INTERNATIONAL STANDARD

ISO/IEC 23001-12

First edition
2015-12-15

Information technology — MPEG systems technologies —

Part 12: Sample Variants in the ISO base media file format

Technologies de l'information — Technologies des systèmes MPEG —

Partie 12: Variantes d'échantillon dans le format ISO de base pour les fichiers médias

Reference number
ISO/IEC 23001-12:2015(E)

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

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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).

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

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

ISO/IEC 23001 consists of the following parts, under the general title *Information technology — MPEG systems technologies*:

— *Part 1: Binary MPEG format for XML*

— *Part 2: Fragment request units*

— *Part 3: XML IPMP messages*

— *Part 4: Codec configuration representation*

— *Part 5: Bitstream Syntax Description Language (BSDL)*

— *Part 7: Common encryption in ISO base media file format files*

— *Part 8: Coding-independent code points*

— *Part 9: Common encryption of MPEG-2 transport streams*

— *Part 10: Carriage of timed metadata metrics of media in ISO base media file format*

— *Part 11: Energy-efficient media consumption (green metadata)*

— *Part 12: Sample Variants in the ISO base media file format*

Information technology — MPEG systems technologies —

Part 12:
Sample Variants in the ISO base media file format

1 Scope

This part of ISO/IEC 23001 defines the carriage of Sample Variants in the ISO base media file format (ISO/IEC 14496-12).

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14496-12:2015[1)], *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 23001-7:2015, *Information technology — MPEG systems technologies — Part 7: Common encryption in ISO base media file format files*

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

3.1.1
Double Encrypted
Sample Variant byte range data encrypted by first a Media Key (as part of the encryption of the complete Sample Variant) and then second a Variant Byte Range key

Note 1 to entry: See 6.1.

3.1.2
Media Key
encryption key associated with one or more media samples

3.1.3
Media KID
encryption KID associated with one or more media samples

3.1.4
Sample Variant
assembled media sample replacing an original sample

3.1.5
Variant Byte Range
location of a sequence of bytes that might constitute a portion of a Sample Variant

1) ISO/IEC 14496-12 is technically identical to ISO/IEC 15444-12.

3.1.6
Variant Constructor
Sample Variant metadata that defines how to assemble an individual Sample Variant

3.1.7
Variant Media Data
media data used to construct a Sample Variant, some of which may come from the original sample media data

3.1.8
Variant Processor
logical module that performs the processing steps that implement the process of assembling Sample Variants

3.2 Abbreviated terms

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

CENC	Common ENCryption (as specified by ISO/IEC 23001-7:2015)
DRM	Digital Rights Management
ISOBMFF	ISO Base Media File Format (as specified by ISO/IEC 14496-12:2015)
IV	Initialization Vector
KID	Key Identifier

4 Overview (informative)

This part of ISO/IEC 23001 defines a framework for the carriage of Sample Variants in the ISOBMFF. Sample Variants are typically used to provide forensic information in the rendered sample data that can, e.g. identify the DRM client. This variant framework is intended to be fully compatible with ISOBMFF and CENC, and agnostic to the particular forensic marking system used.

The Sample Variant framework uses three core constructs to define and carry Sample Variant data in ISOBMFF: Variant Constructors, Variant Byte Ranges and Variant Samples.

NOTE The Variant Process Model described in Clause 9 can also assist in introducing the concepts.

Figure 1 shows a scenario where a sample (Sample 2) has a number of Sample Variants. Figure 1 shows 3 samples in a series left to right, the middle of which has variants. The top row is a conceptual depiction of what is encoded using ISOBMFF 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 in 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.

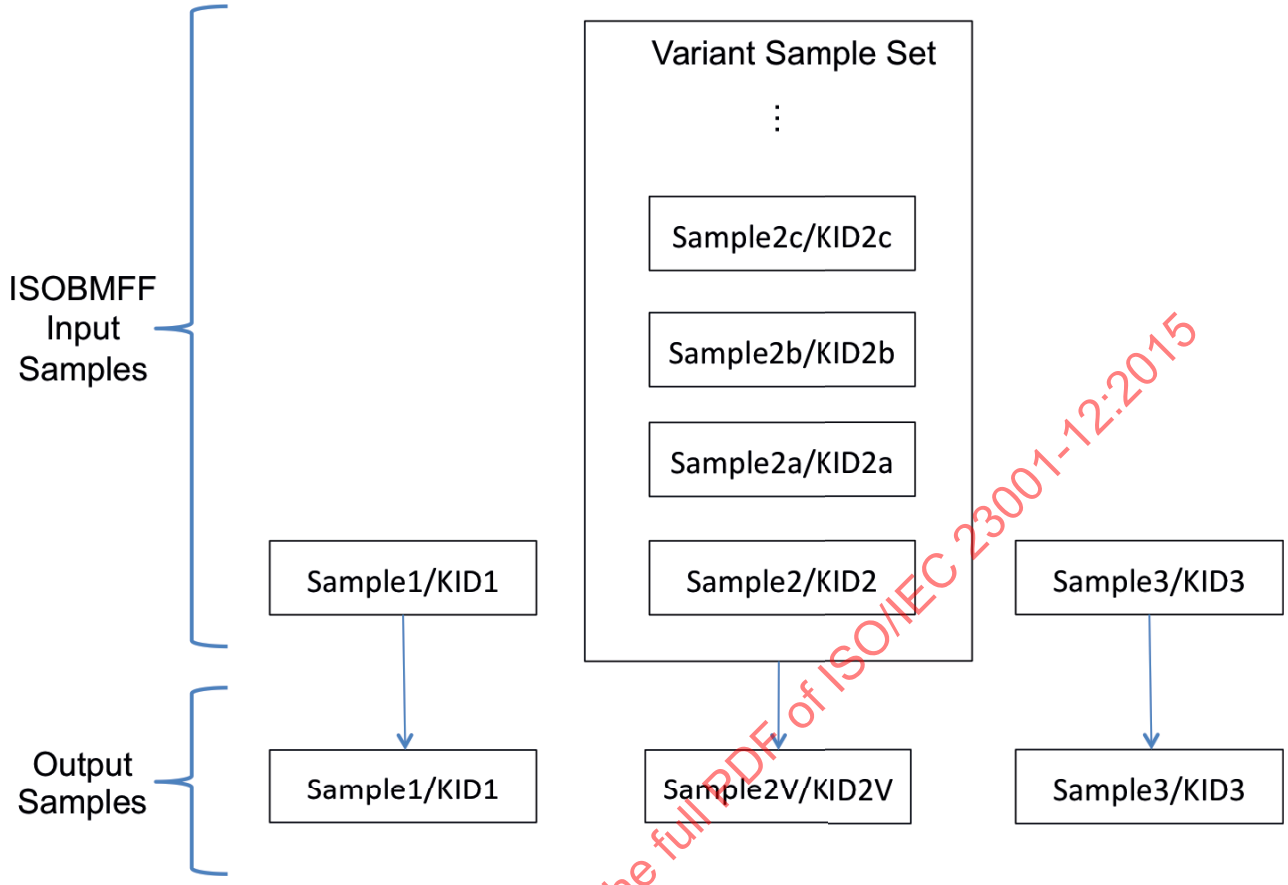

Figure 1 — Sample Variant structure

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

— the content publisher will encode encrypted, compressed Sample Variant data into the ISOBMFF file and ensure that each set of Sample Variant data for a given sample time is encrypted with a different key and signalled with a different KID.

— the content publisher will work with the DRM to manage the release of KIDs/keys 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.

The decoder model for the processing of the file is shown in Figure 2. Critical to the Sample Variant decoding process is control over if and how the Sample Variants are processed.

NOTE The decrypt and decode steps are standard operations as they would be for any CENC-enabled decoder.

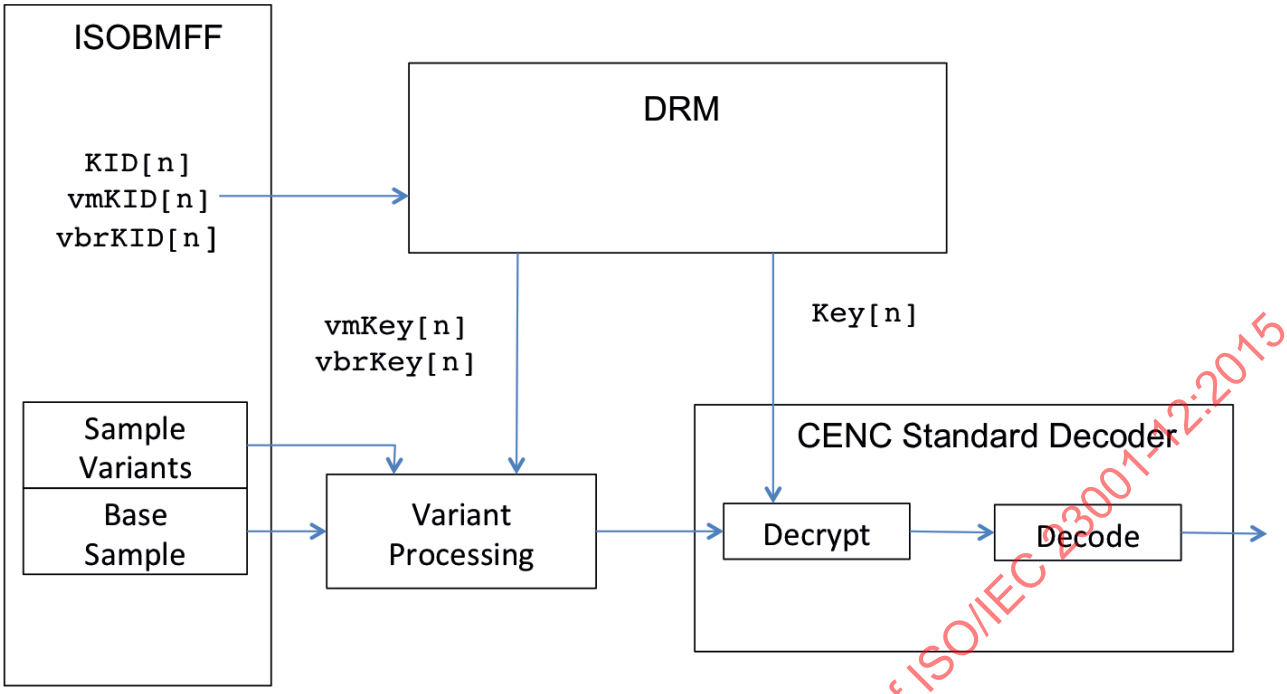

Figure 2 — Variant Decoder Model

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

5 Variant Constructors

5.1 Overview

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 ISOBMFF 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 and initialization vector for decrypting the Sample Variant.

5.2 Access to Variant Constructors

If the decoder is given access to the Media Key for the sample defined by the ISOBMFF media track, Sample Variant processing will not occur for this sample. If the decoder does not have access to the original Media Key for the sample defined by the ISOBMFF media track, the Variant Processor shall be given access to one Variant Constructor associated with the sample.

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. 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 defined in the Variant Constructor.

If the Variant Processor has access to more than one KID/Key associated with Variant Constructor for a given sample, the Variant Processor utilizes the first Variant Constructor that it has access to in data encoding order. The Variant Processor uses exactly one Variant Constructor to assemble a Sample Variant.

5.3 Encryption of Variant Constructors

Each Variant Constructor shall be encrypted with a “Variant Constructor key”.

As a Variant Processor is provided only with the Variant Constructor keys for the 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.

6 Variant Byte Ranges

6.1 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 sequence 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 that 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.

6.2 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 defined by the Variant Constructor if no Variant Byte Range key is defined for the particular Variant Byte Range group or by the Variant Byte Range key 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 made available to the Variant Processor by the DRM. 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, 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; this is inefficient.

6.3 Encryption of Variant Byte range information

Variant Byte Range definitions are not individually encrypted (they are encrypted as part of the Variant Constructor).

7 Sample Variants

7.1 Overview

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 per 8.3.3), or by the media data defined by ISOBMFF. 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 per Clause 6 and this construction shall result in a valid encrypted sample for the signalled underlying encryption system; this sample is a Sample Variant.

7.2 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 defined in the Variant Constructor and is therefore a function of the set of KID/Key value pairs made available to the Variant Processor by the DRM.

7.3 Encryption of Sample Variants

Sample Variants shall always be encrypted according to the scheme signalling of the associated media track. 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" per 8.3.4.

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.

8 ISO storage

8.1 Overview

Variant data is stored in an ISOBMFF metadata tracks (variant track). An ISOBMFF media track (media track) or variant track may be associated with one or more variant tracks as defined in 8.2.2.

— When an association is established between a media track and a variant track, Sample Variant processing will occur whenever a decoder does not have access to the KID/key defined for a sample in the media track as defined in 5.2.

— 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 8.3.5.

8.2 Variant tracks

8.2.1 Definition

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

a) The track shall use the `'meta'` `handler_type` in the Handler Reference Box (`'hdlr'`) per ISOBMFF Clause 12.

b) The track shall use the `VariantMetaDataSampleEntry()` sample entry as defined in 8.2.3.

c) Variant data is stored in the track as samples in accordance with 8.3.

d) The track shall use the same timebase as the corresponding video, audio or other variant tracks.

8.2.2 Association

ISOBMFF tracks may be associated with variant tracks via one of the following means:

— An externally defined context.

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

The following additional requirements apply to `track_IDs` in a Track Reference Type Box of `reference_type 'cvar'`:

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

b) If the `track_ID` does correspond to a `track_ID` of a track in the same file, the corresponding track shall be a variant track which complies with 8.2.1.

Variant track references defined for a media track shall be defined in Variant Constructor search order. The Variant Processor will process variant tracks according to this order when searching for an accessible Variant Constructor.

8.2.3 Variant Metadata Sample Entry

8.2.3.1 Syntax

```
class VariantMetaDataSampleEntry() extends MetaDataSampleEntry ('cvar') {
      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;
}
```

8.2.3.2 Semantics

`variant_constructor_scheme_type` – shall be set to the four character code defining the protection scheme applied to Variant Constructors in the track, per 8.3.4.

`variant_constructor_scheme_version` – shall be set to the version of the protection scheme applied to the Variant Constructors in the track, per 8.3.4.

media_track_scheme_type – shall be set to the four character code defining the protection scheme applied to associated media track, as defined for the schema_type field in the associated media track by ISOBMFF subclause 8.12.5.3.

media_track_scheme_version – shall be set to the version of the protection scheme applied to associated media track, as defined for the scheme_version field in the associated media track by ISOBMFF subclause 8.12.5.3.

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). The IV_Size shall match the IV_Size of the associated media track.

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, per 8.3.4.

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, per 8.3.4.

8.3 Sample data

8.3.1 Variant Data

8.3.1.1 Definition

A sample in a variant track is either empty (zero size) or a VariantData() structure.

8.3.1.2 Syntax

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

8.3.1.3 Semantics

variant_list – the Variant Constructor list as defined in 8.3.2.

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.

8.3.2 Variant Constructor list

8.3.2.1 Definition

The VariantConstructorList() defines sample specific information on the location of potential Variant Constructors for Sample Variants.

Each sample definition in a variant track may have one or more Variant Constructor location entries in the VariantConstructorList(). As required in 5.2, 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.

8.3.2.2 Syntax

```
aligned(8) class 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;
}
```

8.3.2.3 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 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.

vcIV – the "Variant Constructor Initialization Vector". This field shall contain the initialization vector used for decrypting the encrypted Variant Constructor.

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 – the byte array may contain any data and be used to increase the size of the VariantConstructorList().

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

8.3.3 Variant Constructor

8.3.3.1 Syntax

```
aligned(8) class 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_track_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;
}
```

8.3.3.2 Semantics

`KID` – the Media KID. This KID shall indicate the ID of the Media Key that is used for decrypting the encrypted Sample Variant data after re-assembly of the applicable Variant Byte Ranges. Decryption occurs in accordance with the protection scheme signalled in the associated media track.

`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 associated media track.

`variant_byte_ranges_count` – shall be set to the number of Variant Byte Ranges defined for this Variant Constructor. See Clause 6 for more information.

`variant_byte_range_flags` – shall be set as follows:

0x01 encrypted — When set, the Sample Variant data referenced by the Variant Byte Range shall be encrypted with the Media Key.

0x02 double-enc — When set, the Sample Variant 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 Sample Variant 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 6, 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), that the start of Variant Byte Range group be signalled with this singular Variant Byte Range. As per 6.2, 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 This flag can be used by a Variant Processor to determine that a data error has occurred - if no Variant Byte Range is 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 track. When set to 1, the `variant_track_reference_index` indicates which variant track 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_track_reference_index` - shall either be the 1-based index (according to order of reference definition - see 8.2.2) of the track references from this variant sample track to another variant track containing the variant data to be used; or if this value is 0, the data is drawn from this variant track.

`relative_sample_number` – having found the track data source (see the data-source flag and `variant_track_reference_index` field above), this field defines which sample data source shall be used for the Variant Byte Range as follows: when set to 0, the sample data-source is the time-parallel associated sample per 8.3.5; when set to a negative value, the Nth prior sample is used; when set to positive value, the Nth succeeding sample is used.

`variant_byte_range_offset` – is the byte offset from the start of the referenced sample (original sample in the media track, the `VariantData()` that contains this Variant Constructor, or the `VariantData()` in a referenced variant track, depending on the data-source flag and `variant_track_reference_index`) 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 sample. The Variant Byte Range defined by variant_byte_range_offset and variant_byte_range_size shall only reference bytes within the referenced 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 Variant Byte Ranges in a Variant Byte Range group is the same.

`padding` – the byte array may contain any data and be used to increase the size of the Variant Constructor.

NOTE This padding can be used to obfuscate the actual size of the Variant Constructor as it is encrypted.

8.3.4 Encryption

As defined in 5.3, Variant Constructors are always encrypted, and per 8.2.3, the encryption scheme is signaled in the `VariantMetaDataSampleEntry()`. One of the following CENC encryption modes shall be used to encrypt Variant Constructors:

— AES-CTR Full Sample Encryption: signalled with a four character code value of `'cvar'` and a `scheme_version` value of 0x00010000 (Major version 1, Minor version 0) in the `VariantMetaDataSampleEntry()` per 8.2.3.

— AES-CBC-128 Full Sample Encryption: signalled using a four character code value of `scheme_type` field value of `'cva1'` and a `scheme_version` field value of 0x00010000 (Major version 1, Minor version 0) in the `VariantMetaDataSampleEntry()` per 8.2.3.

As defined in 7.3, 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 per 8.2.3, this scheme is also signalled in the `VariantMetaDataSampleEntry()`.

As defined in 7.3, bytes referenced by a Variant Byte Range may be double encrypted with a "Variant Byte Range key" and per 8.2.3, 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:

— AES-CTR Full Sample Encryption: signalled with a four character code value of `'cvar'` and a `scheme_version` value of 0x00010000 (Major version 1, Minor version 0) in the `VariantMetaDataSampleEntry()` per 8.2.3.

— AES-CBC-128 Full Sample Encryption: signalled using a four character code value of `scheme_type` field value of 'cva1' and a `scheme_version` field value of 0x00010000 (Major version 1, Minor version 0) in the `VariantMetaDataSampleEntry()` per 8.2.3.

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.

8.3.5 Association

Samples are associated as follows:

a) 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.

b) 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.

c) Samples are considered to be time-parallel as follows: If To 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 Tv and duration D, such that Tv<=To<(Tv+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 3.

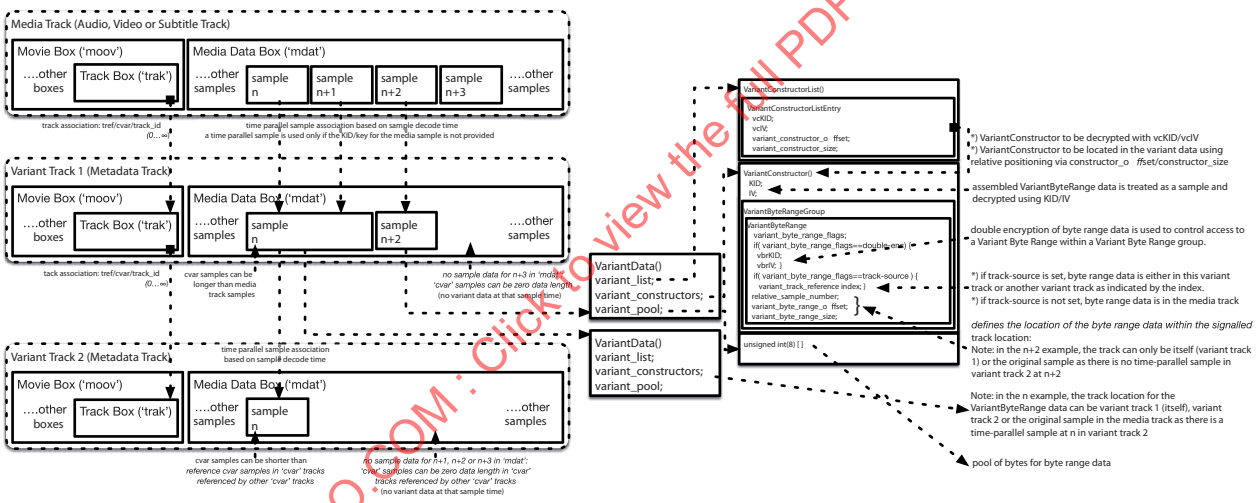

Figure 3 — Media track and variant track referencing

9 Variant Processor Model and Example (Informative)

9.1 Variant Processor Model

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

a) The data source for each sample is evaluated as follows:

1) If the decoder has access to the sample in the media track, the decoder proceeds to render the sample as per 5.2.