ISO/IEC JTC 1/SC 29/WG 03 N0471

**ISO/IEC JTC 1/SC 29/WG 03  
MPEG Systems   
Convenorship: KATS (Korea, Republic of)**

**Document type:** Output Document

**Title:** Text of CDAM ISO/IEC 14496-12:2021 AMD 1 Improved brand documentation and other improvements

**Status:** Approved

**Date of document:** 2022-02-21

**Source:** ISO/IEC JTC 1/SC 29/WG 03

**No. of pages:** 40 (with cover page)

**Email of Convenor:** young.L @ samsung . com

**Committee URL:** <https://isotc.iso.org/livelink/livelink/open/jtc1sc29wg3>

**INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC 1/SC 29/WG 03 MPEG SYSTEMS**

**ISO/IEC JTC 1/SC 29/WG 03 N0471**

**February 2022, Virtual**

|  |  |
| --- | --- |
| **Title** | **Text of CDAM ISO/IEC 14496-12:2021 AMD 1 Improved brand documentation and other improvements** |
| **Source** | **WG 03, MPEG Systems** |
| **Status** | **Approved** |
| **Serial Number** | 21187 |

*In 3.1.35 change the name of the term defined from*

**sample description**

structure which defines and describes the format of some number of samples in a track

*to*

**sample entry**

structure which defines and describes the format of some number of samples in a track

Note 1 to entry The term "sample description" has also been used, and has the same meaning.

*In 3.1.36 change*

four-character code that is either a format value of a SampleEntry directly contained in SampleDescriptionBox or a data\_format value of OriginalFormatBox

*to*

four-character code that is either a format value of a SampleEntry directly contained in SampleDescriptionBox or a data\_format value of OriginalFormatBox

*In 3.1.46 change "NOTE "1 to "Note 1"*

*In 3.1 add the following new definitions*

### **preselection**

a set of one or multiple sub tracks representing one version of the media presentation that may be selected by a user for simultaneous decoding/presentation

Ed. Note: Can “tracks” be used instead of “sub-tracks”?

### **random access**

decoding of an elementary stream starting from a particular access unit without decoding of any access unit in the elementary stream earlier in decoding order

Note 1 to entry Sync samples and SAPs provide random accessing capabilities.

### **sub track**

selectable entity within a track

[Ed. Note: Should the new term be 'sub-track', 'media components' or something else?]

*In 3.2 add the following abbreviated terms*

EDRAP extended dependent random access point

*In 4.2.1 change*

The following basic field types are defined. In these definitions, null-terminated means that the last character of a string is Unicode NUL, and hence an empty string is represented by a single Unicode NUL. Some fields using these types may restrict the characters permitted.

*to*

The following basic field types are defined. In these definitions, null-terminated means that the string is terminated by the first Unicode NUL, which shall be present, and hence an empty string is represented by a single Unicode NUL. Some fields using these types may restrict the characters permitted.

*In 4.2.2 change*

size is an integer that specifies the number of bytes in this box, including all its fields and contained boxes; if size is 1 then the actual size is in the field largesize; if size is 0, then this box shall be in a top-level container, and be the last box in that container (typically, a file or data object delivered over a protocol), and its payload extends to the end of that container (normally only used for a MediaDataBox)

*to*

size is an integer that specifies the number of bytes in this box, including all its fields and contained boxes; if size is 1 then the actual size is in the field largesize; if size is 0, then this box shall be a top-level box (i.e. not contained in another box), and be the last box in its 'file', and its payload extends to the end of that enclosing 'file'. This is normally only used for a MediaDataBox.

*In 4.2.2 after*

The semantics of these two fields are:

version is an integer that specifies the version of this format of the box.

flags is a map of flags

*add*

When the syntax specifies version = 0, it means that this document only defines the syntax for version 0 of the FullBox. When “= 0” is not present, it means different values are possible and the syntax has actual differences depending on the actual version used. Derived specifications are not permitted to define a new version of a box defined in this document.

When the syntax specifies flags = 0 (or sometimes simply “0” for the flags), it means that this document does not define flag values for this FullBox. When “= 0” is not present, it means different flag values are defined. Future editions of this document may define additional flags, but if a new value is introduced in a later edition that impacts the box syntax, a new version will be used. Derived specifications should follow this practice. As a consequence, if a reader supports a version of a FullBox, it can and should keep parsing the box even if it does not recognize a flag value. Writers should not write flag values that they do not understand.

*In section 4.3.1, File-type box, replace*

This box shall be placed as early as possible in the file (e.g. after any obligatory signature, but before any significant variable-size boxes such as a MovieBox, MediaDataBox, or FreeSpaceBox). It identifies which specification is the ‘best use’ of the file, and a minor version of that specification; and also a set of other specifications to which the file complies. Readers implementing this format should attempt to read files that are marked as compatible with any of the specifications that the reader implements. Any incompatible change in a specification should therefore register a new ‘brand’ identifier to identify files conformant to the new specification.

*with*

This box shall be placed as early as possible in the file (e.g. after any obligatory signature, but before any significant variable-size boxes such as a MovieBox, MediaDataBox, or FreeSpaceBox). It identifies which specification is the ‘best use’ of the file (the major\_brand), and a minor version of that specification; and also a set of other specifications to which the file complies (the compatible\_brands); the major\_brand should be repeated in the compatible\_brands list. Readers implementing this format should attempt to read files that are marked as compatible with any of the specifications that the reader implements. Any incompatible change in a specification should therefore register a new ‘brand’ identifier to identify files conformant to the new specification.

*Add subclause 6.2.3 as follows:*

**6.2.3 Data offsets**

ISO base media files contain either or both of:

* tracks, which are defined as a timed sequence of related samples,
* items, which do not require timed processing.

Each sample or item is mapped to a data reference entry that is present in the DataReferenceBox contained in the MediaInformationBox of the track or the MetaBox where the item is defined, respectively. The data reference entry specifies the container for the sample or item data among the following:

* the same file where this data reference entry is present;
* a resource identified by a given URL or URN;
* the IdentifiedMediaDataBox that contains a given identifier value;
* the IdentifiedMediaDataBox that contains the identifier value that is equal to sequence\_number of the MovieFragmentHeaderBox of the movie fragment containing the sample.

The location of samples and item data within its container is indicated through data offsets, which may be provided or inferred, for example based on the ChunkOffsetBox, TrackFragmentHeaderBox, or ItemLocationBox. Table XX summarizes the different types of data offsets.

Table XX — Data offset types

|  |  |  |  |
| --- | --- | --- | --- |
| **Type of data offset** | **Signalling** | **The origin that the data offset is relative to** | **Usage examples** |
| Movie-fragment-relative | The base-data-offset-present and default-base-is-moof flags of the TrackFragmentHeaderBox (see subclause 8.8.7.1) | The first byte of the MovieFragmentBox defining the sample (if this is the first TrackFragmentHeaderBox of this track within the movie fragment), or  the end of the data defined by the preceding track fragment of the same track (otherwise) | Delivery of segments that contain one or more movie fragments |
| File-relative | Data reference entry of type 'url ' or 'urn ', and either no movie-fragment relative offsets indicated or construction\_method equal to 0 in ItemLocationBox | The first byte of the file identified by the data reference entry | Files that do not contain movie fragments.  NOTE There might be implementations that record media data in a movie-fragmented file and use file-relative data offsets. However, movie-fragment-relative data offsets can be used for this purpose and have a benefit that they enable delivery on movie fragment basis. |
| IdentifiedMedia‌DataBox-relative | Data reference entry of type 'imdt' or 'snim' | The first byte of the payload of the referenced IdentifiedMediaDataBox | File editing (see subclause E.15.3.1), partial image file reception (see subclause E.15.3.2), delivery of the MovieFragmentBox in a separate resource from the media data it references. |
| ItemDataBox-relative | construction\_method equal to 1 in ItemLocationBox | The first byte of the payload of the referenced ItemDataBox | Items that are small in size stored in the same MetaBox that defines them. |
| Item-relative | construction\_method equal to 2 in ItemLocationBox | The first byte of the concatenated data (of all the extents) of a given item |  |

*In 6.3.2 change*

Fields shown as “template” in the box descriptions are fields which are coded with a default value unless a derived specification defines their use and permits writers to use other values than the default. If the field is used in another specification, that use must be conformant with its definition here, and the specification must define whether the use is optional or mandatory. Similarly, fields marked “pre-defined” were used in an earlier edition of this document. For both kinds of fields, if a field of that kind is not used in a specification, then it should be set to the indicated default value. If the field is not used it shall be copied un-inspected when boxes are copied, and ignored on reading.

*to*

Fields shown as “template” in the syntax are interpreted as follows:

* The value given after the equals sign in the syntax is a default value.
* Derived specifications are permitted to define additional possible values for the field, but they shall indicate when these should be used, and they shall not modify the remaining syntax of the box to depend on the value of this field.
* Writers should write this default value unless they conform to a derived specification, in which case they may write another value, as permitted by that derived specification.
* Readers should expect any value and should keep parsing the box even if they do not recognize the value.

Fields named “pre\_defined” in the syntaxes are interpreted as follows:

* They had specific semantics in an earlier edition of this document, but their semantics are no longer in use. They are considered legacy fields.
* The value given after the equals sign in the syntax is the only permitted value.
* Derived specifications are not permitted to change this value.
* Writers shall always write this value.
* Readers should expect any value, and should keep parsing the box even if they do not recognize a value. Derived specifications may require stricter behaviour from readers (e.g. requiring reading to stop and emit an error when reading a different value).

Fields name “reserved” in the syntaxes are interpreted as follows:

* They may have a default value defined in this specification (typically 0) or they may not.
* Additional values may be defined in future versions of this specification.
* Derived specifications are not permitted to define values for this field.
* This specification guarantees the following:
  + When used in a FullBox, if a new value is introduced that requires changes to the syntax of the FullBox in a non-compatible way, this change will be done together with a change of version.
  + When used in other structures, new values may be defined but these will not affect the syntax of the structure.
* Writers shall always write the default value, if specified, or shall write a value of 0, if not specified.
* Readers should expect any value, and should keep parsing the box even if they do not recognize the value

*In 6.3.4 change the row in Table 1 for stsd to read*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | stsd | \* | 8.5.2 | *sample description (codec types, initialization etc.)* |

*In 8.3.2.3 adjust the indentation of the paragraphs under width and height, and change to read:*

width and height fixed-point 16.16 values are track-dependent as follows:

For text and subtitle tracks, they may, depending on the coding format, describe the suggested size of the rendering area. For such tracks, the value 0x0 may also be used to indicate that the data may be rendered at any size, that no preferred size has been indicated and that the actual size may be determined by the external context or by reusing the width and height of another track. For those tracks, the flag track\_size\_is\_aspect\_ratio may also be used.

For non-visual tracks (e.g. audio), they should be set to zero.

For all other tracks, they specify the track's visual presentation size. These need not be the same as the pixel dimensions of the images, which is documented in the sample entries; all images in the sequence are scaled to this size, before any overall transformation of the track represented by the matrix. The pixel dimensions of the images are the default values.

*Add new subclause 8.3.3.4*

**8.3.3.4 Associated external stream track reference**

A track reference of type 'aest' (meaning "associated external stream track") may be included in a video track, referencing an associated video track.

When a video track has a track reference of type 'aest', the following applies:

* The video track should have at least one sample that contains an EDRAP picture.
* For each sample sampleA in the video track containing an EDRAP picture, there shall be one and only one sample sampleB in the associated video track that has the same decoding time as sampleA, and a number of consecutive samples in the associated video track, starting from sampleB, shall exclusively contain all the pictures, not contained in the video track that contains sampleA and that are needed when random accessing from the EDRAP picture contained in sampleA.

Every sample in the referenced track shall be identified as a sync sample. The referenced track header flags shall have track\_in\_movie and track\_in\_preview both set to 0.

A restricted scheme shall be used for each referenced track, as follows:

1. At least one sample entry type of each sample entry of the track shall be equal to 'resv'.

Note 1: 'resv' does not have to be the sample entry type of a SampleEntry directly contained in SampleDescriptionBox when the track has undergone several transformations.

1. The untransformed sample entry type is stored within an OriginalFormatBox contained in the RestrictedSchemeInfoBox.
2. The scheme\_type field in the SchemeTypeBox, which is in the RestrictedSchemeInfoBox, is equal to 'aest', indicating that a sample in the track may contain more than one coded picture.
3. Bit 0 of the flags field of the SchemeTypeBox is equal to 0, such that the value of (flags & 0x000001) is equal to 0.

*In 8.3.4.3 add the following new value,* 'pres'

track\_group\_type indicates the grouping\_type and shall be set to one of the following values, or a value registered, or a value from a derived specification or registration:

'msrc' indicates that this track belongs to a multi-source presentation. Specified in 8.3.4.4.1.

'ster' indicates that this track is either the left or right view of a stereo pair suitable for playback on a stereoscopic display. Specified in 8.3.4.4.2.

'pres' indicates that this track contributes to a preselection. Specified in 8.3.4.4.3.

The pair of track\_group\_id and track\_group\_type identifies a track group within the file. The tracks that contain a particular TrackGroupTypeBox having the same value of track\_group\_id and track\_group\_type belong to the same track group.

*Add new subclauses 8.3.4.4.3 through 8.3.4.4.9*

[Ed. Note: The following text describes the PreselectionGroupBox being inherited from TrackGroups according to 8.3.4. An alternative solution under consideration is to inherit it from Entity Grouping according to 8.18. Final location is TBD.

Comments are welcome on whether it is more beneficial to use entity grouping instead. There are two key differences between tracking grouping and entity grouping:

1) Track grouping signalling is track-level while entity grouping for this purpose would be file level. The difference leads to the following pros and cons:

A) A potential advantage of entity grouping can be same bits saving.

B) An arguable advantage of entity grouping is that less parsing is needed to figure out which tracks contribute to a preselection.

C) Track grouping has a potential advantage of being more friendly to editing operations such as removing of some tracks without the need of changing the tracking group box (unless some related parameters e.g., the number of tracks is added therein).

2) Entity grouping can also be applied to items, such as image items; track grouping cannot. If there is a use case of having items in a preselection, then this is an advantage for entity grouping.

A drawback of entity grouping is that it requires parsing the `meta` box and that a `hdlr` type would have to be defined.]

Ed. Note: For the ‘meta’ box parsing part, I think we should either add some detail on why parsing the `meta` box is bad (because using track grouping you would need to parse some other box) or drop it. For the ‘hdlr’ type, due to that a new ‘hdlr’ type was not needed for many other entity groups specified, it seems to me that this is not an issue, thus this part should be dropped.

8.3.4.4.3 Preselection group box

8.3.4.4.3.1 Definition

The presence of a TrackGroupTypeBox with track\_group\_type equal to 'pres', which is also referred to as a PreselectionGroupBox in a track indicates that this track contributes to a preselection.

The tracks that have the same value of track\_group\_id within PreselectionGroupBox are part of the same preselection.

Preselections can be qualified by language, kind or media specific attributes like audio rendering indications, object interactivity or channel layouts. Attributes signalled in a preselection box shall take precedence over attributes signalled in contributing tracks.

All attributes uniquely qualifying a preselection shall be present in at least one Preselection Box of the preselection. If present in more than one Preselection Box of the preselection, the boxes containing the qualifying attributes shall be identical.

Ed. Note: Additional clarity may be needed that a preselection is uniquely identified by its track\_group\_id, so when we talk about "attributes uniquely qualifying a preselection" it means that a qualifying box shall only be present if it's identical to the box contained in a track group with the same id. (We could note that `prsp` can differ between 2 tracks in the same preselection). We should also say that order of boxes does not matter (prsi1=elng+chnl and prsi2=chnl+elng are the same if the contents of chnl and elng are the same).

NOTE 1: Preselections group tracks of the same media type only.

Tracks containing a PreselectionGroupBox and not containing all required sub tracks for at least one preselection shall have the track\_in\_movie flag set to ‘0’ in their Track Header Boxes. This prevents players not understanding the Preselection Box from playing the track resulting in an incomplete experience.NOTE 2: It is good practice to have one track with track\_in\_movie flag set to one. This implies that this track provides at least one complete experience.

Ed. Note: Consider use case implications/constraints of the above.

8.3.4.4.3.2 Syntax

aligned(8) class PreselectionGroupBox extends TrackGroupTypeBox('pres', 0, flags)  
{  
 if (flags & 1) {  
 unsigned int(8) selection\_priority=1;  
 }  
 PreselectionInformationBox presel\_info;  
 PreselectionProcessingBox presel\_processing;  
}

8.3.4.4.3.3 Semantics

selection\_priority is an integer that declares the priority of the preselection in cases where no other differentiation such as through the media language is possible. A lower number indicates a higher priority.

presel\_info is an instance of the PreselectionInformationBox(), providing information that describes the preselection.

presel\_processing is an instance of the PreselectionProcessingBox(), providing information needed for processing the preselection.

8.3.4.4.4 Preselection information box

8.3.4.4.4.1 Definition

Box Type: 'prsi'  
Container: PreselectionGroupBox   
Mandatory: No  
Quantity: Zero or One

This Box aggregates all semantic information about the preselection.

8.3.4.4.4.2 Syntax

aligned(8) class PreselectionInformationBox   
 extends FullBox('prsi', version=0, 0 ){  
 // Boxes describing the preselection  
 unsigned int(8) numTracks;  
}

Ed. Note: Having this parameter signalled here would lose the potential advantage of using track grouping (vs using entity grouping) being more friendly to editing operations such as removing of some tracks without the need of changing the tracking group box.

8.3.4.4.4.3 Semantics

This box contains information on what experience is available when this preselection is selected.

Boxes suitable to describe a preselection include but are not limited to the following list of boxes defined in this specification:

* The audio object box (clause 8.3.4.4.5)
* The audio object selection box (clause 8.3.4.4.7)
* The extended language tag (clause 8.4.6)
* The user data box (clause 8.10.1)
* The track kind (clause 8.10.4)
* The label box (clause 8.18.4)
* The audio rendering indication (clause 8.18.5)
* The channel layout (clause 12.2.4.1)

If the user data box is contained in a PreselectionBox, then it shall not carry any of the above boxes.

numTracks declares how many tracks are contributing to the playout of the preselection. This value shall match the number of tracks containing a PreselectionGroupBox with the same track\_group\_id.

Ed. Note: Is additional clarity needed that numTracks must match the number of tracks that have the same track\_group\_id?

Note 1: not all tracks contributing to the playout of a preselection may be delivered in the same file.

Note2: The kind box might utilize the Role scheme defined in ISO/IEC 23009-1 [ref], clause 5.8.5.5 as it provides a commonly used scheme to describe characteristics of preselections.

Further media type specific boxes may be used to describe properties of the preselection.

8.3.4.4.5 AudioObjectBox

8.3.4.4.5.1 Definition

Box Type: 'aobj'  
Container: AudioObjectSelectionBox ('aosb'), Preselection Information Box ('prsi')  
Mandatory: Yes  
Quantity: Zero or more

This box aggregates all semantic information about the audio object.

8.3.4.4.5.2 Syntax

aligned(8) class AudioObjectBox extends Box(‘aobj’){}

8.3.4.4.6 AudioObjectDescriptionBox

8.3.4.4.6.1 Definition

Box Type: 'aodb'  
Container: AudioObjectBox ('aobj')   
Mandatory: Yes  
Quantity: Exactly One

This box contains information about the interactivity options of the audio object.

8.3.4.4.6.2 Syntax

aligned(8) class AudioObjectDescriptionBox extends FullBox('aodb'){  
 utf8string audioObjectTag;  
 unsigned int(1) isToggleable;  
 unsigned int(1) isDefaultEnabled;  
 unsigned int(1) isPositionInteractivityAllowed;  
 unsigned int(1) isProminenceInteractivityAllowed;  
 unsigned int(4) reserved;  
 if (isPositionInteractivityAllowed) {  
 signed int(16) minAzimuth;  
 signed int(16) maxAzimuth;  
 signed int(16) defaultAzimuth;  
 signed int(8) minElevation;   
 signed int(8) maxElevation;  
 signed int(8) defaultElevation;  
 unsigned int(8) minDistance;  
 unsigned int(8) maxDistance;  
 unsigned int(8) defaultDistance;  
 }  
 if (isProminenceInteractivityAllowed) {  
 signed double(16) minProminence;   
 signed double(16) maxProminence;  
 signed double(16) defaultProminence;   
 }  
}

8.3.4.4.6.3 Semantics

audioObjectTag is an integer that uniquely identifies the audio object that is described by this box within the preselection.

isToggleable specifies whether the described audio object can be switched on / off independently.

isDefaultEnabled specifies whether the described audio object is enabled or disabled by default.

isPositionInteractivityAllowed specifies whether the described audio object by interactively placed within the audio scene.

isProminenceInteractivityAllowed specifies whether prominence interactivity is allowed for the described audio object.

minAzimuth is a signed value in degrees, as defined for LoudspeakerAzimuth in ISO/IEC 23001-8. It specifies the minimum allowed Azimuth angle for the described audio object.

maxAzimuth is a signed value in degrees, as defined for LoudspeakerAzimuth in ISO/IEC 23001-8. It specifies the maximum allowed Azimuth angle for the described audio object.

defaultAzimuth is a signed value in degrees, as defined for LoudspeakerAzimuth in ISO/IEC 23001-8. It specifies the default Azimuth angle setting for the described audio object.

minElevation is a signed value, in degrees, as defined for LoudspeakerElevation in ISO/IEC 23001-8. It specifies the minimum allowed elevation for the described audio object.

maxElevation is a signed value, in degrees, as defined for LoudspeakerElevation in ISO/IEC 23001-8. It specifies the maximum allowed elevation for the described audio object.

defaultElevation is a signed value, in degrees, as defined for LoudspeakerElevation in ISO/IEC 23001-8. It specifies the default elevation setting for the described audio object.

minDistance specifies the minimum allowed distance of the described audio object from the listener.

maxDistance specifies the maximum allowed distance of the described audio object from the listener.

defaultDistance specifies default distance of the described audio object from the listener.

minProminence specifies the minimum allowed prominence of the described audio object in dB.

maxProminence specifies the maximum allowed prominence of the described audio object in dB.

defaultProminence specifies the default prominence setting of the described audio object in dB.

8.3.4.4.7 AudioObjectSelectionBox

8.3.4.4.7.1 Definition

Box Type: 'aosb'  
Container: Preselection Information Box ('prsi')  
Mandatory: No  
Quantity: Zero or more

The audio object selection box signals a collection of audio objects where just one of the objects can be switched on at a given time.

8.3.4.4.7.2 Syntax

aligned(8) class AudioObjectSelectionBox extends Box('aosb'){};

8.3.4.4.8 AudioObjectSelectionDescriptionBox

8.3.4.4.8.1 Definition

Box Type: 'aosd'  
Container: Audio Object Selection Box ('aosb')  
Mandatory: Yes  
Quantity: Exactly One

This box contains semantic information about the audio object selection.

8.3.4.4.8.2 Syntax

aligned(8) class AudioObjectSelectionDescriptionBox   
 extends FullBox('aosd'){   
 utf8string selectionTag;  
};

8.3.4.4.8.3 Semantics

selectionTag is a string that contains an identifier for the audio object selection. This identifier shall uniquely identify the signalled audio object selection.

8.3.4.4.9 Preselection processing box

8.3.4.4.9.1 Definition

Box Type: 'prsp'  
Container: PreselectionGroupBox  
Mandatory: Yes  
Quantity: Exactly One

This box contains information how the tracks contributing to the preselection shall be processed. Media type specific boxes may be used to describe further processing.

8.3.4.4.9.2 Syntax

aligned(8) class PreselectionProcessingBox   
 extends FullBox('prsp', version=0, flags ){  
 utf8string preselection\_tag;  
 if (flags & 1) {  
 unsigned int(8) order=0;  
 }  
 unsigned int(8) track\_order;  
 unsigned int(1) sample\_merge\_flag;  
 unsigned int(7) reserved;  
 // further attributes and Boxes defining additional processing of  
 // the track contributing to the preselection  
}

8.3.4.4.9.3 Semantics

preselection\_tag is a codec specific value that a playback system can provide to a decoder to uniquely identify one out of several preselections in the media.

order specifies the conformance rules for Representations in Adaptation Sets within the Preselection according to ISO/IEC 23009-1 [ref], from the following enumerated set:

0: undefined

1: time-ordered

2: fully-ordered

track\_order defines the order of this track for the merging process described below.

sample\_merge\_flag equal to 1 indicates that this track is enabled to be merged with another track as described below.

Ed. Note: Order vs track\_order under investigation.

Tracks contributing to a selected preselection and having the sample\_merge\_flag equal to 1 shall merge their samples according to the respective media type.

If the media type does not specify such a process, contributing samples may be appended to the samples of the track with the next lower track\_order. If the generated output samples from this merging process shall be embedded into a new track, this track shall be conformant to a media type derived from the base media type.

Ed. Note: The above paragraph needs rephrasing for clarity.

Note: for MPEG-H 3D Audio, this process is defined in ISO/IEC 23008-3, clause 14.6.

*In 8.4.2.2 change*

bit(1) pad = 0;

*to*

bit(1) reserved = 0;

*Change 8.4.6.1:*

Box Type: 'elng'  
Container: MediaBox, PreselectionInformationBox, AudioObjectBox  
Mandatory: No  
Quantity: Zero or one

*In 8.5.2.1 change*

If the ‘format’ field of a SampleEntry is unrecognized, neither the sample description itself, nor the associated media samples, shall be decoded.

*to*

If the ‘format’ field of a SampleEntry is unrecognized, neither the sample entry itself, nor the associated media samples, shall be decoded.

*In 8.5.2.3 change*

data\_reference\_index is an integer that contains the index of the DataEntry to use to retrieve data associated with samples that use this sample description. Data entries are stored in DataReferenceBoxes. The index ranges from 1 to the number of data entries.

*to*

data\_reference\_index is an integer that contains the index of the DataEntry to use to retrieve data associated with samples that use this sample entry. Data entries are stored in DataReferenceBoxes. The index ranges from 1 to the number of data entries.

*In 8.7.2.1 change*

The data reference object contains a table of data references (normally URLs) that declare the location(s) of the media data used within the presentation. The data reference index in the sample description ties entries in this table to the samples in the track. A track may be split over several sources in this way.

*to*

The data reference object contains a table of data references (normally URLs) that declare the location(s) of the media data used within the presentation. The data reference index in the sample entry ties entries in this table to the samples in the track. A track may be split over several sources in this way.

*In 8.7.4.1 change*

Samples within the media data are grouped into chunks. Chunks can be of different sizes, and the samples within a chunk can have different sizes. This table can be used to find the chunk that contains a sample, its position, and the associated sample description.

*to*

Samples within the media data are grouped into chunks. Chunks can be of different sizes, and the samples within a chunk can have different sizes. This table can be used to find the chunk that contains a sample, its position, and the associated sample entry.

*In 8.8.4.1 change*

The movie fragments extend the presentation in time. They provide the information that would previously have been in the MovieBox. The actual samples are in MediaDataBoxes, as usual, if they are in the same file. The data reference index is in the sample description, so it is possible to build incremental presentations where the media data is in files other than the file containing the MovieBox.

*to*

The movie fragments extend the presentation in time. They provide information that, when fragments are not used, would be in the MovieBox. The actual samples are in MediaDataBoxes, as usual, if they are in the same file. The data reference index is in the sample entry, so it is possible to build incremental presentations where the media data is in files other than the file containing the MovieBox.

*In subclause 8.8.7.1, replace the following text and renumber the NOTE in this subclause:*

The data origin that the base data offset is relative and the value of base\_data\_offset, when not present, are inferred as follows:

* If the base-data-offset-present flag is equal to 0 and the default-base-is-moof flag is equal to 1, base\_data\_offset is inferred to be equal to 0 and is relative to the first byte of the MovieFragmentBox containing this box.
* Otherwise, if base-data-offset-present flag is equal to 0 and the default-base-is-moof flag is equal to 0 and this TrackFragmentBox is not the first TrackFragmentBox of the same track in the containing MovieFragmentBox, base\_data\_offset is inferred to be equal to 1 and is relative to the end of the data defined by the preceding track fragment of the same track.
* Otherwise, if base-data-offset-present flag is equal to 0 and the default-base-is-moof flag is equal to 0 and the referenced data reference entry is DataEntryImdaBox or DataEntrySeqNumImdaBox, base\_data\_offset is inferred to be equal to 0 and is relative to the first byte of the payload of the IdentifiedMediaDataBox corresponding to the data reference entry.
* Otherwise, if the referenced data reference entry is DataEntryImdaBox or DataEntrySeqNumImdaBox, the base data offset is relative to the first byte of the payload of the IdentifiedMediaDataBox corresponding to the data reference entry.
* Otherwise, the base data offset is relative to the file identified by the referenced data reference entry.

*with:*

The data origin that base\_data\_offset is relative to and the value of base\_data\_offset, when not present, are inferred as specified in Table XX.

**Table XX — Derivation of the value and data origin of base\_data\_offset**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Conditions**  A: base-data-offset-present flag  B: default-base-is-moof flag  C: the referenced data reference entry is DataEntryImdaBox or DataEntrySeqNumImdaBox  D: this TrackFragmentBox is the first TrackFragmentBox in the containing MovieFragmentBox | | | |  |  |
| **A** | **B** | **C** | **D** | **base\_data\_offset value** | **Data origin of base\_data\_offset** |
| 0 | 1 | true or false | true or false | base\_data\_offset is inferred to be equal to 0 | base\_data\_offset is relative to the first byte of the MovieFragmentBox containing this box |
| 0 | 0 | true or false | false | base\_data\_offset is inferred to be equal to 1 | base\_data\_offset is relative to the end of the data defined by the preceding track fragment |
| 0 | 0 | true | true | base\_data\_offset is inferred to be equal to 0 | base\_data\_offset is relative to the first byte of the payload of the IdentifiedMediaDataBox corresponding to the data reference entry |
| 0 | 0 | false | true | base\_data\_offset is inferred to be equal to 0 | base\_data\_offset is relative to the first byte of the MovieFragmentBox containing this box |
| 1 | 0 or 1 | true | true or false | base\_data\_offset is present | base\_data\_offset is relative to the first byte of the payload of the IdentifiedMediaDataBox corresponding to the data reference entry |
| 1 | 0 or 1 | false | true or false | base\_data\_offset is present | base\_data\_offset is relative to the file identified by the referenced data reference entry |

NOTE 1 base-data-offset-present flag is advised to be set equal to 1 only if the file identified by the referenced data reference entry is not meant to be delivered with movie fragments in separate segments or the referenced data reference entry is DataEntryImdaBox or DataEntrySeqNumImdaBox.

When base-data-offset-present flag is equal to 1 and the referenced data reference entry is of type 'url ' with (entry\_flags & 1) equal to 1 (indicating that the media data is in the same file as the DataReferenceBox containing the referenced data reference entry), this TrackFragmentHeaderBox shall be present in the same file as the MovieBox.

*In subclause 8.8.7.1, replace the following paragraph:*

0x000001 base-data-offset-present: indicates the presence of the base-data-offset field. This provides an explicit anchor for the data offsets in each track run (see below).

*with:*

0x000001 base-data-offset-present: indicates the presence of the base\_data\_offset field. This provides an explicit anchor for the data offsets in each track run (see below).

*In subclause 8.8.8.1, replace the following paragraph:*

If the data-offset is not present, then the data for this run starts immediately after the data of the previous run, or at the base-data-offset defined by the track fragment header if this is the first run in a track fragment, If the data-offset is present, it is relative to the base-data-offset established in the track fragment header.

*with the following two paragraphs:*

When data\_offset is not present, the data for this run starts immediately after the data of the previous run, or at base\_data\_offset defined by the track fragment header if this is the first run in a track fragment.

When data\_offset is present, it is relative to base\_data\_offset established in the track fragment header.

*In subclause 8.8.8.3, replace the following paragraph:*

data\_offset is added to the implicit or explicit data\_offset established in the track fragment header.

*with:*

data\_offset is added to the implicit or explicit base\_data\_offset established in the track fragment header.

*In 8.8.12.1 add to the end of the paragraph*

If the time expressed in the TrackFragmentBaseMediaDecodeTimeBox exceeds the sum of the sample durations of the samples in the preceding movie and movie fragments, then the duration of the last sample preceding this track fragment is extended such that the sum now equals the time given in this box. In this way, it is possible to generate a fragment containing a sample when the time of the next sample is not yet known, by assigning it a small or even zero sample duration, that is then overridden by the time expressed in this box in the following fragment.

*this sentence*

The time expressed in the TrackFragmentBaseMediaDecodeTimeBox shall not be less than the sum of the sample durations of the samples in the preceding movie and movie fragments.

*In subclause 8.8.14, replace the following paragraph:*

When sample auxiliary information (‎8.7.8 and ‎8.7.9) is present in the MovieFragmentBox, the offsets in the SampleAuxiliaryInformationOffsetsBox are treated the same as the data\_offset in the TrackRunBox, that is, they are relative to any base data offset established for that track fragment.

*with:*

When sample auxiliary information (‎8.7.8 and ‎8.7.9) is present in the MovieFragmentBox, the offsets in the SampleAuxiliaryInformationOffsetsBox are treated the same as the data\_offset in the TrackRunBox, that is, they are relative to any base\_data\_offset value established for that track fragment.

*In 8.9.3.1 add at the end of the section*

If version equals 3, the sample group description describes essential information for the associated samples, and file processors shall not attempt to decode any track for which unrecognized sample group descriptions marked as essential are present.

*In 8.10.1.1 change*

Box Type: 'udta'  
Container: MovieBox, TrackBox, MovieFragmentBox, ~~or~~ TrackFragmentBox or PreselectionInformationBox  
Mandatory: No  
Quantity: Zero or one

*In 8.10.4.1 change*

Box Type: 'kind'  
Container: UserDataBox of the corresponding TrackBox, AudioObjectBox or a PreselectionInformationBox  
Mandatory: No  
Quantity: Zero or one

*In 8.11.1.1 change*

The MetaBox is required to contain a HandlerBox indicating the structure or format of the MetaBox contents.

All other contained boxes are specific to the format specified by the HandlerBox.

*to*

When the MetaBox does not contain a PrimaryItemBox, it is required to contain a HandlerBox indicating the structure or format of the MetaBox contents. When the MetaBox does contain a PrimaryItemBox, if that item has a HandlerProperty and there is a HandlerBox present, they shall identify the same handler type. Otherwise, when the primary data is identified by a primary item without a HandlerProperty, and that primary item has an item information entry with an item\_type, the handler type may be the same as the item\_type.

When a HandlerBox is present, all other contained boxes are specific to the format specified by that HandlerBox.

*Change 8.11.1.2 to contain:*

aligned(8) class MetaBox (handler\_type)  
 extends FullBox('meta', version = 0, 0) {  
 HandlerBox(handler\_type) theHandler; // optional  
 PrimaryItemBox primary\_resource; // optional  
 DataInformationBox file\_locations; // optional  
 ItemLocationBox item\_locations; // optional  
 ItemProtectionBox protections; // optional  
 ItemInfoBox item\_infos; // optional  
 IPMPControlBox IPMP\_control; // optional  
 ItemReferenceBox item\_refs; // optional  
 ItemDataBox item\_data; // optional  
 Box other\_boxes[]; // optional  
}

*Delete clause 8.11.1.3*

*Replace the entire contents of 8.11.3.1:*

Box Type: 'iloc'  
Container: MetaBox  
Mandatory: No  
Quantity: Zero or one

The ItemLocationBox provides a directory of resources in this or other files, by locating their container, their offset within that container, and their length. Using byte offsets and lengths enables common handling of this data, even by systems which do not understand the particular metadata system (handler) used. For example, a system might integrate all the externally referenced metadata resources into one place, re-adjusting offsets and references accordingly.

The box starts with three or four values, specifying the size in bytes of the offset field, length field, base\_offset field, and, in versions 1 and 2 of this box, the item\_reference\_index fields, respectively. These values shall be from the set {0, 4, 8}.

The construction\_method field indicates the ‘construction method’ for the item:

1. file\_offset: by absolute byte offsets into the file or the payload of IdentifiedMediaDataBox referenced by data\_reference\_index;  (construction\_method == 0)
2. idat\_offset: by byte offsets into the ItemDataBox in the same MetaBox; neither the data\_reference\_index nor item\_reference\_index fields are used; (construction\_method == 1)
3. item\_offset: by byte offset into the items indicated by the item\_reference\_index field, which is only used (currently) by this construction method. (construction\_method == 2).

The item\_reference\_index is only used for the method item\_offset; it indicates the 1-based index of the item reference with referenceType 'iloc' linked from this item. If index\_size is 0, then the value 1 is implied; the value 0 is reserved.

Items may be stored fragmented into extents, e.g. to enable interleaving. An extent is a contiguous subset of the bytes of the resource; the resource is formed by concatenating the extents in the order specified in this box. If only one extent is used (extent\_count = 1) then either or both of the offset and length may be implied:

 If the offset is not identified (the field has a length of zero), then the beginning of the source (offset 0) is implied.

 If the length is not specified, or specified as zero, then the entire length of the source is implied. References into the same file as this structure-data, or items divided into more than one extent, should have an explicit offset and length, or use a MIME type requiring a different interpretation of the file, to avoid infinite recursion.

The size of the item is the sum of the extent lengths.

NOTE 1 Extents can be interleaved with the chunks defined by the sample tables of tracks.

The offsets are relative to a data origin. That origin is determined as follows:

1. when the MetaBox is in a Movie Fragment, and the construction\_method specifies a file offset, and the data reference indicates ‘same file’, the data origin is the first byte of the enclosing MovieFragmentBox (as for the default-base-is-moof flag in the TrackFragmentHeaderBox);
2. when the construction\_method specifies a file offset and the data reference indicates DataEntryImdaBox or DataEntrySeqNumImdaBox, the data origin is the first byte of the payload of the corresponding IdentifiedMediaDataBox;
3. in all other cases when the construction\_method specifies a file offset, the data origin is the beginning of the file identified by the data reference;
4. when the construction\_method specifies offsets into the ItemDataBox, the data origin is the beginning of data[] in the ItemDataBox;
5. when the data reference specifies another item, the data origin is the first byte of the concatenated data (of all the extents) of that item;

NOTE 2 There are offset calculations in other parts of this file format based on the beginning of a box header; in contrast, item data offsets are calculated relative to the box payload.

The data\_reference\_index may take the value 0, indicating a reference into the same file as this structure-data, or an index into the data references in the DataInformationBox in the containing MetaBox, with value 1 indicating the first entry in the data reference list.

Some referenced data may itself use offset/length techniques to address resources within it (e.g. an MP4 file might be ‘included’ in this way). Normally such offsets in the item itself are relative to the beginning of the containing file. The field ‘base offset’ provides an additional offset for offset calculations within that contained data. For example, if an MP4 file is included within a file formatted to this document, then normally data-offsets within that MP4 section are relative to the beginning of file; the base offset adds to those offsets.

If an item is constructed from other items, and those source items are protected, the offset and length information apply to the source items after they have been de-protected. That is, the target item data is formed from unprotected source data.

For maximum compatibility, version 0 of this box should be used in preference to version 1 with construction\_method==0, or version 2 when possible. Similarly, version 2 of this box should only be used when support for large item\_ID values (exceeding 65535) is required or expected to be required.

NOTE 3 When construction\_method 2 is used and one item needs to have an offset of 0 into another item, the base\_offset field is set to 0.

*with:*

Box Type: 'iloc'  
Container: MetaBox  
Mandatory: No  
Quantity: Zero or one

The ItemLocationBox provides a directory of resources in this or other files, by locating their container, their offset within that container, and their length. Using byte offsets and lengths enables common handling of this data, even by systems which do not understand the particular metadata system (handler) used. For example, a system might integrate all the externally referenced metadata resources into one place, re-adjusting offsets and references accordingly.

The box starts with three or four values, specifying the size in bytes of the offset field, length field, base\_offset field, and, in versions 1 and 2 of this box, the item\_reference\_index fields, respectively. These values shall be from the set {0, 4, 8}.

The construction\_method field indicates the ‘construction method’ for the item:

1. file\_offset: by absolute byte offsets into the file or the payload of IdentifiedMediaDataBox referenced by data\_reference\_index;  (construction\_method == 0)
2. idat\_offset: by byte offsets into the ItemDataBox in the same MetaBox; neither the data\_reference\_index nor item\_reference\_index fields are used; (construction\_method == 1)
3. item\_offset: by byte offset into the items indicated by the item\_reference\_index field, which is only used (currently) by this construction method. (construction\_method == 2).

The item\_reference\_index is only used for the method item\_offset; it indicates the 1-based index of the item reference with referenceType 'iloc' linked from this item. If index\_size is 0, then the value 1 is implied; the value 0 is reserved.

The data\_reference\_index may take the value 0, indicating a reference into the same file as this MetaBox, or an index into the data references in the DataInformationBox in the containing MetaBox, with value 1 indicating the first entry in the data reference list.

NOTE 1 A reference to the ‘same file’ can also be coded by setting data\_reference\_index to a non-zero value pointing to a data reference entry of type 'urn ' with the flag 0x000001 set. However, using data\_reference\_index 0 is recommended.

Items may be stored fragmented into extents, e.g. to enable interleaving. An extent is a contiguous subset of the bytes of the resource identified by an offset and a length; the resource is formed by concatenating the extents in the order specified in this box.

NOTE 2 Extents can be interleaved with the chunks defined by the sample tables of tracks.

The offsets are relative to a data origin. That origin is determined as follows:

1. when construction\_method = 0 (file offset):
2. when the data reference points to the ‘same file’ and when the MetaBox is in a Movie Fragment, the data origin is the first byte of the enclosing MovieFragmentBox (as for the default-base-is-moof flag in the TrackFragmentHeaderBox
3. when the data reference indicates DataEntryImdaBox or DataEntrySeqNumImdaBox, the data origin is the first byte of the payload of the corresponding IdentifiedMediaDataBox;
4. in all other cases, the data origin is the beginning of the file identified by the data reference;
5. when the construction\_method==1 (offsets into the ItemDataBox), the data origin is the beginning of data[] in the ItemDataBox;

NOTE 3 There are offset calculations in other parts of this file format based on the beginning of a box header; in contrast, item data offsets are calculated relative to the box payload.

1. when construction\_method==2 (offsets into another item), the data origin is the first byte of the concatenated data (of all the extents) of that item;

A value of 0 for extent\_length is interpreted as follows:

1. when construction\_method = 0 (file offset):
2. when the data reference points to the ‘same file’, the length of the extent is assumed to be the length of the data between the offset (if specified) or the origin (if not specified), and the end of the file.
3. when the data reference indicates a DataEntryImdaBox or DataEntrySeqNumImdaBox, the extent\_length is assumed to be the length of the data between the offset (if specified) or the origin (if not specified), and the end of the payload of the corresponding IdentifiedMediaDataBox.
4. in all other cases, the extent\_length is assumed to be the length of the referenced file between the offset (if specified) or the origin (if not specified) and the end of the file.
5. when the construction\_method==1 (offsets into the ItemDataBox), the length of the extent is assumed to be the length of the data between the offset (if specified) or the origin (if not specified) and the end of the payload of the ItemDataBox;
6. when construction\_method==2 (offsets into another item), the length of the extent is assumed to be the size of the item minus the offset (if specified)

If only one extent is used (extent\_count = 1) then either or both of the offset and length may be implied, i.e. by setting offset\_size or length\_size to 0.

References into the same file as this structure-data, or items divided into more than one extent, should have an explicit offset and length, or use a MIME type requiring a different interpretation of the file, to avoid infinite recursion.

Some referenced data may itself use offset/length techniques to address resources within it (e.g. an MP4 file might be ‘included’ in this way). Normally such offsets in the item itself are relative to the beginning of the containing file. The field ‘base offset’ provides an additional offset for offset calculations within that contained data. For example, if an MP4 file is included within a file formatted to this document, then normally data-offsets within that MP4 section are relative to the beginning of file; the base offset adds to those offsets.

If an item is constructed from other items, and those source items are protected, the offset and length information apply to the source items after they have been de-protected. That is, the target item data is formed from unprotected source data.

For maximum compatibility, version 0 of this box should be used in preference to version 1 with construction\_method==0, or version 2 when possible. Similarly, version 2 of this box should only be used when support for large item\_ID values (exceeding 65535) is required or expected to be required.

NOTE 3 When construction\_method 2 is used and one item needs to have an offset of 0 into another item, the base\_offset field is set to 0.

*In 8.11.3.3 change the two field definitions*

data-reference-index is either zero ('this file') or an index, with value 1 indicating the first entry, into the data references in the DataInformationBox.

extent\_length provides the absolute length in bytes of this metadata item extent. If length\_size is 0, extent\_length takes the value 0. If the value is 0, then length of the extent is the length of the entire referenced container.

*to*

data\_reference\_index is either zero ('this file') or an index, with value 1 indicating the first entry, into the data references in the DataInformationBox.

extent\_length provides the absolute length in bytes of this metadata item extent. If length\_size is 0, extent\_length takes the value 0.

*In 8.11.4.1 change*

For a given handler, the primary data may be one of the referenced items when it is desired that it be stored elsewhere, or divided into extents; or the primary metadata may be contained in the MetaBox (e.g. in an XMLBox). Either this box shall occur, or there shall be a box within the MetaBox (e.g. an XMLBox) containing the primary information in the format required by the identified handler.

*to*

The primary data may be one of the referenced items when it is desired that it be stored elsewhere, or divided into extents; or the primary metadata may be contained in the MetaBox (e.g. in an XMLBox). Either this box shall occur, or there shall be a box within the MetaBox (e.g. an XMLBox) containing the primary information in the format required by the identified handler.

*In 8.11.5.1 add after*

The ItemProtectionBox provides an array of item protection information, for use by the ItemInfoBox.

*the following*

The ProtectionSchemeInfoBoxes shall not contain an OriginalFormatBox when present in an ItemProtectionBox.

*In 8.11.14.3, change*

property\_index is either 0 indicating that no property is associated (the essential indicator shall also be 0), or is the 1-based index (counting all boxes, including FreeSpace boxes) of the associated property box in the ItemPropertyContainerBox contained in the same ItemPropertiesBox.

*to*

property\_index is either 0 indicating that no property is associated (the essential indicator shall also be 0), or is the 1-based index (counting all boxes, including FreeSpace boxes) of the associated property box in the ItemPropertyContainerBox contained in the same ItemPropertiesBox. property\_index shall not be greater than the number of boxes contained in the associated ItemPropertyContainerBox.

*Add the following as clause 8.11.16*

**8.11.16 Handler property**

**8.11.16.1 Definition**

Box Type: 'hdlp'  
Property Type: Descriptive item property  
Container: ItemPropertyContainerBox  
Mandatory: No  
Quantity: zero or more

HandlerProperty provides a mapping of a media handler with an item in a MetaBox. Items that are alternatives of each other shall have the same handler property, or none.

**8.11.16.2 Syntax**

aligned(8) class HandlerProperty extends ItemFullProperty('hdlp', version=0, flags=0) {  
 unsigned int(32) handler\_type;  
}

**8.11.16.3 Semantics**

handler\_type is a four-character-code which corresponds to a media handler type. When the HandlerProperty is associated with the primary item, handler\_type of the HandlerProperty shall be equal to the handler\_type of the MetaBox. When no specific handler type needs to be signalled for an item, the HandlerProperty for the item may be absent or the handler\_type may be 'null'.

*In 8.12.1 change*

1. The four character code of the sample description is replaced with a four character code indicating protection encapsulation: these codes vary only by media-type. For example, 'mp4v' is replaced with 'encv' and 'mp4a' is replaced with 'enca'.
2. A ProtectionSchemeInfoBox (*defined below*) is added to the sample description, leaving all other boxes unmodified.

*to*

1. The four character code of the sample entry is replaced with a four character code indicating protection encapsulation: these codes vary only by media-type. For example, 'mp4v' is replaced with 'encv' and 'mp4a' is replaced with 'enca'.
2. A ProtectionSchemeInfoBox (*defined below*) is added to the sample entry, leaving all other boxes unmodified.

*In 8.12.2.2 change*

aligned(8) class ProtectionSchemeInfoBox(fmt) extends Box('sinf') {  
 OriginalFormatBox(fmt) original\_format;  
  
 SchemeTypeBox scheme\_type\_box; // optional  
 SchemeInformationBox info; // optional  
}

*to*

aligned(8) class ProtectionSchemeInfoBox(fmt) extends Box('sinf') {  
 OriginalFormatBox(fmt) original\_format;  
 // mandatory for sample protection,   
 // shall not be present for item protection  
  
 SchemeTypeBox scheme\_type\_box; // optional  
 SchemeInformationBox info; // optional  
}

*In 8.12.3.1 change*

The OriginalFormatBox contains the four character code of the original un-transformed sample description.

*to*

The OriginalFormatBox contains the four character code of the original un-transformed sample entry.

*In 8.15.2 change*

A restricted sample entry is defined as a sample entry on which the following transformation procedure has been applied:

1. The four character code of the sample entry is replaced by a new sample entry code 'resv' meaning restricted video.
2. A RestrictedSchemeInfoBox is added to the sample description, leaving all other boxes unmodified.
3. The original sample entry type is stored within an OriginalFormatBox contained in the RestrictedSchemeInfoBox.

*to*

A restricted sample entry is defined as a sample entry on which the following transformation procedure has been applied:

1. The four character code of the sample entry is replaced by a sample entry code as defined in Table XX below.
2. A RestrictedSchemeInfoBox is added to the sample entry, leaving all other boxes unmodified.
3. The original sample entry type is stored within an OriginalFormatBox contained in the RestrictedSchemeInfoBox.

**Table XX — Restricted sample-entry codes**

|  |  |  |
| --- | --- | --- |
| **Stream (Track) Type** | **Sample-Entry Code** | **SampleEntry Class** |
| Video | resv | VisualSampleEntry |
| Audio | resa | AudioSampleEntry or AudioSampleEntryV1 |
| Metadata | resm | MetaDataSampleEntry |
| Text | rest | SimpleTextSampleEntry |
| Subtitle | resu | XMLSubtitleSampleEntry |
| Systema | ress |  |
| Font | resf | FontSampleEntry |
| Haptics | resp | HapticSampleEntry |
| Volumetric visual | res3 | VolumetricVisualSampleEntry |
| a System streams are defined in ISO/IEC 14496-14[22]. | | |

*In 8.15.4.2.2 change*

aligned(8) class StereoVideoBox extends FullBox('stvi', version = 0, 0)  
{  
 template unsigned int(30) reserved = 0;  
 unsigned int(2) single\_view\_allowed;  
 unsigned int(32) stereo\_scheme;  
 unsigned int(32) length;  
 unsigned int(8)[length] stereo\_indication\_type;  
 Box[] any\_box; // optional  
}

*to*

aligned(8) class StereoVideoBox extends FullBox('stvi', version = 0, 0)  
{  
 template unsigned int(30) reserved = 0;  
 unsigned int(2) single\_view\_allowed;  
 unsigned int(32) stereo\_scheme;  
 unsigned int(32) length;  
 unsigned int(8) stereo\_indication\_type[length];  
 Box[] any\_box; // optional  
}

*In 8.15.4.2.3 change*

4: a value of VideoFramePackingTypeas defined in ISO/IEC 23091-2.

*to*

4,5: values of VideoFramePackingType, QuincunxSamplingFlag. PackedContentInterpretationType as defined in ISO/IEC 23091-2.

*In 8.15.4.2.3 replace*

The following applies when the StereoVideoBox is used:

* In the TrackHeaderBox
  + width and height specify the visual presentation size of a single view after unpacking.
* In the SampleDescriptionBox

*with*

The following applies when the StereoVideoBox is used:

* In the TrackHeaderBox
  + width and height specify the visual presentation size of a single view after unpacking.
* In a SampleEntry in the SampleDescriptionBox
* Ed. Note: Is this location permitted?

*In 8.17.2 replace*

A CompleteTrackInfoBox is added to the sample description, leaving all other boxes unmodified.

*with*

A CompleteTrackInfoBox is added to the sample entry, leaving all other boxes unmodified.

*Add new 8.18.4 and 8.18.5*

8.18.4 Label box

8.18.4.1 Definition

Box Type: 'labl'  
Container: UserDataBox of the corresponding TrackBox, AudioObjectBox, AudioObjectSelectionBox, or PreselectionInformationBox   
Mandatory: No  
Quantity: Zero or more

Labels provide the ability to annotate data structures in an ISOBMFF file to provide a description of the context of the entity to which the label is assigned. Such labels may for example be used by playback clients to provide a selection choice to the user. The label may also be used for simple annotation in another context.

Multiple Labels can be used to provide the textual description. To annotate the entity to a multilingual audience, the annotation can be provided in multiple languages.

When is\_group\_label is equal to 1 this indicates that the label text in this box specifies a summary or title of all labels with the same label\_id. This may be used as the title on a selection menu containing a collection of labels.

8.18.4.2 Syntax

aligned(8) class LabelBox   
 extends FullBox('labl', version=0, 0 ){  
 unsigned int(1) is\_group\_label = 0;  
 bit(7) reserved = 0;  
 unsigned int(16) label\_id = 0;  
 utf8string language;  
 utf8string label;  
}

8.18.4.3 Semantics

is\_group\_label specifies if the label contains a summary label for a group of labels.

label\_id is an integer that contains an identifier for the label. Labels with the same value belong to a label group. The default value of zero indicates that the label does not belong to any label group.

language contains an IETF BCP 47 compliant language tag string, such as "en-US", "fr-FR", or "zh-CN", the language being the language the label is targeted at.

label contains the textual description.

8.18.5 Audio rendering indication box

8.18.5.1 Definition

Box Type: 'ardi'  
Container: PreselectionInformationBox   
Mandatory: No  
Quantity: Zero or one

The audio rendering indication box contains a hint for a preferred reproduction channel layout.

8.18.5.2 Syntax

aligned(8) class AudioRenderingIndicationBox   
 extends FullBox('ardi', version=0, 0 ){  
 unsigned int(8) audio\_rendering\_indication = 0;  
}

8.18.5.3 Semantics

audio\_rendering\_indication contains a hint for a preferred reproduction channel layout, coded according to table 2.

Table 1 — Coding of audio rendering indication

|  |  |
| --- | --- |
| **audio\_rendering\_indication** | **Description** |
| 0 | no preference given for the reproduction channel layout |
| 1 | preferred reproduction channel layout is stereo |
| 2 | preferred reproduction channel layout is two-dimensional (e.g. 5.1 multi-channel) |
| 3 | preferred reproduction channel layout is three-dimensional |
| 4 | content is pre-rendered for consumption with headphones |
| 5 to 255 | reserved for future use |

*Amended table 1 from section 8.14.4 removed in this contribution – unchanged compared to N20538.*

*Change the titles of 9.1.2, 9.2.3, 9.3.3, 9.4.1.2, 9.4.2.3, 9.4.3.2, 9.4.4.3 to* "Sample entry format"

*In 9.1.3.3 change*

The sampledescription mode allows sending of sample descriptions (which would contain elementary stream descriptors), by reference, as part of an RTP packet. The index is the index of a SampleEntry in a SampleDescriptionBox, and the offset is relative to the beginning of that SampleEntry.

*to*

The sampledescription mode allows sending of sample entries (which would contain elementary stream descriptors), by reference, as part of an RTP packet. The index is the index of a SampleEntry in a SampleDescriptionBox, and the offset is relative to the beginning of that SampleEntry.

*In 9.1.4.3 change*

At the track level, the structure is similar; however, we already know that this track is an RTP hint track, from the sample description. Therefore the child box merely specifies the description format.

*to*

At the track level, the structure is similar; however, we already know that this track is an RTP hint track, from the sample entry. Therefore the child box merely specifies the format.

*In 9.3.2.3 change*

The optional TSTimingBox in the sample description can be used

*to*

The optional TSTimingBox in the sample entry can be used

*In 9.3.2.5 replace*

It is recommended that the PSI/SI be in the Sample Description so that true random-access with just the media data is possible.

*with*

It is recommended that the PSI/SI be in the sample entry so that true random-access with just the media data is possible.

*In 9.3.2.6 change*

The format of the reception hint samples is indicated by the sample description for the reception hint track.

*to*

The format of the reception hint samples is indicated by the sample entry for the reception hint track.

*In 9.3.3.1 change*

The sample description for an MPEG2-TS reception hint track contains all static metadata that describe the stream or a portion thereof, especially the PSI/SI tables. MPEG-2 TS reception hint tracks use an entry-format in the sample description of 'rm2t' (which indicates *MPEG-2 transport stream*). The entry-format for MPEG2‑TS server hint tracks is 'sm2t'.

*to*

The sample entry for an MPEG2-TS reception hint track contains all static metadata that describe the stream or a portion thereof, especially the PSI/SI tables. MPEG-2 TS reception hint tracks use an entry-format in the sample entry of 'rm2t' (which indicates *MPEG-2 transport stream*). The entry-format for MPEG2‑TS server hint tracks is 'sm2t'.

*In 9.3.4.1 change*

Each MPEG-2 TS packet in the sample may be preceded with a preheader (precedingbytes), or followed by a posttrailer (trailingbytes), as detailed in the Sample Description Format. The size of the preheader and the posttrailer are specified by precedingbyteslen and trailingbyteslen, respectively, in the sample description to allow compact sample tables with fewer chunks.

*to*

Each MPEG-2 TS packet in the sample may be preceded with a preheader (precedingbytes), or followed by a posttrailer (trailingbytes), as detailed in the sample entry. The size of the preheader and the posttrailer are specified by precedingbyteslen and trailingbyteslen, respectively, in the sample entry to allow compact sample tables with fewer chunks.

*In 9.4.1.2 change*

The entry-format in the sample description for the RTP reception hint tracks is 'rrtp'. The syntax of the sample entry is the same as for RTP server hint tracks having the entry-format 'rtp '.

class ReceivedRtpHintSampleEntry() extends HintSampleEntry ('rrtp') {  
 uint(16) hinttrackversion = 1;  
 uint(16) highestcompatibleversion = 1;  
 uint(32) maxpacketsize;  
}

The entry-format identifier in the sample description of the RTP reception hint track is different from the entry-format in the sample description of the RTP server hint track, in order to avoid using an RTP reception hint track that contains errors as a valid server hint track.

…

The SSRC value shall equal the SSRC value in the header of all recorded SRTP packets described by the sample description.

*to*

The entry-format in the sample entry for the RTP reception hint tracks is 'rrtp'. The syntax of the sample entry is the same as for RTP server hint tracks having the entry-format 'rtp '.

class ReceivedRtpHintSampleEntry() extends HintSampleEntry ('rrtp') {  
 uint(16) hinttrackversion = 1;  
 uint(16) highestcompatibleversion = 1;  
 uint(32) maxpacketsize;  
}

The entry-format identifier in the sample entry of the RTP reception hint track is different from the entry-format in the sample entry of the RTP server hint track, in order to avoid using an RTP reception hint track that contains errors as a valid server hint track.

…

The SSRC value shall equal the SSRC value in the header of all recorded SRTP packets described by the sample entry.

*In 9.4.2.3 change*

The entry-format in the sample description for the RTCP reception hint tracks is 'rtcp'. It is otherwise identical in structure to the sample entry format for RTP. There are no defined boxes for the additionaldata field.

*to*

The entry-format in the sample entry for the RTCP reception hint tracks is 'rtcp'. It is otherwise identical in structure to the sample entry format for RTP. There are no defined boxes for the additionaldata field.

*Change the title of 9.4.3.2.1 to "Sample entry" and in it change*

The sample description format for SRTP reception hint tracks is identical to that for RTP reception hint tracks with the exception that the sample entry name is changed from 'rrtp' to 'rsrp' and that it may contain additional boxes:

class ReceivedSrtpHintSampleEntry() extends HintSampleEntry ('rsrp') {  
 uint(16) hinttrackversion = 1;  
 uint(16) highestcompatibleversion = 1;  
 uint(32) maxpacketsize;  
}

Fields and boxes are identical to those of the ReceivedRtpHintSampleEntry ('rrtp'). The addtionaldata[] of each sample description entry of a SRTP reception hint track shall contain exactly one ReceivedSsrcBox.

*to*

The sample entry format for SRTP reception hint tracks is identical to that for RTP reception hint tracks with the exception that the sample entry name is changed from 'rrtp' to 'rsrp' and that it may contain additional boxes:

class ReceivedSrtpHintSampleEntry() extends HintSampleEntry ('rsrp') {  
 uint(16) hinttrackversion = 1;  
 uint(16) highestcompatibleversion = 1;  
 uint(32) maxpacketsize;  
}

Fields and boxes are identical to those of the ReceivedRtpHintSampleEntry ('rrtp'). The addtionaldata[] of each sample entry of a SRTP reception hint track shall contain exactly one ReceivedSsrcBox.

*In 9.4.4.3 change*

The entry-format in the sample description for the SRTCP reception hint tracks is 'stcp'.

*to*

The entry-format in the sample entry for the SRTCP reception hint tracks is 'stcp'.

*Replace 10.8.1 with the following:*

A dependent random access point (DRAP) sample is a sample after which all samples in decoding order and in output order can be correctly decoded if the closest SAP sample of type 1, 2, or 3 preceding the DRAP sample is available for reference.

NOTE 1 The closest SAP sample can be a Sync sample or marked by the SAP sample group.

NOTE 2 DRAP samples can only be used in combination with SAP samples of type 1, 2 and 3. This is in order to enable the functionality of creating a decodable sequence of samples by concatenating the preceding SAP sample with the DRAP sample and the samples following the DRAP sample in decoding order and in output order.

*Add new subclause 10.11*

**10.11 Extended DRAP (EDRAP) sample group**

**10.11.1 Definition**

This sample group is similar to the DRAP sample group as specified in subclause 10.8; however, it enables more flexible cross-RAP referencing.

An EDRAP sample is a sample after which all samples in decoding order and in output order can be correctly decoded if the closest SAP sample of type 1, 2, or 3 preceding the EDRAP sample and zero or more other identified EDRAP samples earlier in decoding order than the EDRAP sample are available for reference.

NOTE: Similarly as for DRAP samples, EDRAP samples can only be used in combination with SAP samples of type 1, 2 and 3.

**10.11.2 Syntax**

class VisualEdrapEntry()  
extends VisualSampleGroupEntry('edrp') {  
 unsigned int(3) edrap\_type;  
 unsigned int(3) num\_ref\_edrap\_pics;  
 unsigned int(26) reserved = 0;  
 for(i=0; i<num\_ref\_edrap\_pics; i++)  
 unsigned int(16) ref\_edrap\_idx\_delta[i];  
}

**10.11.3 Semantics**

edrap\_type is a non-negative integer. When edrap\_type is in the range of 1 to 3 it indicates the SAP\_type (as specified in Annex I) that the EDRAP sample would have corresponded to, had it not depended on the closest preceding SAP or other EDRAP samples. Other type values are reserved.

num\_ref\_edrap\_pics indicates the number of other EDRAP samples that are earlier in decoding order than the EDRAP sample and are needed for reference to be able to correctly decode the EDRAP sample and all samples following the EDRAP sample in both decoding and output order when starting decoding from the EDRAP sample.

reserved shall be equal to 0. The semantics of this subclause only apply to sample group description entries with reserved equal to 0. Parsers shall allow and ignore sample group description entries with reserved greater than 0 when parsing this sample group.

ref\_edrap\_idx\_delta[i] indicates the difference between the sample group index (i.e., the index to the list of all samples in this sample group in decoding order) of this EDRAP sample and the sample group index of the i-th RAP sample that is earlier in decoding order than the EDRAP sample and is needed for reference to be able to correctly decode the EDRAP sample and all samples following the EDRAP sample in both decoding and output order when starting decoding from this EDRAP sample. The value 1 indicates that the i-th RAP sample is the latest RAP sample in the sample group and preceding this EDRAP sample in decoding order, the value 2 indicates that the i-th RAP sample is the second latest RAP sample in the sample group and preceding this EDRAP sample in decoding order, and so on.

*Add new subclause 10.12*

**10.12 Essential descriptions hierarchy sample grouping**

**10.12.1 Definition**

The essential descriptions hierarchy sample group ('esgh') indicates the processing order of the essential sample group descriptions applying to a given sample. This sample group description is an essential sample group description and shall use version 3. It shall be present if at least one essential sample group description with grouping\_type other than 'esgh' is present.

Each essential sample group description, except the essential descriptions hierarchy sample group itself, shall be listed in the EssentialDescriptionsHierarchyEntry.

The grouping\_type\_paramater for an essential descriptions hierarchy sample group description is not defined, and its value shall be set to 0.

The syntax of EssentialDescriptionsHierarchyEntry is the same for all media types.

Samples associated with essential sample groups shall use a restricted sample entry indicating the original media type (e.g. 'resv', 'resa') with a scheme\_type equal to 'essg'. In a sample entry, there shall be at most one sample entry transformation with a scheme\_type equal to 'essg'. If such a transformation is present:

* The transformation shall be the first sample entry transformation,
* There shall be either no other sample entry transformations, or at most one sample entry transformation of type protection, as defined in 8.12.1 (e.g., 'encv')

The transformations given in sample\_group\_description\_type are listed in the order in which a file reader shall apply each transformation: any sample processing described by a sample group of type sample\_group\_description\_type[i] shall be applied before any sample processing described by a sample group of type sample\_group\_description\_type[i+1].

In the sample\_group\_description\_type list, the following transformation values are reserved:

* 'stsd' : indicates the position of the decoding process in the transformation chain.
* 'cenc' : indicates the position of the protection process in the transformation chain.

If 'stsd' is absent from the list of sample\_group\_description\_type, all listed transformations shall apply to decoded samples. If 'cenc' is present in the list, 'stsd' shall be present.

**10.12.2 Syntax**

class EssentialDescriptionsHierarchyEntry ()  
extends SampleGroupDescriptionEntry ('esgh')  
{  
 unsigned int(32) num\_groupings;  
 unsigned int(32) sample\_group\_description\_type[num\_groupings];  
}

**10.12.3 Semantics**

num\_groupings indicates the number of essential sample group description types listed.

sample\_group\_description\_type indicates the four-character code of the essential sample group description (i.e. with version 3) or reserved transformation values applying to the associated samples.

*In 11 change*

As defined above, the Sample Description format may be extended with optional or required boxes. The usual syntax for doing this would be to define a new box with a specific name, extending (for example) Visual Sample Entry, and containing new boxes.

*to*

As defined above, a sample entry may be extended with optional or required boxes. The usual syntax for doing this would be to define a new box with a specific name, extending (for example) VisualSampleEntry, and containing new boxes.

*In 12.1.3.3 change*

width and height are the maximum visual width and height of the stream described by this sample description, in pixels

*to*

width and height are the maximum visual width and height of the stream described by this sample entry, in pixels

*In 12.1.4.2 change*

class CleanApertureBox extends Box('clap'){  
 unsigned int(32) cleanApertureWidthN;  
 unsigned int(32) cleanApertureWidthD;  
  
 unsigned int(32) cleanApertureHeightN;  
 unsigned int(32) cleanApertureHeightD;  
  
  
 unsigned int(32) horizOffN;  
 unsigned int(32) horizOffD;  
  
  
 unsigned int(32) vertOffN;  
 unsigned int(32) vertOffD;  
  
}

*to*

class CleanApertureBox extends Box('clap'){  
 unsigned int(32) cleanApertureWidthN;  
 unsigned int(32) cleanApertureWidthD;  
  
 unsigned int(32) cleanApertureHeightN;  
 unsigned int(32) cleanApertureHeightD;  
  
  
 signed int(32) horizOffN;  
 unsigned int(32) horizOffD;  
  
  
 signed int(32) vertOffN;  
 unsigned int(32) vertOffD;  
  
}

*In 12.2.4.1 change*

Box Type: 'chnl'  
Container: Audio sample entry, AudioObjectBox or PreselectionInformationBox  
Mandatory: No  
Quantity: Zero or one

*In 12.4.1 change*

Hint tracks are used to describe elementary stream data in the file. Each protocol or each family of related protocols has its own hint track format. A server hint track format and a reception hint track format for the same protocol are distinguishable from the associated four character code of the sample description entry. In other words, a different four character code is used for a server hint track and a reception hint track of the same protocol. The syntax of the server hint track format and the reception hint track format for the same protocol should be the same or compatible so that a reception hint track can be used for re-sending of the stream provided that the potential degradations of the received streams are handled appropriately. Most protocols will need only one sample description format for each track.

Servers find their hint tracks by first finding all hint tracks, and then looking within that set for server hint tracks using their protocol (sample description format). If there are choices at this point, then the server chooses on the basis of preferred protocol or by comparing features in the hint track header or other protocol-specific information in the sample descriptions. Particularly in the absence of server hint tracks, servers may also use reception hint tracks of their protocol. However, servers should handle potential degradations of the received stream described by the used reception hint track appropriately.

*to*

Hint tracks are used to describe elementary stream data in the file. Each protocol or each family of related protocols has its own hint track format. A server hint track format and a reception hint track format for the same protocol are distinguishable from the associated four character code of the sample entry. In other words, a different four character code is used for a server hint track and a reception hint track of the same protocol. The syntax of the server hint track format and the reception hint track format for the same protocol should be the same or compatible so that a reception hint track can be used for re-sending of the stream provided that the potential degradations of the received streams are handled appropriately. Most protocols will need only one sample entry format for each track.

Servers find their hint tracks by first finding all hint tracks, and then looking within that set for server hint tracks using their protocol (identified by the sample entry). If there are choices at this point, then the server chooses on the basis of preferred protocol or by comparing features in the hint track header or other protocol-specific information in the sample entries. Particularly in the absence of server hint tracks, servers may also use reception hint tracks of their protocol. However, servers should handle potential degradations of the received stream described by the used reception hint track appropriately.

*In 12.4.4.1 change*

For hint tracks, the sample description contains appropriate declarative data for the streaming protocol being used, and the format of the hint track. The definition of the sample description is specific to the protocol.

The ‘protocol’ and ‘codingname’ fields are registered identifiers that uniquely identify the streaming protocol or compression format decoder to be used. A given protocol or codingname may have optional or required extensions to the sample description (e.g. codec initialization parameters). All such extensions shall be within boxes; these boxes occur after the required fields. Unrecognized boxes shall be ignored.

*to*

For hint tracks, the sample entry contains appropriate declarative data for the streaming protocol being used, and the format of the hint track. The definition of the sample entry is specific to the protocol.

The protocol (codingname) field is a registered identifier that uniquely identifies the streaming protocol or compression format decoder to be used. A given protocol may have optional or required extensions to the sample entry (e.g. codec initialization parameters). All such extensions shall be within boxes; these boxes occur after the required fields. Unrecognized boxes shall be ignored.

*In A.4 change*

Each track has one or more **sample description**s; each sample in the track is tied to a description by reference.

to

Each track has one or more **sample entries**; each sample in the track is tied to an entry by reference.

*In A.10 change*

The basic 'shape' of the movie is set in initial MovieBox: the number of tracks, the available sample descriptions, width, height, composition, and so on.

*to*

The basic 'shape' of the movie is set in initial MovieBox: the number of tracks, the available sample entries, width, height, composition, and so on.

*In B.2.2 change*

These operations might include the obvious reading tracks, finding the data and timing for samples, and their sample description and track type, and so on.

*to*

These operations might include the obvious reading tracks, finding the data and timing for samples, and their sample entry and track type, and so on.

*In Annex D replace*

File format sample description and sample format identifiers (also known as codec names).

*with*

File format sample entry and sample format identifiers (also known as codec names).

*In E.2 change*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | stsd | *sample descriptions (codec types, initialization etc.)* |

*to*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | stsd | *sample description (codec types, initialization etc.)* |

*In E.10, replace the bullet "Recognizing incomplete tracks" with*

* Recognizing incomplete tracks by detecting the following sample entries for incomplete tracks: 'icpv', 'icpa', 'icpt', 'icps', 'icph', 'icpp', 'icp3' and 'icpm'.

NOTE The process of detecting when a track becomes incomplete (before the transformation specified in subclause 8.17.2) and handling incomplete tracks in playback are outside the scope of this specification.

*In E.14, after*

The brand 'isob' requires support for all features of the 'isoa' brand.

*insert*

Support for the following boxes is required under this brand:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | ttyp |  |  |  |  |  | *track type of the track* |
|  |  |  | brnd |  |  |  |  | *brand property* |

Insert at the beginning of E.15.2

Support for the following boxes is required under this brand:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| imda |  |  |  |  |  |  |  | *identified media data* |
|  |  |  | imdt |  |  |  |  | *data entry of imda* |
|  |  |  | snim |  |  |  |  | *data entry sequence number for imda* |

*In section K.3 replace bullet 5 with*

5. When dataFormat indicates a transformed media track:

1. If the transformation type indicates an essential sample group (scheme\_type equal to 'essg'), the value of the codecs MIME parameter is appended by the four-character codes listed in the Essential Descriptions Hierarchy sample description, from the first entry up to but excluding the first occurrence of 'stsd' or 'cenc'. A dot ('.') shall be used to separate the four-character codes.
2. Otherwise, the value of the codecs MIME parameter is appended by the scheme\_type four-character code contained in the SchemeTypeBox of schemeInfoContainerBox.

*Change the contents of K.4 from*

For any file format based on this file format, the 'profiles' parameter, if usedshall list exactly the major-brand, followed by the compatible-brands, as listed in the FileTypeBox ('ftyp') or SegmentTypeBox ('styp'). The major-brand shall be first, and may be removed from the compatible-brands list.

NOTE This document requires that the major brand be repeated in the compatible-brands, but this requirement is relaxed in the 'profiles' parameter for compactness.

An example might be profiles="mp41,isom,qvXt", indicating that MPEG-4 version 1 is the major-brand and preferred use, that the file is compatible with the version of the base file format identified by 'isom', and that it is also compatible with the specification/profile 'qvXt' (whatever that may be).

*to*

For any file format based on this file format, the 'profiles' parameter, if used, shall contain a list, separated by commas:

1. The list shall include the major-brand, followed by the compatible-brands, as listed in the FileTypeBox ('ftyp') or SegmentTypeBox ('styp'). The major-brand shall be first, and may be removed from the compatible-brands list;
2. The list should include the brands from each TypeCombinationBox in the top-level ExtendedTypeBox; in the list, the brands from a single TypeCombinationBox shall be separated by "+".

NOTE This document recommends that the major brand be repeated in the compatible-brands, but this requirement is relaxed in the 'profiles' parameter for compactness.

An example might be profiles="qvXt,isoc,iso8+comp", indicating that 'qvXt' (whatever that may be) is the major-brand and preferred use, that the file is compatible with the version of the base file format identified by 'isoc', and that it is compatible with the combination of 'iso8' and 'comp'.

*Add K.6*

**K.6 Use of the 'essential' parameter**

For files containing essential sample group descriptions, the ‘essential’ parameter, when used, is composed of one or more comma-separated essential hierarchy descriptions.

Each essential hierarchy description is composed of one or more four-character code of essential sample group descriptions, separated with a dot.

* If the ‘codecs’ parameter includes description of the transformation used, the listed four-character codes shall be the ones listed in the Essential Descriptions Hierarchy sample group description, in the same order, from the first code following the last occurrence of 'stsd' until the last listed code.
* Otherwise, the listed four-character codes shall be the ones listed in the Essential Descriptions Hierarchy sample group description in the same order.

Example:

An HEVC sample is encrypted by means other than CENC, signaled through an essential sample group of type 'FOOv'. The resulted decoded sample shall have a post-processing filter applied, signaled through an essential sample group of type 'BARv'. The EssentialDescriptionsHierarchyEntry will list the transformations as ['FOOv', 'stsd', 'BARv'].

The ‘codecs’ and ‘essential’ mime type sub-parameter may be:

codecs=resv.FOOv.hvc1.1.6.L186.80

essential=BARv

or

codecs=hvc1.1.6.L186.80

essential=FOOv.stsd.BARv