ATSC Standard:
A/331:2023-03 Amendment No. 1, “MMT DRM”

Doc. A/331:2023-03 Amend. No. 1
16 August 2023

Advanced Television Systems Committee
1300 I Street, N.W., Suite 400E
Washington, D.C. 20005
202-872-9160

The Advanced Television Systems Committee, Inc. is an international, non-profit organization developing voluntary standards and recommended practices for broadcast television and multimedia data distribution. ATSC member organizations represent the broadcast, professional equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on Inter-society Coordination (JCIC): the Consumer Technology Association (CTA), the Institute of Electrical and Electronics Engineers (IEEE), the National Association of Broadcasters (NAB), the Internet & Television Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). For more information visit www.atsc.org.

Note: The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder.

Implementers with feedback, comments, or potential bug reports relating to this document may contact ATSC at https://www.atsc.org/feedback/.

Revision History

Version	Date
Amendment approved	16 August 2023

ATSC Standard:
A/331:2023-03 Amendment No. 1, "MMT DRM"

1. OVERVIEW

1.1 Definition

An Amendment is generated to document an enhancement, an addition or a deletion of functionality to previously agreed technical provisions in an existing ATSC document. Amendments shall be published as attachments to the original ATSC document. Distribution by ATSC of existing documents shall include any approved Amendments.

1.2 Scope

This document specifies updated signaling in order to enable DRM over MMT, including out-of-order delivery, which is intended to decrease latency. This amendment is in response to New Project Proposal N-045r0, "Modify A/331 to clarify DRM (Digital Rights Management) operation for MMT."

Note that there is a related NPP being pursued in TG3/S36 regarding an amendment to A/362. This amendment references amended text expected in A/362.

1.3 Rationale for Changes

The changes described in this document are being proposed in order to enable DRM over MMT, including out-of-order delivery, which is intended to decrease latency.

1.4 Compatibility Considerations

The changes described in this document are backward-compatible relative to the currently published version of the standard to which this Amendment pertains and any previously approved Amendments for that standard; however currently deployed receivers would not have the capability of decrypting content delivered with this method. Such receivers would need updates if they are to decrypt such content. These changes would not affect any receiver's ability to present unencrypted content or content delivered using ROUTE-DASH.

2. LIST OF CHANGES

Change instructions are given below in *italics*. Unless otherwise noted, inserted text, tables, and drawings are shown in blue; deletions of existing text are shown in ~~red strikeout~~. The text "[ref]" indicates that a cross reference to a cited referenced document should be inserted. Yellow highlights indicate intended cross-references and other information that is expected to be updated prior to publication within a new revision.

2.1 Normative References

Add the following normative reference(s)

[w] DASH-IF: "Identifiers / Content Protection", DASH Industry Forum. https://dashif.org/identifiers/content_protection/

[x] DASH-IF: IOP V5, Part 6 "Content Protection" DASH Industry Forum. https://dashif.org/guidelines/iop-v5#part-6-content-protection-and-security.

[y] IETF: RFC 4648, "The Base16, Base32, and Base64 Data Encodings," Internet Engineering Task Force, Reston, VA, October 2006.
https://www.rfc-editor.org/info/rfc4648.

2.2 Informative References

Add the following informative reference(s)

[z] ATSC: "ATSC Recommended Practice: Digital Rights Management (DRM)," Doc. A/362:2023-02, Advanced Television Systems Committee, Washington, DC, 23 February 2023.

2.3 Acronyms and Abbreviations

No changes.

2.4 Terms

No changes.

2.5 Change Instructions

Replace A/331 Section 7.2.4 with the following text. For easy readability, the changes are not shown in blue.

7.2.4 MMT DRM Signaling

7.2.4.1 General

Content protection in MMT relies on ISO/IEC23001-7 Common encryption in ISO base media file format files [45]. Protection system specific and proprietary signaling information are delivered in two ways:

- by the MMT signaling messages, and
- carried in MPU in designated metadata boxes defined by ISO BMFF

Additional information is available in ATSC A/362 [z].

7.2.4.1.1 Service Signaling

When a Service is encrypted, this is signaled in the SLT by the Service@protected attribute and the Service@drmSystemID attribute as specified in Section 6.3.1. For MMT, this is signaled when the ComponentInfo@componentProtectedFlag of at least one component listed in BundleDescriptionMMT is set to true (See Section 7.2.1).

7.2.4.1.2 Use of the Content Protection Scheme

The scheme_code field of the security_properties_descriptor shall indicate the encryption scheme as specified by A/360, Section 5.7 [10]. The "`urn:mpeg:cenc:2013`" extension namespace is assumed for this field. The default_KID signaled in security_properties_descriptor for each component may be sufficient for the receiver to acquire a DRM license, or identify a previously acquired license that can be used to decrypt the component.

As the default_KID is signaled by security_properties_descriptor for each component, it allows a player to determine if a new license needs to be acquired for each component by comparing their default_KID attributes with each other, and with the default_KID attributes of stored licenses. A player can simply compare these KID strings and determine what unique licenses are necessary without interpreting license information specific to each DRM system.

7.2.4.1.3 Signaling of Information from Protection System Specific Header Box

A 'pssh' box (as specified in ISO/IEC 23001-7 [45]) is used by each DRM system with its registered SystemID, and is nominally stored in the movie box ('moov') and additionally may be present in the movie fragment box ('moof'). The information of the box shall be identical to that of the pssh data of the security_properties_descriptor carried in the MMT ATSC 3.0 Signaling Message by the broadcast server. As specified in Section [6.3.2], initialization segments do not contain any pssh boxes as per DASH-IF IOP V5, Part 6, 6.3 [x] and Receivers can ignore such boxes when encountered.

7.2.4.1.4 Signaling of License Acquisition URL by the MMT Signaling Messages

Information to access a server for the ATSC receive to acquire license, the type of license and the URL to access the server for license acquisition may be added to the security_properties_descriptor carried in the MMT ATSC 3.0 Signaling Message by the broadcast server. More than one of following methods to acquire a valid license may be signaled.

- `license-1.0`. Direct license acquisition by ATSC3 receivers and the URI scheme is a valid endpoint for access.
- **`groupLicense-1.0.`** Provides a path for a group-based license. The URI may need to be parsed specific to that DRM System by the ATSC3 Receiver to access any local group licenses.
- **`contentId-1.0.`** Provides information for the DRM specific content identifier used to generate the KIDs. The URI should be parsed to extract relevant information using REST based notation.

7.2.4.1.5 Additional Signaling for Out-of-Order Delivery

In case of out of order delivery is used, i.e., media data from 'mdat' box is delivered before other boxes are delivered, encryption metadata is delivered as a signaling message so that the ATSC3 Receiver can start decryption of media data immediately without any further delay. The receiver uses the signaling message to acquire the information about the number of bytes of clear data and protected data of each subsamples of each sample, and the count of the encrypted blocks and the unencrypted blocks in the protection pattern for each sample groups are before it receives 'moof' box or 'moov' box.

The low_delay_decryption_information_descriptor() is carried by the MMT ATSC 3.0 Signaling Message, mmt_atsc3_message(), contains encryption metadata for a specific sample. The descriptor uses MPU_sequence_number, movie_fragment_sequence_number and sample_number to identify the specific sample the information in the descriptor is applied to.

7.2.4.2 MMT Signaling Message Support for Encryption and DRM Signaling

7.2.4.2.1 Descriptors

7.2.4.2.1.1 Security Properties Descriptor

For delivery of DRM-related information to prepare the receiver before the content is delivered, some information is sent as MMT signaling messages in addition to the ISOBMFF defined metadata boxes of the MPUs. A Security Properties Descriptor security_properties_descriptor() is carried by the MMT ATSC 3.0 Signaling Message, mmt_atsc3_message(), containing information for CENC and standardized license acquisition information. The syntax of security_properties_descriptor() shall be as provided in Table 7.34.

Table 7.1 Bit Stream Syntax for Security Properties Descriptor

Syntax	No. of Bits	Format
security_properties_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
number_of_assets	8	uimsbf
for (i=0; i<number_of_assets; i++) {		
asset_id_length	32	uimsbf
for (j=0; j<asset_id_length; j++) {		
asset_id_byte	8	uimsbf
}		
scheme_code_present	1	bslbf
default_KID_present	1	bslbf
reserved	6	‘111111’
if (scheme_code_present) {		
scheme_code	4*8	uimsbf
}		
if (default_KID_present) {		
default_KID	16*8	uimsbf
}		
number_of_systems	8	uimsbf
for (j=0; j<number_of_systems; j++) {		
system_UUID_present	1	bslbf
license_info_present	1	bslbf
pssh_present	1	bslbf
reserved	5	‘11111’
if (system_UUID_present) {		
system_UUID	16*8	uimsbf
}		
if (license_info_present) {		
number_of_license_info	8	uimsbf
for (k=0; k<number_of_license_info; k++) {		
license_type	8	uimsbf
LA_URL_length	8	uimsbf
for (m=0; m<LA_URL_length; m++) {		
LA_URL_byte	8	bslbf
}		
}		
}		
if (pssh_present) {		
pssh_length	4*8	uimsbf
for (k=0; k<pssh_length; k++) {		
pssh_byte	8	bslbf
}		
}		
}		
}		
}		

descriptor_tag – A 16-bit unsigned integer field that shall have the value 0x000C, identifying this descriptor as the security_properties_descriptor().

descriptor_length – A 16-bit unsigned integer field that shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

number_of_assets – An 8-bit unsigned integer field that shall specify the number of DRM protected assets described by this descriptor.

asset_id_length – A 32-bit unsigned integer field that shall specify the length in bytes of the DRM protected asset id.

asset_id_byte – An 8-bit unsigned integer field that shall contain a byte of the DRM protected asset id.

scheme_code_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the element scheme_code is present. When set to '0', the flag shall indicate that the element scheme_code is not present.

default_KID_present – A 1-bit Boolean flag that shall indicate, when set to '1', that default KID for this DRM protected asset is present. When set to '0', this flag shall indicate that no default KID for this DRM protected asset is present.

scheme_code – A 32-bit unsigned integer field that shall specify a 4-character code for a protection scheme. The value of these four characters shall be as specified in [45].

default_KID_length – An 8-bit unsigned integer field that shall specify the length in bytes of the default KID for this DRM protected asset. The value of this field shall be set to 16.

default_KID_byte – An 8-bit unsigned integer field that shall contain a byte of the default KID.

number_of_systems – An 8-bit unsigned integer field that shall specify the number of protection systems represented.

system_UUID_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the protection system UUID is present. When set to '0', the flag shall indicate that the protection system UUID information is not present and the protection system UUID is not specified.

license_info_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the URL information for direct license server access is present.

pssh_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the pssh box data is present.

system_UUID – Sixteen unsigned integer bytes that shall contain the UUID of the protection system as specified in DASH-IF Identifiers/ Content Protection [w].

number_of_license_info – An 8-bit unsigned integer field that shall specify the number of license information blocks signaled for the current asset.

license_type – An 8-bit unsigned integer field that shall specify a type of applicable license as specified in Table L.

LA_URL_length – An 8-bit unsigned integer field that shall specify the length in bytes of the LA_URL (the URL for license acquisition).

LA_URL_byte – An 8-bit field that shall contain a byte of the LA_URL. The LA_URL shall be a dashif:Laurl as specified by DASH-IF IOP V5 [x].

pssh_length – A 32-bit unsigned integer field that shall specify the length in bytes of the pssh.

pssh_byte – An 8-bit field that shall contain a byte of the encoded pssh. The pssh shall be encoded as base64 as specified in RFC 4648 [y], Section 4.

Table L. Code Values for license_type

license_type	Meaning
0x00	ATSC Reserved
0x01	**license-1.0,** Direct license acquisition by ATSC3 receivers and the URI scheme is a valid endpoint for access.
0x02	**groupLicense-1.0,** Provides a path for a group-based license. The URI may need to be parsed specific to that DRM System by the ATSC3 Receiver to access any local group licenses.
0x03	**contentId-1.0,** Provides information for the DRM specific content identifier used to generate the KIDs. The URI should be parsed to extract relevant information using REST based notation.
0x04~0xFF	Industry Reserved. See ATSC Code Point Registry.

7.2.4.2.1.2 Low Delay Decryption Information Descriptor

This message supports two modes of operation, MPU-based operation mode and sample-based operation mode, which is distinguished by operation_mode. In MPU-based operation mode, descriptor contains information about all samples in an MPU with a single descriptor. CencSampleEncryptionInformationGroupEntry sample group description structures for all sample groups defined in MPU is provided by a single descriptor in this mode. In addition, a list of samples associated to each sample groups and CencSampleAuxiliaryDataFormat for all of them are also provided by a single descriptor in MPEG-based operation mode. In sample-based operation mode, one descriptor provides information about a single sample. A descriptor carries SampleEncryptionBox associated with this sample. In addition, the descriptor in this mode carries CencSampleEncryptionInformationGroupEntry sample group description structures for the sample group the sample this descriptor is associated with, whenever needed. If the sample group description structure for the same sample group has been already delivered, it can be skipped.

Syntax	No. of Bits	Format
low_delay_decryption_information_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
MPU_sequence_number	32	uimsbf
operation_mode	3	uimsbf
sample_group_info_present	1	uimsbf
reserved	4	1111
if (**operation_mode** == 1) {		
movie_fragment_sequence_number	32	uimsbf
sample_ number	8	uimsbf
sample_group_number	8	uimsbf
if (**sample_group_info_present** ==1) {		
size_of_seig_box	32	uimsbf
for (k=0; k<bytes_of_seig_box; k++) {		
seig_box_byte	8	uimsbf
}		
size_of_senc_box	32	uimsbf
for (n=0; n<size_of_senc_info; n++) {		
senc_box_byte	8	uimsbf

}		
}		
if (**operation_mode** == 2) {		
number_of_movie_fragments	8	uimsbf
for (i=0; i<number_of_movie_fragments; i++) {		
movie_fragment_sequence_number	32	uimsbf
number_of_sample_groups	8	uimsbf
for (j=0; j<number_of_sample_groups; j++) {		
size_of_seig_box	32	uimsbf
for (k=0; k<bytes_of_seig_box; k++) {		
seig_box_byte	8	uimsbf
}		
number_of_samples	8	uimsbf
for (l=0; l<number_of_samples; l++) {		
sample_ number	8	uimsbf
size_of_aux_info	32	uimsbf
for (m=0; m<size_of_aux_info; m++) {		
aux_info_byte	8	uimsbf
}		
}		
}		
}		
}		

descriptor_tag – A 16-bit unsigned integer field that shall have the value 0x000E, identifying this descriptor as the low_delay_decryption_descriptor().

descriptor_length – A 16-bit unsigned integer field that shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

MPU_sequence_number – An 32-bit unsigned integer field that shall specify sequence number of the MPU this descriptor is applied to.

operation_mode – An 3-bit unsigned integer field that shall specify the mode of operation this descriptor is used for as specified in Table M.

Table M. Code Values for operation_mode

license_type	Meaning
0	ATSC Reserved
1	Sample-based operation mode. The descriptor carries decryption information for the sample identified by the combination of MPU_sequence_number, movie_fragment_sequence_number and sample_number.
2	MPU-based operation mode. The descriptor carries decryption information for all samples of the MPU identified by MPU_sequence_number.
3~7	Industry Reserved. See ATSC Code Point Registry.

sample_group_info_present – A 1-bit Boolean flag that shall indicate whether CencSampleEncryptionInformationGroupEntry is carried or not. When set to ‘1’, that CencSampleEncryptionInformationGroupEntry is present.

movie_fragement_sequence_number – A 32-bit unsigned integer field that shall specify the sequence number of the movie fragment to which the information is applied.

sample_number – A 32-bit unsigned integer field that shall specify the sequence number of a sample within a movie fragment to which the information is applied.

size_of_seig_box – A 32-bit unsigned integer field that shall specify the size of CencSampleEncryptionInformationGroupEntry box for a sample group. When the operation_mode is equal to '1', this field will indicate the size of CencSampleEncryptionInformationGroupEntry box for a sample group the sample this descriptor is associated with. When the operation_mode is equal to '2', this field will indicate the size of CencSampleEncryptionInformationGroupEntry box for a sample group whose group_description_index is equal to j.

seig_box_byte – An 8-bit unsigned integer field that shall contain the kth byte of the CencSampleEncryptionInformationGroupEntry box for a sample group. When the operation_mode is equal to '1', this field will carry the kth byte of CencSampleEncryptionInformationGroupEntry box for a sample group the sample this descriptor is associated with. When the operation_mode is equal to '2', this field will carry the kth byte of CencSampleEncryptionInformationGroupEntry box for a sample group whose group_description_index is equal to j.

size_of_senc_box – A 32-bit unsigned integer field that shall specify the size of SampleEncryptionBox for the sample with which this descriptor is associated.

senc_box_byte – An 8-bit unsigned integer field that shall contain the *n*th byte of the SampleEncryptionBox box for the sample with which this descriptor is associated.

number_of_movie_fragments – An 8-bit unsigned integer field that shall specify the number of movie fragments contained in the MPU with which this descriptor is applied.

number_of_sample_groups – An 8-bit unsigned integer field that shall specify the number of sample groups defined in the movie fragment information in the current loop.

number_of_samples – An 8-bit unsigned integer field that shall specify the number of samples that belong to the sample group whose group_description_index is equal to j.

size_of_aux_info – A 32-bit unsigned integer field that shall specify the size of the CencSampleAuxiliaryDataFormat data structure applied to the sample currently being described.

aux_info_byte – An 8-bit unsigned integer field that shall contain the *m*th byte of the CencSampleAuxiliaryDataFormat.

Insert the following section immediately after the section above and prior to the existing Section 7.2.5 in A/331.

7.2.4.2.2 MMT Hint Sample for Low Delay Decryption

7.2.4.2.2.1 General

When out of order delivery mode is used, i.e., media data from 'mdat' box is delivered before other boxes are delivered, metadata required for decryption of that media data is delivered as hint samples in order to reduce the number of signaling messages to be processed. The sender copies the information for decryption to LowDelayProcessingInfo for each hint sample from the SampleEncryptionBox of the corresponding media samples.

The semantics of each field of LowDelayProcessingInfo shall be identical to the fields of the SampleEncryptionBox with the identical name. As the LowDelayProcessingInfo only contains the information for a single sample, the field about the number of samples of the SampleEncryptionBox is not copied.

The sender may deliver such LowDelayProcessingInfo together with media samples. The receiver may use such LowDelayProcessingInfo instead of SampleEncryptionBox for decryption of the media sample without waiting for a 'moof' box or 'moov' box containing a SampleEncryptionBox to be delivered. Note that a complete SampleEncryptionBox corresponding the media samples of a movie fragment is not replaced by LowDelayProcessingInfo and is delivered as a part of a 'moof' box, regardless of the use of LowDelayProcessingInfo.

7.2.4.2.2.2 Sample Description Format

A sample entry type of 'mtha' shall be used for MMT hint tracks. This sample entry type shall include an indication of the version and includes reserved bits for the introduction of new flags for future extensions.

7.2.4.2.2.2.1 Syntax

Note that the following syntax follows the MPEG style guide.

```
aligned(8) class MMTHintATSC3SampleEntry() extends SampleEntry('mtha') {
   unsigned int(16) hinttrackversion = 1;
   unsigned int(16) highestcompatibleversion = 1;
   unsigned int(1) has_mfus_flag;
   unsigned int(1) is_timed;
   unsigned int(1) decryption_info_flag;
   unsigned int(13) reserved;
}
```

7.2.4.2.2.2.2 Semantics

`hinttrackversion` – Specifies the version of this hint track. The current version is 1.

`highestcompatibleversion` – Specifies the oldest version with which this hint track is backward-compatible.

`has_mfus_flag` – A flag indicating whether the MPUs are fragmented into MFUs or not. If this flag is set to FALSE, the hint track applies to the complete MPU, i.e., each track fragment will have a single sample. Otherwise, each hint sample applies to an MFU.

`is_timed` – Indicates whether the media data hinted by this track is timed data or non-timed data.

`low_delay_decryption_info_flag` – Indicates whether the low delay decryption information of the media data hinted by this track is included in the hint samples or not. If the value of this field is set to '1', then the information is included in the samples. If the value of this field is set to '0', then the information is not included in the samples.

7.2.4.2.2.3 Sample Format

The sample structure of MMT hint track for ATSC 3 varies by the value of `hinttrackversion` field set by the sample entry box. According to the value of `hinttrackversion` field, value of the different set of flags set by the sample entry box are evaluated and additional data structure are included in the end of the hint sample.

7.2.4.2.2.3.1 Syntax

```
aligned(8) class MMTHSampleATSC3 extends Full Box ('mhas', version, 0)
{
   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;
   } else {
      unsigned int(16) item_ID;
   }
   if(version == 1){
      if(low_delay_decryption_info_flag){
         unsigned int(24) SampleEnctryptionBoxflags;
         LowDelayDecryptionInfo()
      }
   }
}
```

7.2.4.2.2.3.2 Semantics

`hinttrackversion` – Contains the value set by the sample entry this sample belongs to.

`sequence_number` – An integer number that indicates the sequencing order of this MFU within the MPU. Discontinuity of sequence numbers in an MPU is allowed to indicate that certain MFUs (whose sequence number was not in the sequence) were not processed after packetization of MPU. Examples of MFU processing are delivery and caching by underlying network entity.

`movie_fragment_sequence_number` – The sequence number of the movie fragment to which the media data of this MFU belongs (see ISO/IEC 14496-12 [35], Section 8.8.5). The `movie_fragment_sequence_number` in an MPU shall start by "1" for the first movie fragment of the MPU and shall be incremented by "1" for every succeeding movie fragment in that MPU.

`trackrefindex` – The ID of the media track from which the MFU data is extracted.

`samplenumber` – The number of the sample from which this MFU is extracted. Sample number `n` points to the *n*th sample from accumulated samples of the current movie fragment. The sample number of the first sample in the movie fragment is set to "1" (see ISO/IEC 14496-12 [35], Section 8.8.8).

`priority` – Indicates the priority of the MFU relative to other MFUs within an MPU.

`dependency_counter` – Indicates the number of MFUs whose decoding is dependent on this MFU. The value of this field is equal to the number of subsequent MFUs in the order of `sequence_number` that may not be correctly decoded without this MFU. For example, if the value of this field is equal to `n`, then `n` subsequent MFUs may not be correctly decoded without this MFU.

`offset` – The offset of the media data contained in this MFU. The offset base is the beginning of the containing "`mdat`" box.

`length` – The length of the data corresponding to this MFU in bytes.

`item_ID` – For non-timed media data, this is the ID of the item that is contained in this MFU.

`hinttrackversion` – The version of this hint track indicated by the sample entry of this track.

`SamepleEncryptionBoxflags` - Carries the same value of the flags field of `SampleEncryptionBox` to which this sample belongs.

`low_delay_decryption_info_flag` – Contains the value set by the sample entry this sample belongs to.

7.2.4.2.2.4 Low Delay Decryption Info

7.2.4.2.2.4.1 Syntax

```
aligned(8) class LowDelayDecryptionInfo()
{

   unsigned int(8) Per_Sample_IV_Size
   unsigned int(Per_Sample_IV_Size*8)  InitializationVector;
   if (flags & 0x000002)
   {
      unsigned int(16)  subsample_count;
      {
         unsigned int(16)  BytesOfClearData;
         unsigned int(32)  BytesOfProtectedData;
      }[ subsample_count ]
   }
}
```

7.2.4.2.2.4.2 Semantics

`Per_Sample_IV_Size` – Same as the semantics of `Per_Sample_IV_Size` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `Per_Sample_IV_Size` field of `SampleEncryptionBox` corresponding the same sample.

`InitializationVector` – Same as the semantics of `InitializationVector` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `InitializationVector` field of `SampleEncryptionBox` corresponding the same sample.

`subsample_count` – Same as the semantics of `subsample_count` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `subsample_count` field of `SampleEncryptionBox` corresponding the same sample.

`BytesOfClearData` – Same as the semantics of `BytesOfClearData` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `BytesOfClearData` field of `SampleEncryptionBox` corresponding the same sample.

`BytesOfProtectedData` – Same as the semantics of `BytesOfClearData` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `BytesOfProtectedData` field of `SampleEncryptionBox` corresponding the same sample.

7.2.4.2.3 MPU Fragment Type for MMT Hint Sample for ATSC 3

As hint samples are not an independent media but an integral part of an MPU, they shall be delivered in the same MPU as the media data they are describing, which means that they shall be delivered by the MMTP packets whose value of the packet_id field is the same as that of the MMTP packet delivering media data they are describing. The hint samples shall be delivered immediately before their associated MFUs.

An MPU is delivered with the MMTP packets the value of whose packet_id field of the MMTP packet header is same. When an MPU is delivered, the MPU Fragment Type, FT, field of the MMTP payload header is used to distinguish the types of data from a MPU. To differentiate the packets containing hint sample data from the packets containing media data new fragment type value is defined for hint sample as shown in Table AA.

Table AA 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	MMT hint sample for ATSC 3	Contains a sample data of MMT hint sample for ATSC 3 type describing media data delivered by the MMT packets with same packet_id
4 ~ 15	Reserved for private use	reserved

The DU header for timed media is used for the hint samples. The value of the fields of the DU header for hint sample are set to the same value of the DU header for media sample which is described by that hint sample except the offset field.

Change Section 6.3.2 as follows:

6.3.2 SLT Semantics

…

`@drmSystemID` – This attribute identifies one or more DRM systems related to this service as a space-separated list of xs:anyURI. When `@serviceCategory` is "6" (DRM Data Service), this attribute is required and shall contain a single URI. This attribute shall be set to a URI identifying the DRM system ID associated with this DRM Data Service. The URI shall contain a UUID formatted as a URN, the same as the value of the `@schemeIdUri` used for the DASH MPD **ContentProtection** Descriptor, per Section 7.5 of DASH-IF [12]. The "`urn:uuid:`" prefix shall be included. More than one DRM Data Service may appear in a given SLT. The same value in the `@drmSystemID` attribute may be included in multiple instances of a Service identified as an DRM Data Service. When `@serviceCategory` is other than "6", this attribute is optional. When present, it shall contain one or more URIs. These URIs have the same syntax and semantics as for `@serviceCategory`="6" except that they identify one or more DRM systems that can decode the service. Receivers can, in advance of an MPD, initialize a supported DRM system. Receivers not supporting any of the DRM systems listed can choose not to offer the service to the viewer. Contrary to DASH-IF ATSC 1.1 [12], initialization segments should not contain any pssh boxes as per DASH-IF IOP V5, Part 6, 6.3 [x]) and Receivers may ignore such boxes when encountered.

– End of Document –