the sample_presentation_time in the parent
box
```

Note: multiple SCTE35_Signal_Messages may be present over the logical ‘MPU’, but must be as close (or preceeding) to the MFU emission from the packager. This enables compatibility of both SCTE35 messages in which the pts_execute timestamp is not provided, and where the pts_execute is provided, and muxed-in as a preroll or multiple times in a single GOP. The splice_immeidate use case would then use the nearest sample time (ideally pared with an IDR frame from the encoder) to perform the splice.

Objective #3: Packager to Signaler emission of a representative A/331

Signaler requirements for translation of NTP (mjsd) and SCTE35_Message (mjgp) box into normalized A/331:2019 atsc3_message_content payload (application-level carriage of SCTE35 message payload):

atsc3_message_content_type	Meaning
0x1337	MMT SCTE-35 Message encapsulated clock scalar reference via <NTP, PTS> tuple.

```
aligned(8) class atsc3_message_content_type_0x1137 {

   unsigned int(64)   mmt_ntp_timestamp;

  unsigned bit(7)    reserved '1111111'
  unsigned bit(33)   pts_timestamp;

  unsigned int(16)   signal_data_length;
  bit(8)             signal_data[];
}
```

mmt_ntp_timestamp: indicates the presentation time of the designated MFU sample in which the original SCTE35_signal was muxed nearest, 64-bit NTP time stamp format

pts_timestamp: indicates the MPEG-TS clock reference (90,000Hz) of the SCTE35_signal muxed in relation to the sample_presentation_time in the parent box

Objective #4: Removal of ISOBMFF child boxes appended on per-sample basis

Removal of the mjsd and mjgp boxes by the packager emission is recommended, as early testing of receivers do not appear to honor the ISOBMFF requirements of parsing the box size of the muli length. This is recommended only as needed for non-complaint CE-devices as needed.

ATSC A/331:2019 - mmt_atsc3_message SLS/Signalling Information support

Overview and Background
In my original Hunt Valley capture from back in December of last year, there is about 5% average packet loss – and MFU has shown pretty strong decoding durability. I'm very close to having robust a/v sync even with the possible loss of MMT signalling information MP_table – which would usually contain the corresponding mpu_timestamp_descriptor for the first sample in the referenced MPU.

Problem Statement
But the next challenge I am trying to determine is the most accurate source of truth of the supported A/341 profile constraints with proper **`picture_rate`** metadata. This will be important for both video timing durability (along with audio sample duration) especially when crossing over a MPU boundary which is missing one or both mpu_timestamp_descriptors.

Additionally, one of the A/331 metadata emissions as a MMT signalling_information message has descriptors to address this challenge, but so far the only message payload I've seen for a while in this descriptor is the **BundleDescriptionMMT, and lacking many of the other A/331:2019 defined descriptor properties.** Please see the attached extract of the MMT SI emission from a Sept. pcap in the Hunt Valley lab. Is there hopefully a missing configuration setting between the ATEME emission and the Enensys Signaller for proper a/v descriptor payloads?

Possible Solution approaches
From my read, it seems like there are 2 different specified mechanisms in A/331 and A/341 for this critical data that we need to are insure in the Encoder to Gateway chain to align with effective MMT emission.

1. A/331:2019 defines an extension into ISO23008-1 MMT signalling_information emission types with a **`mmt_atsc3_message,`** containing the data needed for proper sample a/v timing, but in reviewing recent emissions, I'm yet to see this SI message/descriptor payload detail:

Table 7.7 Bit Stream Syntax for mmt_atsc3_message()

Syntax	No. of Bits	Format
mmt_atsc3_message() {		
message_id	16	uimsbf
version	8	uimsbf
length	32	uimsbf
message payload {		
service_id	16	uimsbf
atsc3_message_content_type	16	uimbsf
atsc3_message_content_version	8	uimbsf
atsc3_message_content_compression	8	uimbsf
URI_length	8	uimbsf
for (i=0;i< URI_length;i++) {		
URI_byte	8	uimsbf
}		
atsc3_message_content_length	32	uimsbf
for (i=0;i<atsc3_message_content_length;i++) {		
atsc3_message_content_byte	8	uimsbf
}		
for (i=0;i<length-11-URI_length-atsc3_message_content_length) {		
reserved	8	uimsbf
}		
}		
}		

Corresponding atsc3_message_content_type descriptors:

Table 7.8 Code Values for atsc3_message_content_type

atsc3_message_content_type	Meaning
0x0000	ATSC Reserved
0x0001	**`UserServiceDescription`** as given in Table 7.6.
0x0002	MPD as given in DASH-IF [12].
0x0003	HELD
0x0004	Application Event Information as given in A/337, Application Signaling [7].
0x0005	Video Stream Properties Descriptor (Sec. 7.2.3.2)
0x0006	ATSC Staggercast Descriptor (Sec. 7.2.3.3)
0x0007	Inband Event Descriptor as given in A/337, Application Signaling [7].
0x0008	Caption Asset Descriptor (Sec. 7.2.3.5)
0x0009	Audio Stream Properties Descriptor (Sec. 7.2.3.4)
0x000A	DWD
0x000B	RSAT as given in A/200, Regional Service Availability [47].
0x000C	Security Properties Descriptor (Sec. 7.2.4.1)
0x000D~0xFFFF	Industry Reserved. See ATSC Code Point Registry [63].

A/331 - Example of the Video Stream properties descriptor:

7.2.3.2.1.4 Picture Rate Information

Table 7.15 Bit Stream Syntax for Picture Rate Information

Syntax	No. of Bits	Format
pr_info(maxSubLayersMinus1) {		
for (i = 0; i <= maxSubLayersMinus1; i++) {		
picture_rate_code[i]	8	uimsbf
if(picture_rate_code[i] == 255) {		
average_picture_rate[i]	16	uimsbf
}		
}		
}		

picture_rate_code[i] – This 8-bit unsigned integer field shall provide information about the picture rate for the i^{th} temporal sub-layer of this video asset. The picture_ rate_code[i] code indicates the following values for picture rate for the i^{th} temporal sub-layer: 0 = unknown, 1 = 23.976 Hz, 2 = 24 Hz, 3 = 29.97 Hz, 4 = 30 Hz, 5 = 59.94 Hz, 6 = 60 Hz, 7 = 25 Hz, 8 = 50 Hz, 9 = 100 Hz, 10 = 120/1.001 Hz, 11 = 120 Hz, 12-254 = ATSC Reserved, 255 = Other. When picture_rate_code[i] is equal to 255 the actual value of picture rate shall be indicated by the average_picture_rate[I] element.

average_picture_rate[i] – This 16-bit unsigned integer field shall indicate the average picture rate, in units of pictures per 256 seconds, of the i^{th} temporal sub-layer. The semantics of avg_pic_rate[0][i] defined in H.265 (10/2014) HEVC specification Section F.7.4.3.1.4 (VPS VUI Semantics) [38] shall apply.

2. A/341:2019 also provides a set of similar “allowed picture rates”, but the including relevant VPS attributes indicating the `pic_rate_present/constant_pic_rate_idc` a are only defined under Section 6.3.1 for “Specific Constraints regarding Spatial Scalable Coding”. For my Android use case, this will probably be acceptable, as the MediaCodec class for OMX decoding requires a translated hvcC payload for CSD Buffer, in which I am already extracting the VPS/SPS/PPS for decoder configuration settings:

A/341:2019 Relevant Requirements:

6.3.1.2 Picture Rate Related Constraints

The following constraints result in a constant picture rate:

- The vps_vui_present_flag in each VPS shall be set equal to 1, pic_rate_present_vps_flag shall be set equal to 1, pic_rate_present_flag[i][j] shall be set equal to 1 and constant_pic_rate_idc[i][j] shall be set equal to 1 for all i, for all j.
- For the ‘layer set’[3] to be carried in the video subsystem of this specification the list of allowed values for avg_pic_rate[i][j] shall be those values that indicate the picture rates defined in Section 6.2.3.1.
- The vui_parameters_present_flag in each SPS shall be set equal to 1, vui_timing_info_present_flag in each SPS shall be set equal to 1, vui_hrd_parameters_present_flag in each SPS shall be set equal to 1, and fixed_pic_rate_general_flag[i] shall be set equal to 1 or fixed_pic_rate_within_cvs_flag [i] shall be set equal to 1 for all values of i in the range 0 to maxNumSubLayersMinus1, inclusive.

Appendix A: Sample MPU for MFU flow analysis

American Version (Target):

```
#mp4dumpv 5761-usa.v

[ftyp] size=8+28
  major_brand = mpuf
  minor_version = 0
  compatible_brand = isom
  compatible_brand = mpuf
  compatible_brand = iso4
  compatible_brand = iso5
  compatible_brand = dash
[mmpu] size=8+29
[moov] size=8+1242
  [mvhd] size=12+96
    timescale = 1000000
    duration = 0
    duration(ms) = 0
  [trak] size=8+623
    [tkhd] size=12+80, flags=1
      enabled = 1
      id = 1
      duration = 0
      volume = 0
      layer = 0
      alternate_group = 0
      matrix_0 = 1.000000
      matrix_1 = 0.000000
      matrix_2 = 0.000000
      matrix_3 = 0.000000
      matrix_4 = 1.000000
      matrix_5 = 0.000000
      matrix_6 = 0.000000
      matrix_7 = 0.000000
      matrix_8 = 16384.000000
      width = 1280.000000
      height = 720.000000
    [edts] size=8+28
      [elst] size=12+16
        entry count = 1
        entry/segment duration = 0
        entry/media time = 3
        entry/media rate = 1
    [mdia] size=8+487
      [mdhd] size=12+20
        timescale = 1000000
        duration = 0
        duration(ms) = 0
        language = und
      [hdlr] size=12+41
        handler_type = vide
        handler_name = Bento4 Video Handler
      [minf] size=8+394
        [vmhd] size=12+8, flags=1
          graphics_mode = 0
          op_color = 0000,0000,0000
        [dinf] size=8+28
          [dref] size=12+16
            [url ] size=12+0, flags=1
              location = [local to file]
        [stbl] size=8+330
          [stsd] size=12+234
            entry-count = 1
            [hev1] size=8+222
              data_reference_index = 1
              width = 1280
              height = 720
              compressor = TitanLive Coding
              [hvcC] size=8+136
                Configuration Version = 1
                Profile Space = 0
                Profile = Main 10
                Tier = 0
                Profile Compatibility = 20000000
                Constraint = b00000000000
```

```
              Level = 120
              Min Spatial Segmentation = 0
              Parallelism Type = 0
              Chroma Format = 1
              Chroma Depth = 10
              Luma Depth = 10
              Average Frame Rate = 15345
              Constant Frame Rate = 1
              Number Of Temporal Layers = 1
              Temporal Id Nested = 0
              NALU Length Size = 4
        [stts] size=12+4
          entry_count = 0
        [stss] size=12+4
          entry_count = 0
        [stsc] size=12+4
          entry_count = 0
        [stsz] size=12+8
          sample_size = 0
          sample_count = 0
        [stco] size=12+4
          entry_count = 0
[trak] size=8+407
  [tkhd] size=12+80, flags=1
    enabled = 1
    id = 2
    duration = 0
    volume = 0
    layer = 0
    alternate_group = 0
    matrix_0 = 1.000000
    matrix_1 = 0.000000
    matrix_2 = 0.000000
    matrix_3 = 0.000000
    matrix_4 = 1.000000
    matrix_5 = 0.000000
    matrix_6 = 0.000000
    matrix_7 = 0.000000
    matrix_8 = 16384.000000
    width = 0.000000
    height = 0.000000
  [tref] size=8+12
    [hint] size=8+4
      track_id_count = 1
      track id  = 2
  [mdia] size=8+287
    [mdhd] size=12+20
      timescale = 1000000
      duration = 0
      duration(ms) = 0
      language = und
    [hdlr] size=12+40
      handler_type = hint
      handler_name = Bento4 Hint Handler
    [minf] size=8+195
      [hmhd] size=12+16
        max_pdu_size = 0
        avg_pdu_size = 0
        max_bitrate = 0
        avg_bitrate = 0
      [dinf] size=8+28
        [dref] size=12+16
          [url ] size=12+0, flags=1
            location = [local to file]
      [stbl] size=8+123
        [stsd] size=12+27
          entry-count = 1
          [mmth] size=8+15
            data_reference_index = 1
        [stts] size=12+4
          entry_count = 0
        [stss] size=12+4
          entry_count = 0
        [stsc] size=12+4
          entry_count = 0
        [stsz] size=12+8
          sample_size = 0
          sample_count = 0
        [stco] size=12+4
          entry_count = 0
```

```
  [mvex] size=8+80
    [trex] size=12+20
      track id = 1
      default sample description index = 1
      default sample duration = 1
      default sample size = 0
      default sample flags = 10000
    [trep] size=8+8
    [trex] size=12+20
      track id = 2
      default sample description index = 1
      default sample duration = 0
      default sample size = 34
      default sample flags = 0
 [mdat] size=8+335332
..
```

(built from anon sample hint track prefix where mpu_fragment_type = 0x02 && mpu_fragmentation_indicator == 0x00 | |0x01))

```
[moof] size=8+1092
  [mfhd] size=12+4
    sequence number = 1
  [traf] size=8+1016
    [tfhd] size=12+4, flags=20000
      track ID = 1
    [tfdt] size=12+8, version=1
      base media decode time = 0
    [trun] size=12+968, flags=f01
      sample count = 60
      data offset = 1108
      entry 0000 = sample_duration:16683, sample_size:166705, sample_flags:2000000,
sample_composition_time_offset:50050
      entry 0001 = sample_duration:16683, sample_size:7983, sample_flags:10000, sample_composition_time_offset:166833
      entry 0002 = sample_duration:16683, sample_size:1391, sample_flags:10000, sample_composition_time_offset:83417
      entry 0003 = sample_duration:16683, sample_size:371, sample_flags:10000, sample_composition_time_offset:33367
..omitted..
      entry 0059 = sample_duration:16683, sample_size:140, sample_flags:10000, sample_composition_time_offset:33367
  [traf] size=8+44
    [tfhd] size=12+12, flags=20018
      track ID = 2
      default sample duration = 1
      default sample size = 34
    [trun] size=12+8, flags=1
      sample count = 60
      data offset = 335716
      entry 0000 =
..omitted..
      entry 0059 =
```

Korean Version (In-Market):

```
sdg-komo-mac148:mpu jjustman$ mp4dumpv 5761.v
[ftyp] size=8+28
  major_brand = mpuf
  minor_version = 0
  compatible_brand = isom
  compatible_brand = mpuf
  compatible_brand = iso4
  compatible_brand = iso5
  compatible_brand = dash
[mmpu] size=8+29
[moov] size=8+1242
  [mvhd] size=12+96
    timescale = 1000000
    duration = 0
    duration(ms) = 0
  [trak] size=8+623
    [tkhd] size=12+80, flags=1
      enabled = 1
      id = 1
      duration = 0
      volume = 0
      layer = 0
      alternate_group = 0
      matrix_0 = 1.000000
      matrix_1 = 0.000000
      matrix_2 = 0.000000
      matrix_3 = 0.000000
      matrix_4 = 1.000000
      matrix_5 = 0.000000
      matrix_6 = 0.000000
      matrix_7 = 0.000000
      matrix_8 = 16384.000000
      width = 1280.000000
      height = 720.000000
    [edts] size=8+28
      [elst] size=12+16
        entry count = 1
        entry/segment duration = 0
        entry/media time = 3
        entry/media rate = 1
    [mdia] size=8+487
      [mdhd] size=12+20
        timescale = 1000000
        duration = 0
        duration(ms) = 0
        language = und
      [hdlr] size=12+41
        handler_type = vide
        handler_name = Bento4 Video Handler
      [minf] size=8+394
        [vmhd] size=12+8, flags=1
          graphics_mode = 0
          op_color = 0000,0000,0000
        [dinf] size=8+28
          [dref] size=12+16
            [url ] size=12+0, flags=1
              location = [local to file]
        [stbl] size=8+330
          [stsd] size=12+234
            entry-count = 1
            [hev1] size=8+222
              data_reference_index = 1
              width = 1280
              height = 720
              compressor = TitanLive Coding
              [hvcC] size=8+136
                Configuration Version = 1
                Profile Space = 0
                Profile = Main 10
                Tier = 0
                Profile Compatibility = 20000000
                Constraint = b00000000000
                Level = 120
                Min Spatial Segmentation = 0
                Parallelism Type = 0
                Chroma Format = 1
                Chroma Depth = 10
```

```
              Luma Depth = 10
              Average Frame Rate = 15345
              Constant Frame Rate = 1
              Number Of Temporal Layers = 1
              Temporal Id Nested = 0
              NALU Length Size = 4
        [stts] size=12+4
          entry_count = 0
        [stss] size=12+4
          entry_count = 0
        [stsc] size=12+4
          entry_count = 0
        [stsz] size=12+8
          sample_size = 0
          sample_count = 0
        [stco] size=12+4
          entry_count = 0
[trak] size=8+407
  [tkhd] size=12+80, flags=1
    enabled = 1
    id = 2
    duration = 0
    volume = 0
    layer = 0
    alternate_group = 0
    matrix_0 = 1.000000
    matrix_1 = 0.000000
    matrix_2 = 0.000000
    matrix_3 = 0.000000
    matrix_4 = 1.000000
    matrix_5 = 0.000000
    matrix_6 = 0.000000
    matrix_7 = 0.000000
    matrix_8 = 16384.000000
    width = 0.000000
    height = 0.000000
  [tref] size=8+12
    [hint] size=8+4
      track_id_count = 1
      track id  = 2
  [mdia] size=8+287
    [mdhd] size=12+20
      timescale = 1000000
      duration = 0
      duration(ms) = 0
      language = und
    [hdlr] size=12+40
      handler_type = hint
      handler_name = Bento4 Hint Handler
    [minf] size=8+195
      [hmhd] size=12+16
        max_pdu_size = 0
        avg_pdu_size = 0
        max_bitrate = 0
        avg_bitrate = 0
      [dinf] size=8+28
        [dref] size=12+16
          [url ] size=12+0, flags=1
            location = [local to file]
      [stbl] size=8+123
        [stsd] size=12+27
          entry-count = 1
          [mmth] size=8+15
            data_reference_index = 1
        [stts] size=12+4
          entry_count = 0
        [stss] size=12+4
          entry_count = 0
        [stsc] size=12+4
          entry_count = 0
        [stsz] size=12+8
          sample_size = 0
          sample_count = 0
        [stco] size=12+4
          entry_count = 0
[mvex] size=8+80
  [trex] size=12+20
    track id = 1
    default sample description index = 1
    default sample duration = 1
```

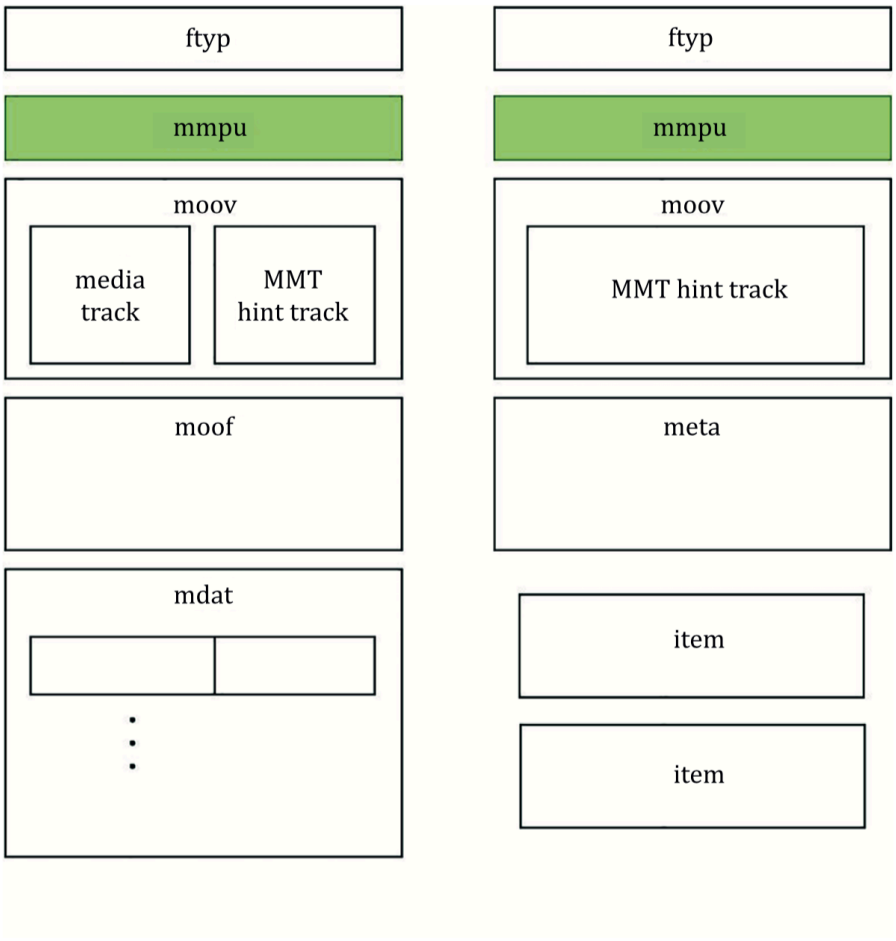
```
      default sample size = 0
      default sample flags = 10000
    [trep] size=8+8
    [trex] size=12+20
      track id = 2
      default sample description index = 1
      default sample duration = 0
      default sample size = 34
      default sample flags = 0
[moof] size=8+1092
  [mfhd] size=12+4
    sequence number = 1
  [traf] size=8+1016
    [tfhd] size=12+4, flags=20000
      track ID = 1
    [tfdt] size=12+8, version=1
      base media decode time = 0
    [trun] size=12+968, flags=f01
      sample count = 60
      data offset = 1108
      entry 0000 = sample_duration:16683, sample_size:166705, sample_flags:2000000,
sample_composition_time_offset:50050
      entry 0001 = sample_duration:16683, sample_size:7983, sample_flags:10000, sample_composition_time_offset:166833
       ...omitted..
      entry 0059 = sample_duration:16683, sample_size:140, sample_flags:10000, sample_composition_time_offset:33367
  [traf] size=8+44
    [tfhd] size=12+12, flags=20018
      track ID = 2
      default sample duration = 1
      default sample size = 34
    [trun] size=12+8, flags=1
      sample count = 60
      data offset = 335716
      entry 0000 =
       ..omitted..
      entry 0058 =
      entry 0059 =
[mdat] size=8+335332
```

Appendix B - Informative vs. Normative Ambiguity
ISO/IEC 23008-1:2017 - 7.1:

"Informative" MPU

movie fragments comprising the MPU. The "moov" box shall contain all codec configuration information for decoding and presentation of media data.

Timed media data are stored as track of the ISOBMFF (a single media track is allowed). Non-timed media are stored as part of metadata in an ISOBMFF. Figure 7 depicts two examples of MMT encapsulation, one for timed and the other for non-timed media. For packetized delivery of MPU, an MMT hint track provides the information to convert encapsulated MPU to MMTP payloads and MMTP packets.



a) MPU with timed media **b) MPU with non-timed media**

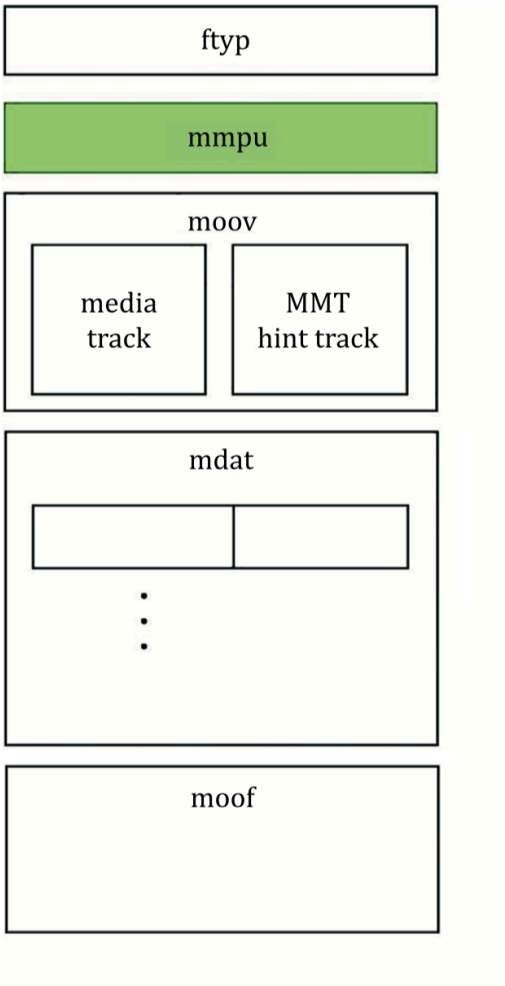
Figure 7 — Examples of MPU encapsulation

NOTE Caption should be "Informative examples of MPU re-constitution"

*Normative MFU Fragmentation - Note, this diagram does not exist in the specification

movie fragments comprising the MPU. The "moov" box shall contain all codec configuration information for decoding and presentation of media data.

Timed media data are stored as track of the ISOBMFF (a single media track is allowed). Non-timed media are stored as part of metadata in an ISOBMFF. Figure 7 depicts two examples of MMT encapsulation, one for timed and the other for non-timed media. For packetized delivery of MPU, an MMT hint track provides the information to convert encapsulated MPU to MMTP payloads and MMTP packets.



*Informative Data type definition

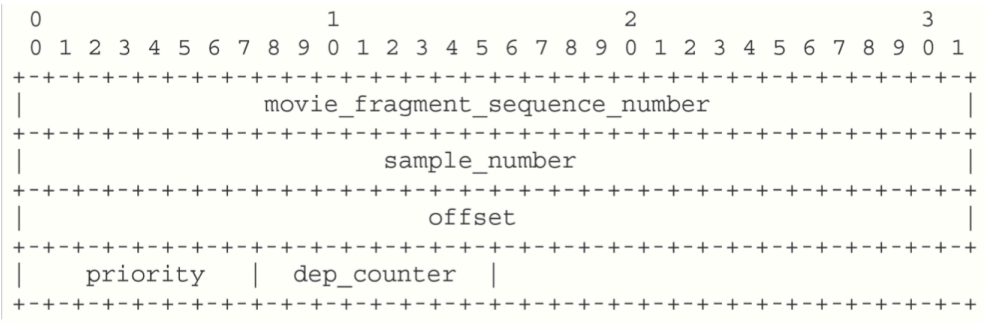


Figure 12 — DU header for timed-media MFU

For non-timed media, the DU header fields are shown in Figure 13.

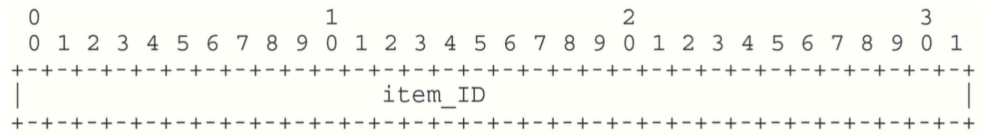


Figure 13 — DU header for non-timed media MFU

9.3.2.3 Semantics

`length (16 bits)` – indicates the length of payload excluding this field in byte.

`MPU Fragment Type (FT: 4 bits)` – this field indicates the fragment type and its valid values are shown in Table 13.

Table 13 — Data type and definition of data unit

FT	Description	Content
0	MPU metadata	contains the ftyp, mmpu, moov, and meta boxes, as well as any other boxes that appear in between.
1	Movie fragment metadata	contains the moof box and the mdat box, excluding all media data inside the mdat box but including any chunks of auxiliary sample information.
2	MFU	contains a sample or subsample of timed media data or an item of non-timed media data.
3 ~ 15	Reserved for private use	reserved

*Normative Data type ordering - Note, this diagram does not exist in the specification

Table 13 — Data type and definition of data unit

FT	Description	Content
0	MPU metadata	contains the ftyp, mmpu, moov, and meta boxes, as well as any other boxes that appear in between.
2	MFU	contains a sample or subsample of timed media data or an item of non-timed media data.
1	Movie fragment metadata	contains the moof box and the mdat box, excluding all media data inside the mdat box but including any chunks of auxiliary sample information.

Receiver Privilege implies Encoder must support MFU emission

9.4.2.2 MMT receiving entity operation

The depacketization procedure is performed at the MMT receiving entity to rebuild the transmitted MPU. The depacketization procedure may operate in one of the following modes, depending on the application needs:

— MPU mode: in the MPU mode, the depacketizer reconstructs the full MPU before forwarding it to the application. This mode is appropriate for non-time critical delivery, i.e. the MPU's presentation time as indicated by the presentation information document is sufficiently behind its delivery time.

— Movie Fragment mode: in the Fragment mode, the depacketizer reconstructs a complete fragment including the fragment metadata and the "`mdat`" box with media samples before forwarding it to the application. This mode does not apply to non-timed media. This mode is suitable for delay-sensitive applications where the delivery time budget is limited but is large enough to recover a complete fragment.

— MFU mode: in the media unit mode, the depacketizer extracts and forwards media units as fast as possible to the application. This mode is applicable for very low delay media applications. In this mode, the recovery of the MPU is not required. The processing of the fragment media data is not required but may be performed to resynchronize. This mode tolerates out of order delivery of the fragment metadata MFUs, which may be generated after the media units are generated. This mode applies to both timed and non-timed media.

Using the MFU sequence numbers, it is relatively easy for the receiver to detect missing packets and apply any error correction procedures such as ARQ to recover the missing packets. The payload type may be used by the MMT sending entity to determine the importance of the payload for the application and to apply appropriate error resilience measures.

Document Updates:

2019-05-22 - Added signal_data_size as attribute in `mjgp box.`

2019-03-25 - Added PTS_timestamp in `mjae_scte35_binary_payload and` SCTE35_Signal_Descriptor message for clock correlation between MPEGTS and NTP

2019-08-08 - Updated pseduo-code translation from MJGP box to SCTE35_Signal_Message and collection of SCTE35_Signal_Descriptor

2019-08-14 - Addition of business use cases for ATSC 3.0 Enhanced Content Monetization.

2019-08-20 - Adding optional removal of mjsd and mjgp boxes from packager once MMT SI/A337 message emission is processed

2019-08-29 - Additional detail regarding assumptions of ntp_timestamp and pts_timestamp expectations between encoder, packager and signaler

2019-10-18 - Renamed signal_data_size to signal_data_length, added additional atsc3_message_content_type and A/331:2019 integration flows to address A/337 and A/334 applicaiton runtime requirements for SCTE35 Message Parsing

Open Items:

Validating end-to-end flow and A/337 signalling reception

Developing A/334 application to validate technology use cases

Proposal of RP to the TG3 committee, with the inclusion of A/344 "Core" application for AdTech use-cases.