INTERNATIONAL STANDARD

ISO/IEC 23001-12

Second edition
2018-12

Information technology — MPEG systems technologies —

Part 12: Sample variants

Technologies de l'information — Technologies des systèmes MPEG —
Partie 12: Variantes d'échantillon

Reference number
ISO/IEC 23001-12:2018(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 23001-12:2015), which has been technically revised.

The main changes compared to the previous edition are as follows:

— support for using sample variants for multiple alternate samples;

— support for using sample variants for multiple alternate protection schemes;

— support for carriage of sample variants in MPEG-2 transport streams.

A list of all parts in the ISO/IEC 23001 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Information technology — MPEG systems technologies —

Part 12:
Sample variants

1 Scope

This document defines sample variants and their carriage in the ISO base media file format (ISO/IEC 14496-12) and MPEG-2 transport stream (ISO/IEC 13818-1).

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 13818-1:2018, *Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems*

ISO/IEC 14496-12:2015, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 23001-7, *Information technology — MPEG systems technologies — Part 7: Common encryption in ISO base media file format files*

ISO/IEC 23001-9, *Information technology — MPEG systems technologies — Part 9: Common encryption of MPEG-2 transport streams*

3 Terms and definitions

For the purpose of this document, the terms and definitions given in ISO/IEC 13818-1, ISO/IEC 14496-12, and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
double encrypted
encrypted first by a *media key* (3.3) (as part of the encryption of the complete sample variant) and then by a variant byte range key

Note 1 to entry: See 7.2.

3.2
media data stream
track or packetized elementary stream containing audio-visual content

Note 1 to entry: Track is as specified in ISO/IEC 14496-12.

Note 2 to entry: Packetized elementary stream is as specified in ISO/IEC 13818-1.

3.3
media key
encryption key associated with one or more media *samples* (3.5)

3.4
media KID
encryption KID associated with one or more media *samples* (3.5)

3.5
sample
data of a sample or of an access unit

Note 1 to entry: Data of a sample is as specified in ISO/IEC 14496-12.

Note 2 to entry: Data of an access unit is as specified in ISO/IEC 13818-1.

3.6
sample variant
assembled media *sample* (3.5) replacing an original sample

3.7
sample variants media data stream Extractor
logical module that performs the steps that implement the process of generating a complete compliant *media data stream* (3.2) composed of *sample variants* (3.6)

3.8
variant byte range
location of a sequence of bytes that can constitute a portion of a *sample variant* (3.6)

3.9
variant constructor
sample variant (3.6) metadata that defines how to assemble an individual sample variant

3.10
variant data stream
track or packetized elementary stream for variant data

Note 1 to entry: Track is as specified in ISO/IEC 14496-12.

Note 2 to entry: Packetized elementary stream is as specified in ISO/IEC 13818-1.

3.11
variant media data
media data used to construct a *sample variant* (3.6)

Note 1 to entry: Some of the media data can come from the original media data stream.

3.12
variant processor
logical module that implements the process of assembling *sample variants* (3.6)

4 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

CENC Common Encryption (as specified in ISO/IEC 23001-7)

CETS Common Encryption of MPEG-2 Transport Streams (as specified in ISO/IEC 23001-9)

DRM Digital Rights Management

ISOBMFF ISO Base Media File Format (as specified in ISO/IEC 14496-12)

IV Initialization Vector

KID Key Identifier

MPEG-2 TS MPEG-2 Transport Stream (as specified in ISO/IEC 13818-1)

5 Overview

This document defines a framework for the carriage of sample variants in the ISOBMFF and MPEG-2 TS. Sample variants are alternative samples which can be used to replace the original samples in the media data stream. Sample variants can be used, for example, to provide forensic information in the rendered sample that can identify the DRM client or to provide appropriately encrypted samples with multiple protection schemes. This variant framework is fully compatible with ISOBMFF and CENC for carriage in ISOBMFF or fully compatible with MPEG-2 TS and CETS for carriage in MPEG-2 TS. The variant framework is agnostic to any particular forensic marking system or DRM system used.

The sample variant framework uses three core constructs to define and carry sample variant data: variant constructors, variant byte ranges and variant media data.

NOTE 1 The variant process model described in Clause 12 can also assist in introducing the concepts.

Figure 1 shows a scenario where a sample (sample 2) has a number of sample variants. The figure shows 3 samples in a series from left to right, the middle of which has variants. The top row is a conceptual depiction of what is encoded using ISOBMFF or MPEG-2 TS and the bottom row shows what is output after sample variant processing. Access to samples is under the control of KIDs as depicted in the top row of Figure 1. For sample variants, a hierarchy of KIDs is used to provide access to data, with the higher level KIDs providing access to sample variant metadata and the lower level KIDs providing access to media data.

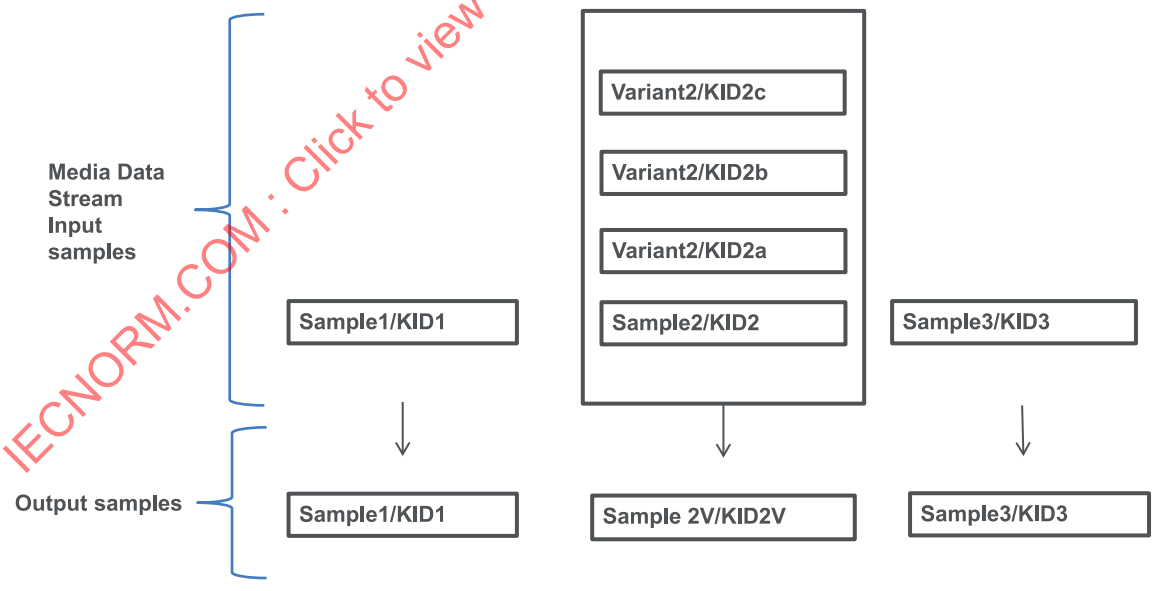

Figure 1 — Example sample variant structure for multiple alternate samples

Figure 2 shows another scenario in which a sample (sample 2) has a number of sample variants. In this case, however, only the data that differs from the original sample is carried in the samples of the variant data stream. As in Figure 1, the top row is a conceptual depiction of what is encoded using ISOBMFF or MPEG-2 TS and the bottom row shows what is output after sample variant processing, with access to samples controlled via KIDs as depicted in the top row. Under some use cases, such as subsample pattern encryption, the amount of redundant data between an original sample and a corresponding

sample variant may be relatively large. Sample variants can reference byte ranges of the original media data stream in addition to those of the current variant data stream, as well as additional variant data streams. This can enable more efficient carriage of sample variants than if sample variants had to be encoded in their entirety.

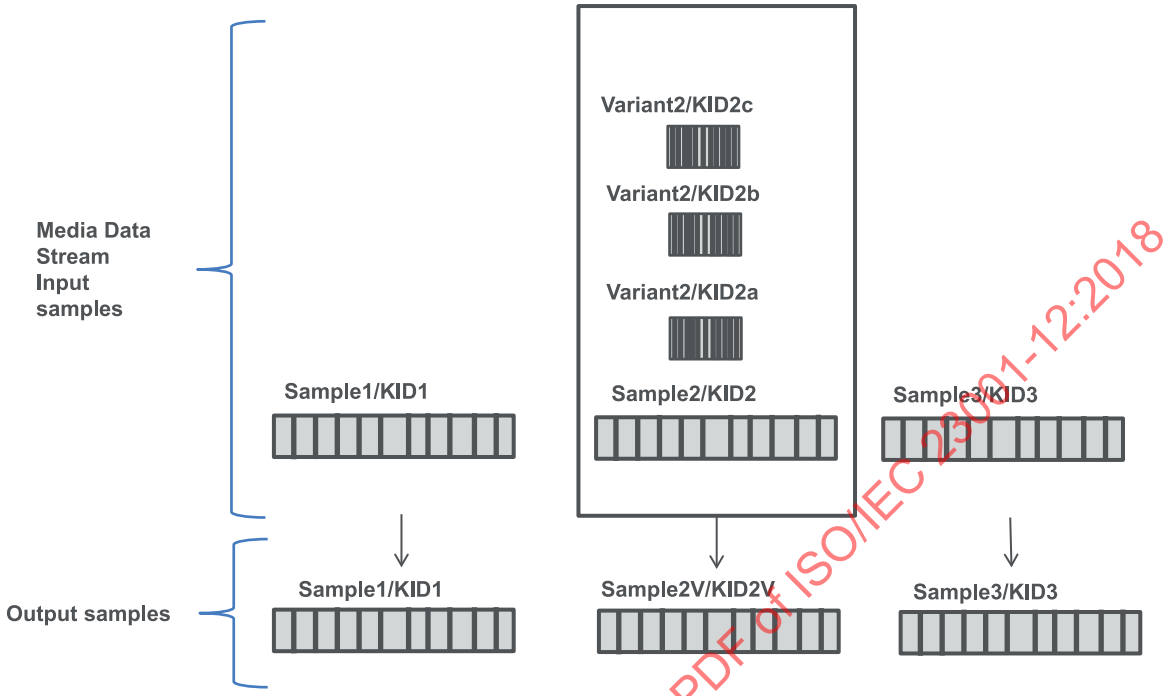

Figure 2 — Example sample variant byte range encoding for multiple alternate samples

Figure 3 further shows an example of how a media data stream prepared for one protection scheme can be efficiently adapted using sample variants to support another protection scheme. The figure shows samples in a series from left to right. The top row is made of samples from the original media data stream and the bottom row is made of sample variants from the variant media data. The sample variant may have the same KID or it may have a different KID from the samples of the original media data stream. The protection scheme can be different between samples from the original media data stream and samples from the variant media data.

The bottom two rows show the sample variant processing output samples of a single protection scheme. Access to the samples is under control of KIDs and protection scheme. It enables the application to carry media data stream with more than one protection scheme.

The control point for the use of the proposed framework is the content publisher:

— The content publisher encodes encrypted, compressed variant media data into the ISOBMFF file or MPEG-2 TS and ensures that each set of variant media data for a given sample time is encrypted with a key and signalled with a KID and protection scheme.

— The content publisher works with the DRM system to manage the release of KIDs/keys and protection scheme information such that the playback path (the actual sample data used during playback) is controlled and the player can only decrypt and render the data that it has been authorized to render.

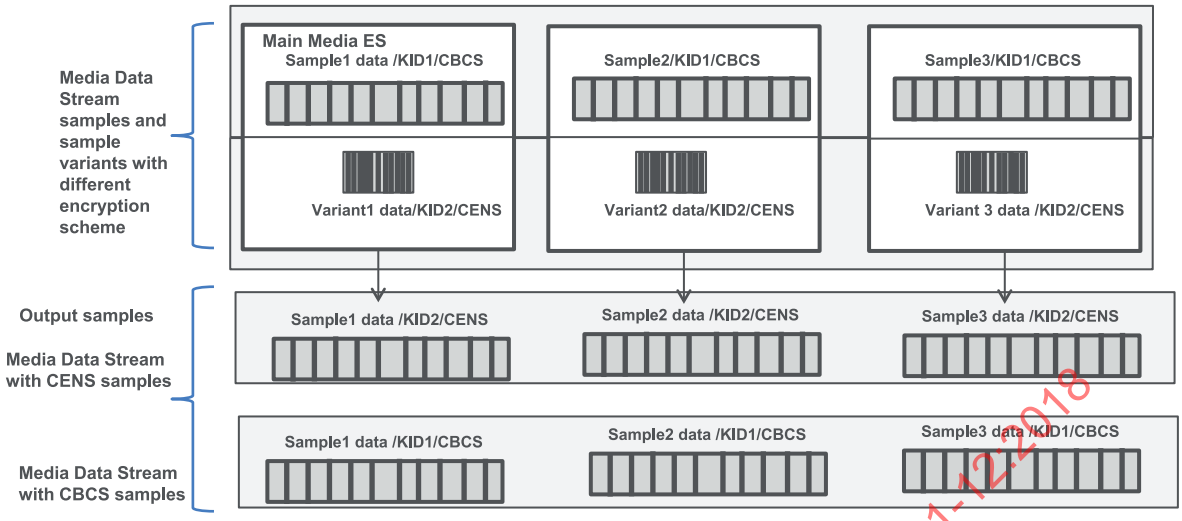

Figure 3 — Example sample variant structure for multiple protection schemes

Figure 4 shows the decoder model for the processing of files or transport streams that utilize sample variants. Critical to the sample variant decoding process is control over if and how the sample variants are processed.

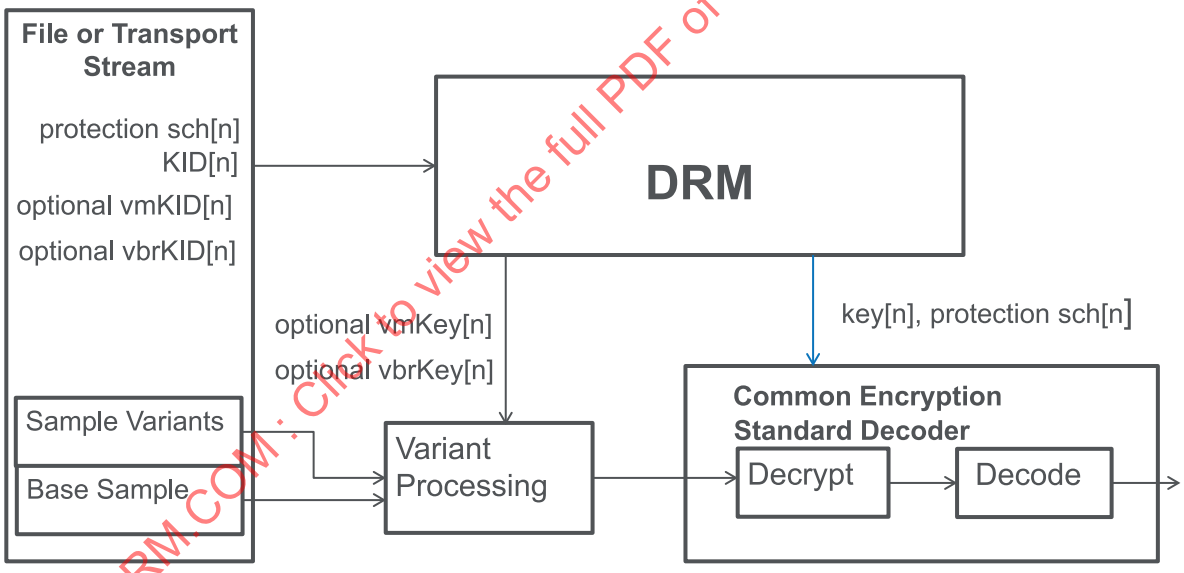

Figure 4 — Variant decoder model

NOTE 2 The decrypt and decode steps are standard operations as they would be for any common encryption enabled decoder.

By operating in the encrypted/compressed domain, secure baseband link operation (e.g. dedicated, secure video pathways) is preserved and is fully compatible with CENC or CETS.

As an alternative to the sample variants processing by the decoder, an extractor can also be used to pre-process the sample variants to generate a new media data stream and associated metadata for ISOBMFF or MPEG-2 TS. Clause 13 provides a high-level informative description of the extractor model.

6 Variant constructors

6.1 General

A variant constructor defines which bytes are used to assemble a sample variant. There may be one or more variant constructors defined for a given sample.

The variant processor may use a variant constructor if the variant processor has access to the variant constructor. In addition to the presence of the variant constructor, "access" includes cryptographic access. A variant constructor defines which data is used to assemble a sample variant and the associated media KID, protection scheme and initialization vector for decrypting the sample variant.

6.2 Access to variant constructors

If the decoder is given access to the media key, based on media KID and protection scheme, for the sample defined by the media data stream, sample variant processing does not occur for this sample. If the decoder does not have access to the original media key for the sample defined by the media data stream, e.g. due to mismatch of media KID or protection scheme, the variant processor shall be given access to one variant constructor associated with the sample. The variant constructor may be either unencrypted or encrypted.

NOTE Some application use cases, such as forensic watermarking, require that all variant constructors be encrypted to protect the integrity of the use case.

If a variant constructor is encrypted, a KID is associated with the variant constructor that identifies the key with which that variant constructor has been encrypted. The KID/key associated with the variant constructor controls access to a particular variant constructor and is therefore a function of the set of KID/key value pairs made available to the variant processor by the DRM system. Only one variant constructor per sample should be made available to the variant processor. If the variant processor is given access to a variant constructor, the decoder shall also be given access to the media key associated with the media KID and protection scheme defined in the variant constructor.

For a given variant data stream, all the variant constructors are either encrypted with the same protection scheme or all the variant constructors are unencrypted.

If the variant processor has access to more than one KID/key associated with encrypted variant constructors for a given sample, the variant processor utilizes the first variant constructor that it has access to in data encoding order.

If the variant processor has access to more than one unencrypted variant constructors for a given sample, the variant processor utilizes the first variant constructor that has matching media KID/key and protection scheme.

The variant processor uses exactly one variant constructor to assemble a sample variant.

6.3 Encryption of variant constructors

An encrypted variant constructor shall be encrypted with a "variant constructor key".

As a variant processor is provided only with the variant constructor keys for the encrypted variant constructor that is to be used by that particular variant processor, any variant constructors not used by that variant processor are not exposed by a security compromise of that variant processor.

7 Variant byte ranges

7.1 Overview

Each variant constructor defines a sequence of one or more variant byte ranges. Each variant byte range defines the location of a sequence of bytes that might constitute bytes in a sample variant. Variant byte ranges can contain unused data.

The sequences of variant byte ranges defined in a variant constructor are grouped into one or more variant byte range groups. Each variant byte range group shall define one or more variant byte ranges. An individual variant byte range within a variant byte range group:

— may reference bytes of data which constitute bytes in a sample variant that is made available to certain variant processors ("real variant byte range");

— may reference bytes of data that are not made available to any variant processor ("fake variant byte range").

A "fake variant byte range" can be used to hide the amount of actual "real variant byte ranges" defined within a variant constructor. The variant processor uses all variant byte ranges that it has access to. In addition to the presence of the variant byte range, "access" includes cryptographic access.

Data for different sample variants can be stored non-contiguously as referenced by different variant constructors. Data for a particular sample variant can also be stored non-contiguously using a sequence of two or more variant byte ranges.

7.2 Access to variant byte ranges

If a variant byte range within a variant byte range group signals that the data referenced by the variant byte range is unencrypted (and the variant processor has access to the variant constructor), then the variant processor has access to the variant byte range and the associated unencrypted bytes.

If the variant byte range defined within a variant byte range group signals that the data referenced by the variant byte range is encrypted, then access to the variant byte range and the associated bytes is controlled by the KID/key associated with each variant byte range — either the media key and protection scheme defined by the variant constructor if no variant byte range key and protection scheme are defined for the particular variant byte range group, or by the variant byte range key and protection scheme if one is defined. Access to the variant byte range and the associated data referenced by a variant byte range is therefore a function of the set of KID/key value pairs and protection scheme made available to the variant processor by the DRM system. Only one variant byte range within a variant byte range group should be made available to the variant processor.

If the variant processor has access to more than one KID/key associated with variant byte ranges within the same variant byte range group for a given sample, the variant processor uses the first variant byte range that it has access to in data encoding order. The variant processor uses at most one variant byte range within a variant byte range group to assemble a sample variant.

Variant byte ranges can be used to efficiently encode only the (typically small) differences in sample variants for a given presentation time without repeating non-difference data or exposing sample variant differences to the variant processor. This is achieved through double encryption, where the difference data is first encrypted by the media key and then encrypted by the variant byte range key. The variant processor requires access to the variant byte range key to decrypt such difference data and therefore access to the difference data can be controlled via the variant byte range key. This enables reuse of common data and preserves sample variant compatibility with CENC or CETS, which requires that only one media key be applied to a given sample. If variant byte ranges did not provide this capability then it would be necessary to repeat all data for each sample variant, including difference and non-difference data, so as to protect difference data with a different key, which is inefficient.

7.3 Encryption of variant byte range information

Variant byte range definitions are not individually encrypted (they are encrypted as part of the variant constructor). Variant byte range definitions are unencrypted if the variant constructor is unencrypted.

8 Sample variants

8.1 General

The data used for rendering a sample is defined by either a variant constructor (if the variant processor has access to the variant constructor for the sample) or the media data defined by the media data stream. When variant constructors are used, the actual data used for reconstructing the sample is obtained by assembling, in the order of appearance in the variant constructor, the byte data referenced by the variant byte ranges made available to the variant processor as per Clause 7. This construction shall result in a valid encrypted sample for the signalled underlying encryption system; this sample is a sample variant.

8.2 Access to sample variants

Once the sample variant is assembled from the variant byte ranges, access to the sample data is controlled by the media key and the protection scheme defined in the variant constructor and is therefore a function of the set of KID/key value pairs and the protection scheme made available to the variant processor by the DRM system.

8.3 Encryption of sample variants

Sample variants shall always be encrypted according to the scheme signalling of the corresponding variant media data. The protection scheme of the variant media data may be different from the associated media data stream. Variant byte ranges of a sample variant may be unencrypted or may be encrypted with a media key. The media key is associated with one or more samples.

When bytes in a sample variant are encrypted with a media key, one or more byte ranges of the encrypted variant media data may be further encrypted (double encrypted) according to the common encryption signalling with a “variant byte range key”.

As a variant processor is provided only with the variant byte range keys for double encrypted variant media data that are to be used by that particular variant processor, double encrypted variant media data not used by that variant processor are not exposed by a compromise of that variant processor.

9 Variant data stream

9.1 Variant data

9.1.1 General

This section defines common data structures of variant data for carriage in both ISOBMFF and MPEG-2 TS.

9.1.2 Definition

A sample in a variant data stream is either empty (zero size) or a `VariantData()` structure. Each `VariantData()` shall be aligned to the beginning of a byte.

9.1.3 Syntax

```
VariantData()
{
   VariantConstructorList() variant_list;
   VariantConstructor()[] variant_constructors;
   unsigned int(8)[] variant_pool;
}
```

9.1.4 Semantics

`variant_list` — the variant constructor list as defined in 9.2.

`variant_constructors` — the array of variant constructors referenced by the variant constructor list.

`variant_pool` — a pool of variant bytes that may be referenced by a variant constructor.

9.2 Variant constructor list

9.2.1 Definition

The `VariantConstructorList()` defines sample specific information on the location of potential variant constructors for sample variants.

Each sample definition in a variant data stream may have one or more variant constructor location entries in the `VariantConstructorList()`. As required in 6.2, exactly one individual variant constructor location entry is used during playback of a given sample and the variant processor uses the first variant constructor that it has access to in order of definition in the `VariantConstructorList()` structure.

Individual entries in the `VariantConstructorList()`:

— may reference a `VariantConstructor()` for the sample definition that is used during particular playback scenarios ("real variant constructors"); or

— may reference bytes that are not made available to any variant processor ("fake variant constructor").

NOTE Without access to the decryption key referenced by `vcKID`, fake variant constructors and real variant constructors are indistinguishable. Fake variant constructors can be used to hide the number of real variant constructors defined in the `variant_constructors` array.

9.2.2 Syntax

```
VariantConstructorList()
{
   unsigned int(32)      size;
   unsigned int(8)       variant_constructors_count;
   for(i=1 ; i<= variant_constructors_count; i++) {
   unsigned int(8)[16]   vcKID;
   unsigned int(8*IV_Size) vcIV;
   unsigned int(32)      variant_constructor_offset;
   unsigned int(32)      variant_constructor_size;
   }
   unsigned int(8)[]     padding;
}
```

9.2.3 Semantics

`size` — shall be set to the size, in bytes, of the `VariantConstructorList()`.

`variant_constructors_count` — shall be set to the number of variant constructor entries in the variant_constructors array in the `VariantData()`.

vcKID — the "variant constructor KID". This KID shall indicate the ID of the variant constructor metadata key used for decrypting the encrypted variant constructor. If the variant constructor is unencrypted, this field shall be set to 0.

vcIV — the "variant constructor Initialization Vector". This field shall contain the initialization vector used for decrypting the encrypted variant constructor. If the variant constructor is unencrypted, this field shall be set to 0.

variant_constructor_offset — the byte offset of the corresponding VariantConstructor(). This offset is relative to the start of the VariantData().

variant_constructor_size — the length, in bytes, of the VariantConstructor(). The combination of variant_constructor_offset and variant_constructor_size indicates the location and size of the VariantConstructor(). The byte range defined by variant_constructor_offset and variant_constructor_size shall only reference bytes within the variant_constructors array in the VariantData() and no other bytes.

padding — this byte array may contain any data and be used to increase the size of the VariantConstructorList() up to the size specified by the size field above.

NOTE 1 This padding can be used to obfuscate the actual size of the VariantConstructorList() if it is encrypted.

NOTE 2 IV_Size is defined in 10.2.3 and 11.2.4, respectively, for carriage in ISOBMFF and MPEG-2 TS.

9.3 Variant constructor

9.3.1 Syntax

```
VariantConstructor()
{
   unsigned int(8)[16]                KID;
   unsigned int(8*IV_Size)            IV;
   unsigned int(32)                   variant_byte_ranges_count;
   for( i=1; i<= variant_byte_ranges_count; i++ )
    {
      unsigned int(8)                 variant_byte_range_flags;
      if( variant_byte_range_flags & 0x02 )
      {
            unsigned int(8)[16]       vbrKID;
            unsigned int(8*IV_Size)   vbrIV;
      }
      if( variant_byte_range_flags & 0x08 ) {
         unsigned int(8)              variant_stream_reference_index;
      }
      signed int(8)                   relative_sample_number;
      unsigned int(32)                variant_byte_range_offset;
      if( variant_byte_range_flags & 0x06 != 0x02 ) {
         unsigned int(32)             variant_byte_range_size;
      }
   }
   unsigned int(8) []                 padding;
}
```

9.3.2 Semantics

KID — the media KID. This KID shall indicate the ID of the media key that is used for decrypting the encrypted variant media data after re-assembly of the applicable variant byte ranges. Decryption occurs in accordance with the protection scheme signalled in the corresponding variant data stream.

IV — the Initialization Vector that shall be used for decrypting the encrypted variant media data after re-assembly of the applicable variant byte ranges in accordance with the protection scheme signalled in the corresponding variant data stream.

`variant_byte_ranges_count` — shall be set to the number of variant byte ranges defined for this variant constructor. See Clause 7 for more information.

`variant_byte_range_flags` — shall be set as follows:

— 0x01 encrypted When set, the variant media data referenced by the variant byte range shall be encrypted with the media key.

— 0x02 double-enc When set, the variant media data referenced by the variant byte range shall be double encrypted with a variant byte range key. The meaning is undefined when `variant_byte_range_flags` signals that the variant media data referenced by the variant byte range is unencrypted.

— 0x04 group-start When set, the variant byte range shall be the start of a variant byte range group and thus provides a marker for variant byte range groups within the `VariantConstructor()`. As per Clause 7, the variant byte ranges defined in the `VariantConstructor()` are grouped into one or more variant byte range groups, and one variant byte range from each variant byte range group is used by the variant processor. This therefore requires that even if there is only one variant byte range defined in the `VariantConstructor()`, or there is only one variant byte range within a variant byte range group (i.e. there are no alternative variant byte ranges for a particular byte range of the variant media data), the start of variant byte range group be signalled with this singular variant byte range. As per 7.2, if more than one variant byte range appears in a single variant byte range group, each is double encrypted in order to limit the variant processor access to one byte range within the byte range group.

NOTE 1 This flag can be used by a variant processor to determine that a data error has occurred. If no variant byte range in a variant byte range group is recognized, an error has occurred.

— 0x08 data-source When set to 0, the data source for this range shall be the original media data stream. When set to 1, the variant_stream_reference_index indicates which variant data stream shall be the data source.

`vbrKID` — the "variant byte range KID". This KID shall indicate the ID of the variant byte range key used for decrypting the double encrypted variant media data.

`vbrIV` — the "variant byte range Initialization Vector". This field shall contain the initialization vector used for decrypting the double encrypted variant media data.

`variant_stream_reference_index` — If this value is 0, the variant byte range references data drawn from this variant data stream. Otherwise, the variant byte range references data drawn from a separate variant data stream indicated by reference. In the case of non-zero value in ISOBMFF, this field shall be set to the 1-based index, in order of reference definition, of the track reference from this variant data stream to another variant data stream containing the variant media data to be used, as defined in 10.2.2. In the case of non-zero value in MPEG-2 TS, this field shall be set to the 1-based index of `variant_es_PID`, in order of reference definition, of the packetized elementary stream references from this variant data stream to another variant data stream containing the variant media data to be used, as defined in 11.2.2.

`relative_sample_number` — having found the data source (see the data-source flag and `variant_stream_reference_index` field above), this field defines which data source sample shall be used for the variant byte range as follows: when set to 0, the data source sample is the time-parallel associated sample per 10.3.2 or 11.3; when set to a negative value, the Nth prior sample is used; when set to a positive value, the Nth succeeding sample is used.

`variant_byte_range_offset` — is the byte offset from the start of the referenced data source sample to the beginning of the data for this variant byte range.

`variant_byte_range_size` — the size of the variant byte range in bytes. The combination of `variant_byte_range_offset` and `variant_byte_range_size` indicates a byte range for the variant byte range in the referenced data source sample. The variant byte range defined by `variant_byte_range_offset` and `variant_byte_range_size` shall only reference bytes within the

referenced data source sample and no other bytes. If there is more than one variant byte range in a variant byte range group, this field only exists for the first variant byte range, as the size of each of the variant byte ranges in a variant byte range group is the same.

`padding` — this byte array may contain any data and be used to increase the size of the variant constructor up to the size specified for this variant constructor in the variant constructor list.

NOTE 2 This padding can be used to obfuscate the actual size of the variant constructor as it is encrypted.

10 Carriage of variant data stream in ISOBMFF

10.1 General

Variant data is stored in an ISOBMFF metadata track (variant track). An ISOBMFF media track (media track) or variant track may be associated with one or more variant tracks as defined in 10.2.2.

— When an association is established between a media track and a variant track, sample variant processing occurs whenever a decoder (or sample variants media data stream Extractor) does not have access to the KID/key and the protection scheme defined for a sample in the media track as defined in 6.2.

— When an association is established from a variant track (original variant track) to another variant track (other variant track), variant data contained in the other variant track can be utilized by the original variant track.

— Samples within associated tracks are associated if they are time-parallel as defined in 10.3.2.

10.2 Variant tracks

10.2.1 Definition

Variant data shall be stored in an ISOBMFF metadata track that complies with the following constraints:

1. The track shall use the `'meta'` `handler_type` in the Handler Reference Box (`'hdlr'`) as per ISO/IEC 14496-12:2015, Clause 12.
2. The track shall use the `VariantMetaDataSampleEntry()` sample entry as defined in 10.2.3.
3. Variant data is stored in the track as samples in accordance with 10.3.
4. The track shall use the same timebase as the corresponding video, audio or other variant tracks.
5. If the variant track protection scheme is different from the protection scheme used with the associated media track, the variant track shall be fully compatible with CENC.

10.2.2 Association

There are two types of track associations involving variant tracks:

— A media data stream containing audio-visual content may be associated with one or more variant data streams that define possible alternate samples.

— A variant data stream may be associated with zero or more additional variant data streams (in addition to the original media data stream) from which it may reference data bytes to use as variant media data.

NOTE In the context of the latter case, when a variant data stream is associated with another variant data stream as a reference for data bytes to use as variant media data, the variant constructors in the referenced stream are not being processed. Instead, the referenced variant data is just being used as a pool of available data bytes, and those bytes can come equally from any part of the variant data (e.g. `variant_list`, `variant_constructors` or `variant_pool`).

For both types of track associations, tracks may be associated with variant data streams via one of the following means:

— An externally defined context providing an ordered list of variant data stream references.

— In the source track (e.g. in the original media data stream or source variant data stream), using a Track Reference Type Box in the Track Reference Box (`'tref'`) of the Track Box (`'trak'`) that has a `reference_type` of either `'cvar'` or `'cva2'` and one or more `track_IDs` that each correspond to a `track_ID` of a variant data stream in the same file.

The following additional requirements apply to `track_IDs` in a Track Reference Type Box of `reference_type 'cvar'` or the variant track(s) of `reference_type 'cva2'`:

— `track_ID` may have a value that does not correspond to a `track_ID` of a track in the same file. This document does not define how the referenced file containing such a track is located.

— If the `track_ID` does correspond to a `track_ID` of a track in the same file, the corresponding track shall be a variant data stream that complies with 10.2.1.

Whichever method is used to define an association (i.e. whether by externally defined context or Track Reference Box), there shall be no more than 255 tracks referenced by a single source track.

Variant track references defined for a media data stream shall be defined in variant constructor search order. The variant processor processes variant tracks according to this order when searching for an accessible variant constructor.

Variant track references defined for a variant data stream shall be defined in variant data stream reference order such that values of `variant_stream_reference_index` in a variant constructor represent a 1-based offset into the list of available variant data stream references to identify the stream from which to draw bytes of variant media data.

10.2.3 Variant metadata sample entry

10.2.3.1 Syntax

```
class VariantMetaDataSampleEntry() extends MetaDataSampleEntry
('reference_type') {
   unsigned int(32)   variant_constructor_scheme_type;
   unsigned int(32)   variant_constructor_scheme_version;
   unsigned int(32)   media_track_scheme_type;
   unsigned int(32)   media_track_scheme_version;
   unsigned int(32)   IV_Size;
   unsigned int(32)   variant_byte_range_scheme_type;
   unsigned int(32)   variant_byte_range_scheme_version;
}
```

10.2.3.2 Semantics

`reference_type` — shall be set to the four-character code as follows:

`'cvar'`: when variant constructors are encrypted and the protection scheme applied to the variant track is the same as the associated media track.

'cva2': when variant constructors can be encrypted or unencrypted and the protection scheme applied to the variant track can be the same as or different than the protection scheme applied to associated media track.

NOTE 1 The four-character used for the previous edition was 'cvar'.

NOTE 2 The recommended four-character to use with this document is 'cva2'.

variant_constructor_scheme_type — shall be set to the four-character code defining the protection scheme applied to variant constructors in the track, as per 10.3.1.

variant_constructor_scheme_version — shall be set to the version of the protection scheme applied to the variant constructors in the track, as per 10.3.1.

media_track_scheme_type — shall be set to the four-character code defining the protection scheme applied to the variant track, as defined for the scheme_type field in the Protection Scheme Information Box ('sinf') of the variant track by ISO/IEC 14496-12:2015, subclause 8.12.5.3.

NOTE 3 When VariantMetaDataSampleEntry() reference_type is set to 'cvar', it is set to same protection schema_type applied to the associated media track.

media_track_scheme_version — shall be set to the version of the protection scheme applied to the variant track, as defined for the scheme_version field in the Protection Scheme Information Box ('sinf') of the variant track by ISO/IEC 14496-12:2015, subclause 8.12.5.3.

NOTE 4 When VariantMetaDataSampleEntry() reference_type is set to 'cvar', it is set to same protection scheme_version applied to the associated media track.

IV_Size — shall signal the size of the IV in bytes that is applied to the variant track (as used in the VariantConstructorList and VariantConstructor structures). If the associated variant track protection scheme is the same as the associated media track protection scheme, the IV_Size shall match the IV_Size of the associated media track. Otherwise, the IV_Size shall match the IV_Size defined in the associated sample variant track 'tenc'.

variant_byte_range_scheme_type — shall be set to the four-character code defining the protection scheme applied to the double encryption of bytes referenced by variant byte ranges, as per 10.3.1.

variant_byte_range_scheme_version — shall be set to the version of the protection scheme applied to the double encryption of bytes referenced by variant byte ranges, as per 10.3.1.

10.3 Variant data

10.3.1 Encryption

As defined in 6.3, variant constructors may be encrypted, and as per 10.2.3, the protection scheme is signalled in the VariantMetaDataSampleEntry(). One of the following CENC encryption modes shall be used to encrypt variant constructors:

1. AES-CTR Full Sample Encryption: signalled with a four-character code value of variant_constructor_scheme_type field value of 'cvar' and a variant_constructor_scheme_version value of 0x00010000 (Major version 1, Minor version 0) in the VariantMetaDataSampleEntry() as per 10.2.3.

2. AES-CBC-128 Full Sample Encryption: signalled using a four-character code value of variant_constructor_scheme_type field value of 'cva1' and a variant_constructor_scheme_version field value of 0x00010000 (Major version 1, Minor version 0) in the VariantMetaDataSampleEntry() as per 10.2.3.

In case the variant constructors are not encrypted, it shall be signalled with a four-character code value of variant_constructor_scheme_type field value of 'cva2' and variant_

`constructor_scheme_version` value of 0x00010000 (Major version 1 and minor version 0) in the `VariantMetadataSampleEntry()`.

NOTE When `VariantMetaDataSampleEntry()` `reference_type` is set to `'cvar'`, the variant constructors are encrypted.

As defined in 8.3, sample variants assembled from variant byte ranges defined in a variant constructor are encrypted according to the scheme signalling of the associated media track and as per 10.2.3 this scheme is also signalled in the `VariantMetaDataSampleEntry()`.

As defined in 8.3, bytes referenced by a variant byte range may be double encrypted with a "variant byte range key" and as per 10.2.3 the double encryption scheme is signalled in the `VariantMetaDataSampleEntry()`.

One of the following CENC encryption modes shall be used for double encryption of byte data in a variant track:

1. AES-CTR Full Sample Encryption: signalled with a four-character code value of `variant_byte_range_scheme_type` field value of `'cvar'` and a `variant_byte_range_scheme_version` value of 0x00010000 (Major version 1, Minor version 0) in the VariantMetaDataSampleEntry() as per 9.2.3.

2. AES-CBC-128 Full Sample Encryption: signalled using a four-character code value of `variant_byte_range_scheme_type` field value of `'cvar'` and a `variant_byte_range_scheme_version` field value of 0x00010000 (Major version 1, Minor version 0) in the `VariantMetaDataSampleEntry()` as per 10.2.3.

The bytes referenced by a variant byte range shall be treated as a single sample for the purposes of applying one of these CENC encryption modes.

10.3.2 Association

Samples are associated as follows:

1. A sample in a media track shall be associated with a sample in a variant track referenced by the media track if the samples are time-parallel.

2. A sample in a variant track shall be associated with a sample in another variant track referenced by the variant track if the samples are time-parallel.

3. Samples are considered to be time-parallel as follows: if t_0 is the decode time of the sample in the original track, then the time-parallel sample in a referenced track is the sample in that referenced track that has a decode time t_v and a duration D such that $t_v \leq t_0 < t_v + D$.

NOTE 1 Sample association occurs at media decode time before any consideration of edit lists or composition offset.

NOTE 2 A sample in a variant track can have zero data size if no variant data is to be provided at that particular sample time.

An example of media track and variant track referencing is shown in Figure 5.

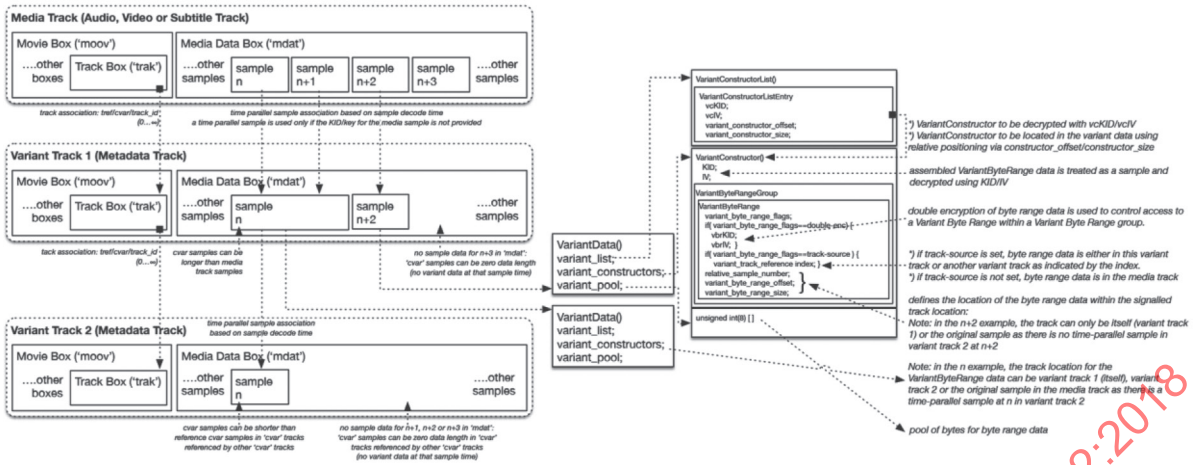

Figure 5 — Media track and variant track referencing

11 Carriage of variant data stream in MPEG-2 TS

11.1 General

Variant data is carried as MPEG-2 TS metadata stream (sample variants metadata stream) as defined in ISO/IEC 13818-1:2018, 2.12. The sample variant metadata stream shall be carried in a packetized elementary stream (PES) as defined in 11.2. A sample variant metadata stream is associated with an audio-visual content in an MPEG-2 TS. An audio-visual content stream or sample variant metadata stream may be associated with one or more sample variant metadata streams as defined in 11.2.2.

— When an association is established between an audio-visual content stream and a sample variant metadata stream, sample variant processing occurs whenever a decoder (or sample variants media data stream Extractor) does not have access to the KID/key and protection scheme defined for a sample in the audio-visual content stream as defined in 6.2.

— When an association is established from a sample variant metadata stream (original sample variant metadata stream) to another sample variant metadata stream, variant data contained in the other sample variant metadata stream can be utilized by the original sample variant metadata stream.

— Access units within associated packetized elementary streams are associated if they are time-parallel as defined in 11.3.

11.2 Sample variant metadata streams

11.2.1 Definition

Variant data shall be carried in MPEG-2 TS metadata stream that complies with the following constraints:

NOTE 1 The sample variant metadata stream shall be carried in a private PES stream with `stream_id` set to 0xBD (`private_stream_id_1`).

NOTE 2 The sample variant metadata stream shall be carried in a PES stream with `stream_type` set to 0x15.

1. Variant data shall be stored in the packetized elementary stream as access units in accordance with 11.2.2.

2. The sample variant metadata stream shall use `metadata_descriptor()` as defined in 11.2.3 to signal the sample variant metadata stream in the Program Map Table (PMT).

3. The sample variant metadata stream shall use the same timebase as the corresponding audio-visual content stream.

4. If the sample variant metadata stream protection scheme is different from the protection scheme used with the associated audio-visual content stream, the sample variant metadata stream shall be fully compatible with CETS.

11.2.2 Association

There are two types of stream associations involving variant streams:

— A media data stream containing audio-visual content may be associated with one or more variant data streams that define possible alternate samples.

— A variant data stream may be associated with zero or more additional variant data streams (in addition to the original media data stream) from which it may reference data bytes to use as variant media data.

NOTE In the context of the latter case, when a variant data stream is associated with another variant data stream as a reference for data bytes to use as variant media data, the variant constructors in the referenced stream are not being processed. Instead, the referenced variant data is just being used as a pool of available data bytes, and those bytes can come equally from any part of the variant data (e.g. `variant_list`, `variant_constructors` or `variant_pool`).

For both types of packetized elementary stream associations, streams may be associated with variant data streams via one of the following means:

— An externally defined context providing an ordered list of variant data stream references.

— In the variant data stream, using `metadata_descriptor()` in PMT that has `metadata_application_format_identifier` and `metadata_format_identifier` of either `'cvar'` or `'cva2'`, and using the decoder configuration data, `Sample_Variant_Metadata_Config()`, in the `metadata_descriptor()` that has media_es_PID value set to the PID value of corresponding media data stream (e.g. original audio-visual content stream) and zero or more `variant_es_PID` values set to additional variant data streams from which it may reference data bytes to use as variant media data.

There can be only one media data stream associated with a variant data stream. There can be more than one variant data stream in a program that are associated with the same media data stream.

Whichever method is used to define an association (i.e. whether by externally defined context or list of `variant_es_PID`), there shall be no more than 255 additional variant data streams referenced by a single variant data stream.

If there is more than one variant data stream specified in the PMT, then the variant processor processes variant data streams associated with a given media data stream according to the order specified in the PMT when searching for an accessible variant constructor.

Variant Data Streams referenced via `variant_es_PID` values shall be defined in order such that values of `variant_stream_reference_index` in a variant constructor represent a 1-based offset into the list of available variant data stream references to identify the stream from which to draw bytes of variant media data (e.g. a `variant_stream_reference_index` value of 1 would correspond to the initial entry in the list of `variant_es_PID` values).

11.2.3 Metadata descriptor for sample variant metadata stream

The presence of sample variant metadata stream is signalled with `metadata_descriptor()` in the PMT, as defined in ISO/IEC 13818-1:2018, 2.6.60, with additional requirements as defined in this subclause.

— The `metadata_application_format` shall be set to 0xFFFF with `metadata_application_format_identifier` set to either `'cvar'` or `'cva2'`.

— The metadata _format and metadata shall be set to 0xFF with `metadata_format_identifier` set to either `'cvar'` or `'cva2'`.

— The `metadata_application_format_identifier` and the `metadata_format_identifier` shall be set to the same value.

— The `metadata_service_id` shall be set to '1'.

— The `decoder_config_flags` shall be set to '001' providing a decoder configuration data of sample variant metadata configuration, `Sample_Variant_Metadata_Config()`, as defined in 11.2.4.

11.2.4 Sample variant metadata configuration

11.2.4.1 General

Decoding of sample variants requires the availability of decoder configuration data. When sample variants are carried in a MPEG-2 TS, then the metadata descriptor shall provide associated decoder configuration data in the same MPEG-2 TS.

11.2.4.2 Syntax

The syntax is as shown in Table 1.

Table 1 — Sample variant metadata configuration

Syntax	No. of bits	Mnemonic
Sample_Variant_Metadata_Config() {		
variant_constructor_scheme_type	**32**	**uimsbf**
variant_constructor_scheme_version	**32**	**uimsbf**
variant_es_scheme_type	**32**	**uimsbf**
variant_es_scheme_version	**32**	**uimsbf**
IV_size	**32**	**uimsbf**
variant_byte_range_scheme_type	**32**	**uimsbf**
variant_byte_range_scheme_version	**32**	**uimsbf**
IV_size	**32**	**uimsbf**
reserved	**3**	**uimsbf**
media_es_PID	**13**	**uimsbf**
num_of_variant_es	**8**	**uimsbf**
for (i=0; i< num_of_variant_es; i++) {		
reserved	**3**	**uimsbf**
variant_es_PID[num_of_variant_es**]**	**13**	**uimsbf**
}		
}		

11.2.4.3 Semantics

variant_constructor_scheme_type: The four-character code defining the protection scheme applied to the constructors in the sample variant metadata stream, as defined in 11.2.6.

variant_constructor_scheme_version: The version of the protection scheme applied to the constructors in the sample variant metadata stream, as defined in 11.2.6.

variant_es_scheme_type: A four-character code defining the protection scheme applied to the sample variant metadata stream, as defined for the `scheme_type` field in the Protection Scheme Information Box (`'sinf'`) of the variant track by ISO/IEC 14496-12:2015, subclause 8.12.5.3.

variant_es_scheme_version: The version of the protection scheme applied to the sample variant metadata stream, as defined for the `scheme_version` field in the Protection Scheme Information Box (`'sinf'`) of the variant track by ISO/IEC 14496-12:2015, subclause 8.12.5.3.

IV_size: The size of the IV in bytes that is applied to the sample variant metadata stream (as used in the variant constructor list and variant constructor). If the associated sample variant metadata stream protection scheme is the same as the corresponding audio-visual content stream protection scheme, the IV_Size shall match the IV_Size of the corresponding audio-visual content stream.

variant_byte_range_scheme_type: The four-character code defining the protection scheme applied to the double encryption of bytes referenced by variant byte ranges, as per 11.2.6. If double encryption is not used, it shall be set to 0.

variant_byte_range_scheme_version: The version of the protection scheme applied to the double encryption of bytes referenced by variant byte ranges, as per 11.2.6. If double encryption is not used, it shall be set to 0.

media_es_PID: The PID for corresponding audio-visual content stream.

num_of_variant_es: The number of variant metadata streams.

variant_es_PID: The PID for corresponding variant metadata stream.

11.2.5 PES Packetization

For PES packetization, no specific data alignment constraints apply. For synchronization and system target decoder (STD) management, presentation time-stamps (PTSs) and, when appropriate, decoding time-stamps (DTSs) are encoded in the header of the PES packet that carries the sample variants metadata stream data. For PTS and DTS encoding, the constraints and semantics apply as defined in ISO/IEC 13818-1:2018, 2.4.3.7 and 2.7.

The PES for sample variants metadata stream carries VariantData() as defined in 9.1.

In order to assist variant processing, the PES packets of the variant data should be packed into the MPEG-2 TS before packing PES packets for corresponding audio-visual content stream access unit. This allows variant processor to have access to all the variant data at the time of processing the corresponding audio-visual content access unit.

11.2.6 Encryption

As defined in 6.3, variant constructors may be encrypted or unencrypted, and as per 11.2.3, the protection scheme is signalled in the `Sample_Variant_Metadata_Config()`.

One of the following encryption modes shall be with encrypted variant constructors:

1. CENC AES-CTR Full Sample Encryption: signalled with a four-character code value of `variant_constructor_scheme_type` field value of `'cvar'` and a variant_constructor_scheme_version value of 0x00010000 (Major version 1, Minor version 0) in the `Sample_Variant_Metadata_Config()` as per 11.2.4.

2. CENC AES-CBC-128 Full Sample Encryption: signalled using a four-character code value of `variant_constructor_scheme_type` field value of `'cva1'` and a `variant_constructor_scheme_version` field value of 0x00010000 (Major version 1, Minor version 0) in the `Sample_Variant_Metadata_Config()` as per 11.2.4.

In case the variant constructors are not encrypted, it shall be signalled with a four-character code of `variant_constructor_scheme_type` field value of `'cva2'` and `variant_constructor_scheme_version` value of 0x00010000 (Major version 1 and minor version 0) in the `Sample_Variant_Metadata_Config()`.

As defined in 8.3, sample variants assembled from variant byte ranges defined in a variant constructor are encrypted according to the scheme signalling of the associated sample variants metadata stream and, as per 11.2.4, this scheme is also signalled in the `Sample_Variant_Metadata_Config()`.

As defined in 8.3, bytes referenced by a variant byte range may be double encrypted with a "variant byte range key" and, as per 11.2.4, the double protection scheme is signalled in the `Sample_Variant_Metadata_Config()`.

One of the following CETS encryption modes shall be used for double encryption of byte data in a variant packetized elementary stream:

1. CENC AES-CTR Full Sample Encryption: signalled with a four-character code value of `variant_byte_range_scheme_type` field value of `'cvar'` and a `variant_byte_range_scheme_version` value of 0x00010000 (Major version 1, Minor version 0) in the `Sample_Variant_Metadata_Config()` as per 11.2.4.

2. CENC AES-CBC-128 Full Sample Encryption: signalled using a four-character code value of `variant_byte_range_scheme_type` field value of `'cva1'` and a `variant_byte_range_scheme_version` field value of 0x00010000 (Major version 1, Minor version 0) in the `Sample_Variant_Metadata_Config()` per 11.2.4.

The bytes referenced by a variant byte range shall be treated as a single sample for the purposes of applying one of these CETS encryption modes.

11.3 Association

Samples are associated as follows:

1. An access unit in sample variants metadata stream shall always be associated with an access unit in an audio-visual content stream referenced by the sample variants metadata stream if the access units are time-parallel.

2. Access units are considered to be time-parallel as follows: if t_o is the decode time of the access unit in the audio-visual content stream, then the time-parallel sample in a referenced sample variants metadata stream is the access unit in that referenced packetized elementary stream that has a decode time t_v and a duration D such that $t_v \leq t_o < t_v + D$.

12 Variant processor models & examples

12.1 Variant processor model for ISOBMFF

This subclause describes variant processor model for the framework based on ISOBMFF.

The variant processing may be used by the decoder model or the extractor model (see Clause 13 for the details of the extractor model).

The rendering of a sample is expected to satisfy the observable behaviour defined by the following model:

1. The data source for each sample is evaluated as follows:

    a. If the decoder or extractor has access to the sample in the media track (based on the access to media KID and protection scheme), the decoder or extractor proceeds to render the sample as per 6.2.

    b. If the decoder or extractor does not have access to the sample in the media track, the variant processor determines which variant constructor is the data source for the sample as defined in 6.2. The variant processor searches for an accessible variant constructor as follows:

        i. The variant processor searches each variant track referenced by the media track in order of reference definition e.g. the order of track references in the Track Reference Box (`'tref'`). See 10.2.2 for more information.

        ii. In each variant track searched, the variant processor determines if variant data exists for the time-parallel sample in the variant track. If variant data exists, the variant processor searches the `VariantConstructorList()` in the time-parallel sample in the variant track.

        iii. The variant processor continues to search until it finds a variant constructor KID ('vcKID') in a `VariantConstructorList()` that matches a KID/key the variant processor has access to or it finds a variant constructor KID ('vcKID') in a `VariantConstructorList()` set to 0.

2. In case the variant constructor is encrypted, using the variant constructor key and initialization vector defined in the `VariantConstructorList()` for the variant constructor selected by the variant processor, the variant processor decrypts the `VariantConstructor()` structure defined in 9.3.

3. The variant processor sequentially processes each variant byte range in the sequence of variant byte ranges defined in the decrypted or unencrypted variant constructor and assembles the variant media data for the sample as follows:

    a. If the variant byte range is signalled to be unencrypted per the definition of `variant_byte_range_flags` in 9.3, the byte range is put directly in the sample assembly and identified as unencrypted.

    b. If the variant byte range is signalled to be encrypted per the definition of `variant_byte_range_flags` in 9.3:

        i. If the variant byte range data is signalled as single encrypted with the media key per the definition of `variant_byte_range_flags` in 9.3, it is put directly in the sample assembly and identified as encrypted.

        ii. If the variant byte range media is signalled as double encrypted per the definition of `variant_byte_range_flags` in 9.3:

            a. If the variant byte range KID (`'vbrKID'`) defined by the variant byte range is available to the variant processor, the variant byte range data referenced by the variant byte range is decrypted using the variant byte range key referenced by the variant byte range KID (`'vbrKID'`) and the variant byte range initialization vector referenced by the variant byte range IV (`'vbrIV'`). This operation results in single encrypted data which is put in the sample assembly and identified as encrypted.

            b. If the variant byte range KID (`'vbrKID'`) defined by the variant byte range is not available to the variant processor, the variant byte range is skipped.

4. The assembled variant media data is decrypted using the media key defined by the variant metadata (as referenced by the KID field in the variant metadata defined in 9.3), in accordance with CENC.