**ISO/IEC 14496-12:2024/AMD 2:202x(E)**

ISO/IEC JTC1/SC 29

Secretariat: JISC

**Information technology — Coding of audio-visual objects — Part 12: ISO base media file format — Amendment 2: Tools for enhanced CMAF and DASH integration**

WD stage

**Copyright notice**

This ISO document is a working draft or committee draft and is copyright-protected by ISO. While the reproduction of working drafts or committee drafts in any form for use by participants in the ISO standards development process is permitted without prior permission from ISO, neither this document nor any extract from it may be reproduced, stored or transmitted in any form for any other purpose without prior written permission from ISO.

Requests for permission to reproduce this document for the purpose of selling it should be addressed as shown below or to ISO's member body in the country of the requester:

ISO copyright office

Case postale 56 • CH-1211 Geneva 20

Tel. + 41 22 749 01 11

Fax + 41 22 749 09 47

E-mail copyright@iso.org

Web www.iso.org

Reproduction for sales purposes may be subject to royalty payments or a licensing agreement.

Violators may be prosecuted.

Contents

[Foreword iv](#_Toc181533028)

[1 Add restrictions on 'altr' groups 1](#_Toc181533029)

[2 New switching group box 2](#_Toc181533030)

[3 New screen target orientation 4](#_Toc181533031)

[4 Forbid other MetaBox versions and flags 6](#_Toc181533032)

[5 Explicitly mention preprocessing 6](#_Toc181533033)

[6 Additional track reference types 7](#_Toc181533034)

[7 An entry for encs 7](#_Toc181533035)

[8 FullBox definition headers 7](#_Toc181533036)

[9 Missing definitions 8](#_Toc181533037)

[6.5.37 Track groups 8](#_Toc181533038)

[6.5.38 Sample Description Boxes 9](#_Toc181533039)

[6.5.39 Video media 11](#_Toc181533040)

[10 Item\_name clarifications 13](#_Toc181533041)

[11 On codecs string 13](#_Toc181533042)

[12 ExternalTracks 14](#_Toc181533043)

[13 Change to ‘kind’ box containers 17](#_Toc181533044)

[14 Sidx change 18](#_Toc181533045)

[15 Unif clarifications 19](#_Toc181533046)

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1.  In particular the different approval criteria needed for the different types of documents 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](http://www.iso.org/directives) or [www.iec.ch/members\_experts/refdocs](http://www.iec.ch/members_experts/refdocs)).

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](http://www.iso.org/patents)) or the IEC list of patent declarations received (see [patents.iec.ch](https://patents.iec.ch/)).

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

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL:  [www.iso.org/iso/foreword.html](http://www.iso.org/iso/foreword.html). In the IEC, see [www.iec.ch/understanding-standards](https://www.iec.ch/understanding-standards)

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

A list of all parts in the ISO/IEC 14496 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at [www.iso.org/members.html](https://www.iso.org/members.html) and [www.iec.ch/national-committees](http://www.iec.ch/national-committees).

**Information technology — Coding of audio-visual objects — Part 12: ISO base media file format — Amendment 2: Tools for enhanced CMAF and DASH integration**

# Add restrictions on 'altr' groups

In clause 8.18.3.1, change:

'altr': The items and tracks mapped to this grouping are alternatives to each other, and only one of them should be played (when the mapped items and tracks are part of the presentation; e.g. are displayable items or tracks) or processed by other means (when the mapped items or tracks are not part of the presentation; e.g. are metadata). A player should select the first entity from the list of entity\_id values that it can process (e.g. decode and play for mapped items and tracks that are part of the presentation) and that suits the application needs. Any entity\_id value shall be mapped to only one grouping of type 'altr'. An alternate group of entities consists of those items and tracks that are mapped to the same entity group of type 'altr'.

To:

'altr': The items and tracks mapped to this grouping are alternatives to each other, and only one of them should be played (when the mapped items and tracks are part of the presentation; e.g. are displayable items or tracks) or processed by other means (when the mapped items or tracks are not part of the presentation; e.g. are metadata). A player should select the first entity from the list of entity\_id values that it can process (e.g. decode and play for mapped items and tracks that are part of the presentation) and that suits the application needs. Any entity\_id value shall be mapped to only one grouping of type 'altr'. An alternate group of entities consists of those items and tracks that are mapped to the same entity group of type 'altr'. None of the entity\_id values in an 'altr' group shall map to another 'altr' group. An 'altr' group shall not contain both entity groups and items.

In clause 8.18.3.3, change:

entity\_id is resolved to an item, when an item with item\_ID equal to entity\_id is present in the hierarchy level (file, movie or track) that contains the GroupsListBox, or to a track, when a track with track\_ID equal to entity\_id is present and the GroupsListBox is contained in the file level.

To:

entity\_id is resolved to an item, when an item with item\_ID equal to entity\_id is present in the hierarchy level (file, movie or track) that contains the GroupsListBox, or to a track, when a track with track\_ID equal to entity\_id is present and the GroupsListBox is contained in the file level. entity\_id shall not have the same value as group\_id.

# New switching group box

In clause 8.18.3, change:

'prsl': The tracks mapped to this grouping are belonging to a preselection as specified in 8.18.4.1.

NOTE EntityToGroupBox can have grouping\_type specific extensions.

To:

'prsl': The tracks mapped to this grouping belong to a preselection as specified in 8.18.4.1.

'swit': The tracks mapped to this grouping belong to a switching group as specified in 8.18.4.2

NOTE EntityToGroupBox can have grouping\_type specific extensions.

Add the following new subclause after subclause 8.18.4.1:

**x.x.x.x Switching group box**

**x.x.x.x.1 Definition**

Box Type: 'swit'  
Container: GroupsListBox in a MetaBox on movie level  
Mandatory: No  
Quantity: Zero or more

The SwitchingGroupBox is used to facilitate the generation of adaptive streaming manifests or descriptions such as DASH MPD. Generating DASH or CMAF groupings, such as CMAF Switching Sets or DASH Adaptation Sets, from a set of ISOBMFF tracks may require out-of-band knowledge, e.g. knowing which tracks contain the same source content meant to be used in adaptive streaming switching, or may require deep parsing of the tracks to determine if a decoder can decode all the tracks only with the sample description of a single track. This box allows signaling of generic properties used in adaptive streaming such as switching, time alignment or initialization characteristics, where the precise semantics of the properties are deferred to DASH or CMAF, and identified by identifiers defined in these specifications.

**x.x.x.x.2 Syntax**

aligned(8) class SwitchingGroupBox

extends EntityToGroupBox('swit', version=0, flags)

{

unsigned int(1) switch\_flag;

unsigned int(1) timed\_aligned\_flag;

unsigned int(2) init\_type;

unsigned int(4) reserved;

if (flags & 0x001000) utf8string tag;

if (flags & 0x002000) utf8string int(32) structural\_brand;

if (flags & 0x004000) utf8string int(32) mediaprofile\_brand;

Box boxes[]; // optional other boxes e.g. ExtendedLanguageBox

}

Editor’s note: We should consider using the ‘flags’ of the box instead of defining bit fields in the box payload.

**x.x.x.x.3 Semantics**

switch\_flag equal 1 indicates that the track of this group are alternative encodings of the same source content intended for adaptive streaming switching. The normative requirements applying to tracks belonging to such group are defined by DASH or CMAF and identified by the structural brand and/or media profile brand fields.

time\_aligned\_flag equal 1 indicates the tracks of this group have some timed alignment characteristics. The normative requirements applying to tracks belonging to such group are defined by DASH and CMAF and identified by the structural brand and/or the media profile brand fields.

init\_type with the following values:

* 0: The entity which its entity\_id is first listed in this box can be used to initialize a decoder for decoding any track that directly or indirectly belongs to this group.
* 1: Every track directly or indirectly belonging to this group can be used to initialize a decoder for decoding any track that directly or indirectly belongs to this group.
* 2: reserved
* 3: reserved

tag specifies additional information about the entity group which may be used for selection purposes . Derived specifications define the use of this field. For MPEG-H Audio the value of this field shall contain the whitespace-separated list of mae\_GroupIDs that are contained in the described switching group.

structural\_brand specifies an identifier defined in derived specifications that corresponds to structural constraints of all direct and indirect entities of this group.

mediaprofile\_brand specifies the media profile brand that all direct and indirect entities of this group conform to.

boxes is an array of boxes providing information about the group that can be used to generate DASH or CMAF groupings. Boxes suitable include but are not limited to the following list of boxes defined in this document:

* + ExtendedLanguageBox (subclause 8.4.6)
  + UserDataBox (subclause 8.10.1)
  + KindBox (subclause 8.10.4)
  + LabelBox (subclause 8.10.5)

Editor’s note: Changing the name is under consideration. One candidate:

AdaptiveStreamingEntityGrouping 'aseg'

# New screen target orientation

In clause 12.1.3.2, change:

class VisualSampleEntry(codingname) extends SampleEntry (codingname)  
{  
 unsigned int(16) pre\_defined = 0;  
 const unsigned int(16) reserved = 0;  
 unsigned int(32) pre\_defined[3] = 0;  
 unsigned int(16) width;  
 unsigned int(16) height;  
 template unsigned int(32) horizresolution = 0x00480000; // 72 dpi  
 template unsigned int(32) vertresolution = 0x00480000; // 72 dpi  
 const unsigned int(32) reserved = 0;  
 template unsigned int(16) frame\_count = 1;  
 uint(8) compressorname[32];  
 template unsigned int(16) depth = 0x0018;  
 int(16) pre\_defined = -1;  
 // other boxes from derived specifications  
 CleanApertureBox clap; // optional  
 PixelAspectRatioBox pasp; // optional  
}

To:

class VisualSampleEntry(codingname) extends SampleEntry (codingname)  
{  
 unsigned int(16) pre\_defined = 0;  
 const unsigned int(16) reserved = 0;  
 unsigned int(32) pre\_defined[3] = 0;  
 unsigned int(16) width;  
 unsigned int(16) height;  
 template unsigned int(32) horizresolution = 0x00480000; // 72 dpi  
 template unsigned int(32) vertresolution = 0x00480000; // 72 dpi  
 const unsigned int(32) reserved = 0;  
 template unsigned int(16) frame\_count = 1;  
 uint(8) compressorname[32];  
 template unsigned int(16) depth;  
 int(16) pre\_defined = -1;  
 // other boxes from derived specifications  
 CleanApertureBox clap; // optional  
 PixelAspectRatioBox pasp; // optional  
 // other optional boxes  
 ScreenOrientationBox ornt; // optional

}

Editors’ Note: lisiting the optional boxes is under discussion: whether to list them here, or whether in the definitions of the box, list the boxes in which they can be included. More input is welcomed.

In clause 12.1.3.3, change:

resolution fields give the resolution of the image in pixels-per-inch, as a fixed 16.16 number

frame\_count indicates how many frames of compressed video are stored in each sample. The default is 1, for one frame per sample; it may be more than 1 for multiple frames per sample

compressorname is a name, for informative purposes. It is formatted in a fixed 32-byte field, with the first byte set to the number of bytes to be displayed, followed by that number of bytes of displayable data encoded using UTF-8, and then padding to complete 32 bytes total (including the size byte). The field may be set to 0.

depth takes one of the following values

0x0018 – images are in colour with no alpha  
width and height are the maximum visual width and height of the stream described by this sample entry, in pixels

To:

resolution fields give the resolution of the image in pixels-per-inch, as a fixed 16.16 number

frame\_count indicates how many frames of compressed video are stored in each sample. The default is 1, for one frame per sample; it may be more than 1 for multiple frames per sample

compressorname is a name, for informative purposes. It is formatted in a fixed 32-byte field, with the first byte set to the number of bytes to be displayed, followed by that number of bytes of displayable data encoded using UTF-8, and then padding to complete 32 bytes total (including the size byte). The field may be set to 0.

depth takes one of the following values

0x0018 – the video sequence is in colour with no alpha

0x0028 – the video sequence is in grayscale with no alpha

0x0020 – the video sequence has alpha (gray or colour)

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

Editors’ Note: Other boxes that are allowed to be inserted in a VisualSampleEntry need a definition table (as we define for all other boxes) which will tell information about the Container, Quantity, Presence (mandatory vs not) etc.

Add the following new subclause after subclause 12.1.9:

12.1.10 Screen Orientation Box

12.1.10.1 Definition

When a video track is intended for a specific screen orientation for consumption, this creator’s intent is signalled using the ScreenOrientationBox. A player is expected to detect the current screen orientation of the device and then select an appropriate track based on this information.

When multiple video tracks are alternative of the same content but for different screen orientations, those tracks may be grouped in the same 'altr' entity group.

A given video track can be suitable for multiple screen orientations.

12.1.10.2 Syntax

[Ed. (MH): Why does 'ornt' extend a Box rather than a FullBox?]

aligned(8) class ScreenOrientationBox extends Box('ornt')   
{  
 bit(1) target\_screen\_orientation\_landscape;  
 bit(1) target\_screen\_orientation\_portrait;  
 bit(1) target\_screen\_orientation\_square;  
 bit(5) reserved;  
}

12.1.10.3 Semantics

target\_screen\_orientation\_landscape equal to 1 indicates that landscape is a suitable screen orientation, target\_screen\_orientation\_portrait equal to 1 indicates that portrait is a suitable screen orientation and target\_screen\_orientation\_square equal to 1 indicates that square is a suitable screen orientation.

# Forbid other MetaBox versions and flags

*In subclause 8.11.1.1, replace NOTE 2:*

NOTE 2 The MetaBox is unusual in that it is a container box yet extends FullBox, not Box.

*with the following:*

NOTE 2: In some previous editions of this document MetaBox was defined as a FullBox. Other specifications based on the box structure defined in this document use a Box instead of a FullBox for meta. Readers supporting several of these specifications need to be careful when parsing the MetaBox.

*Replace subclause 8.11.1.2 with the following:*

aligned(8) class MetaBox (handler\_type) extends Box('meta')

{

bit(32) obsolete\_full\_box\_fields = 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

ItemPropertiesBox item\_properties; // optional

ItemDataBox item\_data; // optional

GroupsListBox entity\_groups; // optional

Box other\_boxes[]; // optional

}

*Add subclause 8.11.1.3*

**8.11.1.3 Semantics**

obsolete\_full\_box\_fields A 32-bit field that replaces the version and flags fields from the FullBox definition in previous editions of this specification. The value of this field is set to 0 and has no defined semantics.

NOTE 3: future versions of this specification can only use values 0, 2, 4, or 6

# Explicitly mention preprocessing

*In clause 8.1, add the following text:*

Some parts of the file may have to be pre-processed before being compliant with this document.

NOTE An example is a CompressedMovieBox that has to be pre-processed into a MovieBox for the file to be compliant with this document. Another example is an ItemProtectionBox.

*In subclause 8.2.1.1, replace the following text:*

Box Type: 'moov'  
Container: File  
Mandatory: Yes  
Quantity: Exactly one

*with the following text:*

Box Type: 'moov'  
Container: Movie file  
Mandatory: Yes  
Quantity: Exactly

1

If an ItemProtectionBox occurs, then some or all of the metadata, including possibly the primary resource, may have been protected and be un-readable unless the protection system is taken into account. Files shall not contain a MetaBox with version or flags fields set to other values than 0. Readers shall ignore any MetaBox with the version field set to another value than 0.

2 Explicitly mention preprocessing*In clause 8.1, add the following text:*Some parts of the file may have to be pre-processed before being compliant with this document.NOTE An example is a CompressedMovieBox that has to be pre-processed into a MovieBox for the file to be compliant with this document. Another example is an ItemProtectionBox.

Box Type: 'moov'  
Container: Movie

# Additional track reference types

*Add the following 2 track reference types in clause 8.3.3.3*

|  |  |  |
| --- | --- | --- |
| — | 'adda' | Track reference for additional audio track |
| — | 'adrc' | Track reference for DRC metadata track |

# An entry for encs

*Add an entry for encs:*

Table 13 — Protected sample-entry codes

|  |  |  |
| --- | --- | --- |
| **Stream (Track) Type** | **Sample-Entry Code** | **SampleEntry Class** |
| ... | ... | ... |
| Systema | encs | MpegSampleEntry |
| ... | ... | ... |

# FullBox definition headers

*Add max version and flags to FullBox definitions:*

[Ed. Note (DP): this document lists an example on how this should be implemented. The actual implementation of this proposal needs to happen in the next edition of the ISOBMFF specification. Refer to m69015 for more details]

Box Type: 'iinf'  
Container: MetaBox  
Mandatory: No  
Quantity: Zero or one  
Max Version: 1  
Flags: None

# Missing definitions

*Add missing definitions:*

### Track groups

**8.3.4.2 Syntax**

aligned(8) class TrackGroupBox extends Box('trgr')  
{  
 ~~Box boxes[];~~  
}

...

**8.3.4.4.1 Multi-source presentation**

**Definition**

Box Type: 'msrc'  
Container: TrackGroupBox  
Mandatory: No  
Quantity: Zero or one

track\_group\_type equal to 'msrc' indicates that this track belongs to a multi-source presentation. The tracks that have the same value of track\_group\_id within a TrackGroupTypeBox of track\_group\_type 'msrc' are mapped as being originated from the same source. For example, a recording of a video telephony call may have both audio and video for both participants, and the value of track\_group\_id associated with the audio track and the video track of one participant differs from value of track\_group\_id associated with the tracks of the other participant.

**Syntax**

aligned(8) class MultiSourcePresentationBox extends TrackGroupTypeBox('msrc')   
{  
}

**8.3.4.4.2 Stereoscopic pair**

**8.3.4.4.2.1 Definition**

Box Type: 'ster'  
Container: TrackGroupBox  
Mandatory: No  
Quantity: Zero or one

TrackGroupTypeBox with track\_group\_type equal to 'ster' indicates that ...

**8.3.4.4.3 Preselection group box**

**8.3.4.4.3.1 Definition**

Box Type: 'pres'  
Container: TrackGroupBox  
Mandatory: No  
Quantity: Zero or one

The presence of a TrackGroupTypeBox with track\_group\_type equal to 'pres', which is also referred to as ...

### Sample Description Boxes

BitRateBox is defined within the Sample description box. Below is the proposed clause 8.5.2 that was restructured and split into multiple subclauses.

**8.5.2 Sample description box**

**8.5.2.1 Definition**

Box Types: 'stsd'  
Container: SampleTableBox  
Mandatory: Yes  
Quantity: Exactly one

...

The definition of sample entries specifies boxes in a particular order, and this is usually also followed in derived specifications. For maximum compatibility, writers should construct files respecting the order both within specifications and as implied by the inheritance, whereas readers should be prepared to accept any box order.

All SampleEntry boxes may contain “extra boxes” not explicitly defined in the box syntax of this or derived specifications. When present, such boxes shall follow all defined fields and should follow any defined contained boxes. Decoders shall presume a sample entry box could contain extra boxes and shall continue parsing as though they are present until the containing box length is exhausted.

~~An optional BitRateBox may be present in any SampleEntry to signal the bit rate information of a stream. This can be used for buffer configuration.~~

All string fields shall be of type utf8string and null-terminated, even if unused. “Optional” means there is at least one null byte.

Entries that identify the format by MIME type, such as ...

**8.5.2.2 Syntax**

aligned(8) abstract class SampleEntry (unsigned int(32) format)  
 extends Box(format) {  
 const unsigned int(8)[6] reserved = 0;  
 unsigned int(16) data\_reference\_index;  
}

~~class BitRateBox extends Box('btrt'){  
 unsigned int(32) bufferSizeDB;  
 unsigned int(32) maxBitrate;  
 unsigned int(32) avgBitrate;  
}~~

aligned(8) class SampleDescriptionBox ()  
 extends FullBox('stsd', version, 0){  
 int i;  
 unsigned int(32) entry\_count;  
 for (i = 1 ; i <= entry\_count ; i++){  
 SampleEntry sample\_entry;~~// an instance of a class derived from SampleEntry~~  
 }  
}

**8.5.2.3 Semantics**

version is set to zero. A version number of 1 shall be treated as a version of 0.

entry\_count is an integer that gives the number of entries in the following table.

sample\_entry is an instance of a class derived from SampleEntry that represents the appropriate sample entry.

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.

~~bufferSizeDB gives the size of the decoding buffer for the media stream in bytes.~~

~~maxBitrate gives the maximum rate in bits/second over any window of one second; this is a measured value for stored content, or a value that a stream is configured not to exceed; the stream shall not exceed this bitrate.~~

~~avgBitrate gives the average rate in bits/second of the stream; this is a measured value (cumulative over the entire presentation) for stored content, or the configured target average bitrate for a stream.~~

**8.5.2.4 Generic sample entry boxes**

**8.5.2.4.1 Bitrate Box**

**Definition**

Box Types: 'btrt'  
Container: SampleEntry  
Mandatory: No  
Quantity: Zero or one

An optional BitRateBox may be present in any SampleEntry to signal the bit rate information of a stream. This can be used for buffer configuration.

**Syntax**

class BitRateBox extends Box('btrt'){  
 unsigned int(32) bufferSizeDB;  
 unsigned int(32) maxBitrate;  
 unsigned int(32) avgBitrate;  
}

**Semantics**

bufferSizeDB gives the size of the decoding buffer for the media stream in bytes.

maxBitrate gives the maximum rate in bits/second over any window of one second; this is a measured value for stored content, or a value that a stream is configured not to exceed; the stream shall not exceed this bitrate.

avgBitrate gives the average rate in bits/second of the stream; this is a measured value (cumulative over the entire presentation) for stored content, or the configured target average bitrate for a stream.

### Video media

**12.1.3 Sample entry**

**12.1.3.1 Definition**

Box Types: codingname  
Container: SampleTableBox  
Mandatory: Yes, for video tracks  
Quantity: One or More

Video tracks use VisualSampleEntry.

In video tracks, the frame\_count field shall be 1 unless the specification for the media format explicitly documents this template field and permits larger values. That specification must document both how the individual frames of video are found (their size information) and their timing established. That timing might be as simple as dividing the sample duration by the frame count to establish the frame duration.

The width and height in the video sample entry document the pixel counts that the codec will deliver; this enables the allocation of buffers. Since these are counts they do not take into account pixel aspect ratio.

**12.1.3.2 Syntax**

class VisualSampleEntry(codingname) extends SampleEntry (codingname){  
 unsigned int(16) pre\_defined = 0;  
 const unsigned int(16) reserved = 0;  
 unsigned int(32)[3] pre\_defined = 0;  
 unsigned int(16) width;  
 unsigned int(16) height;  
 template unsigned int(32) horizresolution = 0x00480000; // 72 dpi  
 template unsigned int(32) vertresolution = 0x00480000; // 72 dpi  
 const unsigned int(32) reserved = 0;  
 template unsigned int(16) frame\_count = 1;  
 uint(8)[32] compressorname;  
 template unsigned int(16) depth = 0x0018;  
 int(16) pre\_defined = -1;  
 // other boxes from derived specifications  
 Box other\_boxes[];  
 ~~CleanApertureBox clap; // optional  
 PixelAspectRatioBox pasp; // optional~~  
}

...

**12.1.4 Pixel aspect ratio and clean aperture**

**12.1.4.1 Definition**

Box Types: pasp  
Container: VisualSampleEntry  
Mandatory: No  
Quantity: Zero or one

Box Types: clap  
Container: VisualSampleEntry  
Mandatory: No  
Quantity: Zero or one

...

**12.1.5 Colour information**

**12.1.5.1 Definition**

Box Types: colr  
Container: VisualSampleEntry  
Mandatory: No  
Quantity: Zero or more

...

12.1.5.2 Syntax

ICC\_profile should be defined as bit(8) icc\_profile[];

**12.1.6 Content light level**

**12.1.6.1 Definition**

Box Types: clli  
Container: VisualSampleEntry  
Mandatory: No  
Quantity: Zero or one

...

**12.1.6.3 Semantics**

max\_content\_light\_level: **TBD**.

max\_pic\_average\_light\_leve: **TBD**

**TBD** same as above for mdcv, cclv and amve

# Item\_name clarifications

*Clarify the format and permissible values for item\_name. Remove the following from clause 8.11.6.1:*

The item\_name shall be a valid URL (e.g. a simple name, or path name) and shall not be an absolute URL.

*Replace the following in clause 8.11.6.3.*

item\_name is the symbolic name of the item (source file for file delivery transmissions).

*with*

item\_name shall be a string which when spaces and special characters (as defined in RFC 3986) are percent-encoded (as define in RFC 3986) represent a valid relative URL (e.g. a simple name, or path name). When item\_name is used in the context of fragment identifiers as specified in Annex C, it shall be percent-encoded.

NOTE: The empty string is a valid value for item\_name but it cannot be used as value for the item\_name fragment identifier.

[Ed. note: We could add the examples to the spec and a NOTE indicating how a resource (e.g. HTML) and its sub-resources (e.g. PNG) can be packaged into multiple items while preserving the links.]

*To ensure consistency, update Annex C to clarify the handling of item\_name in fragment identifiers:*

c) item\_name=<item\_name>, identifying the item of the MetaBox at the file level that has the given name (as provided in the ItemInfoBox). The item\_name shall be percent-encoded in compliance with RFC 3986 and shall not be an empty string.

e) /item\_name=<item\_name>, identifying the item of the MetaBox at the movie level that has the given name (as provided in the ItemInfoBox). The item\_name shall be percent-encoded in compliance with RFC 3986 and shall not be an empty string.

g) track\_ID=<track\_ID>/item\_name=<item\_name>, identifying the item that has the given name (as provided in the ItemInfoBox) in the MetaBox located in the track with the given track\_ID. The item\_name shall be percent-encoded in compliance with RFC 3986 and shall not be an empty string.

Examples of item\_name usage in fragment identifiers:

* item\_name=simple-name
* item\_name=path/to/resource
* item\_name="" -> **not allowed**
* item\_name=HEVC%20Image

# On codecs string

*Add a new clause K.2.3:*

## K.2.3 Rendering Capabilities as an Extension of the 'codecs' parameter

To signal rendering requirements within the existing codecs parameter, this clause defines an optional syntax extension using the reserved four-character code 'also'. This extension allows for rendering capabilities to be embedded directly in the codecs string, supporting codec-agnostic signalling for media that requires specific rendering configurations.

### K.2.3.1 Structure of the 'also' 4CC extension

Rendering capabilities are defined through key-value pairs, concatenated with plus signs (+), following the 'also' 4CC prefix. The supported parameters include:

[Ed. note: the exact format and what separators to use is a subject to change]

* Image Type: Specifies the image type (e.g., regular video/texture, alpha, or depth), potentially based on ISO/IEC 23091-2 values.
* Color Space: Indicates the color space (e.g., BT709), aligned with ISO/IEC 23091-2.
* Subsampling: Specifies the subsampling method (e.g., center), also based on ISO/IEC 23091-2.

[Ed. note: other parameters may include film grain signalling or the group could even think about signalling encryption related information]

Example of 'codecs' string with rendering capabilities:

codecs=”also.type=alpha+color=BT709+subsample=center.hev1….”

NOTE: Applications can choose this extension for a consolidated signaling method that combines rendering and codec information within a single codecs parameter. Alternatively, applications may use a separate rendering MIME type parameter as defined in K.7.

*Add a new clause K.7:*

## K.7 Use of the 'rendering' parameter

This clause introduces a new MIME type parameter, 'rendering', to provide a codec-agnostic signalling mechanism for rendering capabilities. This parameter is intended for applications that prefer to separate rendering requirements from the 'codecs' parameter, offering a distinct location for rendering information without modifying the 'codecs' identifier.

Applications may choose between using the 'rendering' MIME type parameter defined in this clause or the 'codecs' parameter extension specified in K.2.3. The choice is application-specific, allowing flexibility based on playback requirements and compatibility considerations.

The 'rendering' parameter uses the same key-value pairs and syntax outlined in K.2.3.1, ensuring consistent and interoperable signalling across implementations.

Example of Separate 'rendering' parameter:

rendering="type=alpha+color=BT709+subsample=center"

[Ed. note: This is a preliminary example of the signalling and is subject to change after a larger discussion on the format, separators, sub-parameters, etc.]

# ExternalTracks

*Add new section in track structure (section 8.3)*

8.3.7 External Tracks

8.3.7.1 External Track Box

8.3.7.1.1 Definition

BoxType: 'extk'   
Container: ‘moov’   
Mandatory: No  
Yes Quantity: zero or more

An ExternalTrackBox can be used to include a track from another ISO Base Media file, as defined by its TrackBox and other track-related structures. The track being referred to is called an external track. The file containing the ExternalTrackBox is hereafter called the referring file, and the file containing the external track is called the referred file. Referred files shall be ISOBMFF compliant files.

External tracks may be fragmented or not, independently of whether the referring file is fragmented or not. Derived specifications may further restrict possible combinations.

The timeline of an external track may be modified by an edit list in the referring file.

The UserDataBox and MetaBox of an external track can be overridden or augmented. UserDataBox present at movie level or MetaBox present at file or movie level in the referred files shall be ignored, and only UserDataBox present at movie level or MetaBox present at file or movie level, if any, of the referring file shall apply.

Track references and track groups of the referred files are ignored and only track references and groups (track groups or entity groups) defined in the referring file are valid.

The track\_ID of the TrackHeaderBox present in ExternalTrackBox gives the identifier of the track in the referring file and can be used to describe track references, track groups and other track relationships relying on track identifiers within the referring file. This allows defining track relations or track groups independently from the identifiers used in the referred file(s).

Additionally, the TrackHeaderBox provides the presentation information of the external track within the presentation of the referring file, such as track width/height, matrix, volumes and track flags.

The following restrictions are set on the TrackHeaderBox in the ExternalTrackBox of an external track:

* If the duration field is undefined (all 1s) and there is no edit list for this track, then the duration of the track is the duration of the referenced track.

The sample description, offsets, sizes… for an external track are the ones declared in the referred file indicated by location in the ExternalTrackLocationBox.

8.3.7.1.2 Syntax

aligned(8) class ExternalTrackBox extends Box('extk'){

ExternalTrackLocationBox extl; //shall be first

TrackHeaderBox tkhd; //shall be second

Box other[]; // any valid children of ‘trak’ except ‘mdia’ and ‘tkhd’, or ExtendedLanguageBox

}

8.3.7.1.3 Semantics

extl indicates the location of the external track. It shall be present and shall be the first box occurring in ExternalTrackBox

tkhd indicates the track header of the external track within the context of the referring file. It shall be present and shall be the second box occurring in ExternalTrackBox.

other indicate any possible box allowed as child of TrackBox except MediaBox and TrackHeaderBox, or ExtendedLanguageBox. Further restrictions on the type of children allowed in ExternalTrackBox are given by the ExternalTrackLocationBox.

8.3.7.2 External Track Location Box

8.3.7.2.1 Definition

BoxType: 'extl'  
Container: ‘extk’   
Mandatory: Yes   
Quantity: one

The ExternalTrackLocationBox allows identifying an external track by its track\_ID in a referred file.

The following values are defined for the flags field of the ExternalTrackLocationBox:

* EXTERNAL\_TRACK\_EDTS\_SKIP (flag mask is 0x000001): shall be set if any edit list present in the external track shall be ignored. If an edit list is present in the container for the edit lists in the ExternalTrackBox for this track, flag shall be set and any edit list present in the external track shall be ignored. Otherwise, (not set) there shall be no EditBox in the ExternalTrackBox, and the edit lists of the external track apply.
* EXTERNAL\_TRACK\_URN (flag mask is 0x000002): if this flag is set, it indicates that the location field is a URN string, otherwise (not set) the location string is a URL,
* EXTERNAL\_TRACK\_UDTA\_IGNORE (flag mask is 0x000004): if this flag is set, this indicates that any UserDataBox defined in the ‘trak’ box of the external track shall be ignored. Otherwise (not set), UserDataBox present in the referring track completes UserDataBox information of the external track. The resulting user data consists in the union of the user data declared in the different UserDataBoxes.
* EXTERNAL\_TRACK\_META\_IGNORE (flag mask is 0x000008): if this flag is set, this indicates that any MetaBox defined in the ‘trak’ box of the external track shall be ignored. Otherwise (not set), MetaBox present in the referring track completes MetaBox information of the external track. The resulting meta data consists in the union of the meta data declared in the different MetaBoxes.

If the indicated location is a URL, it can be an absolute or a relative URL, and the located resource shall be a compliant ISOBMF file. Relative URLs are relative to the file that contains this location.

When EXTERNAL\_TRACK\_EDTS\_SKIP is set and no edit list is present in the ExternalTrackBox, this implies that any edit present in the referred track is ignored and no edit is applied to the track.

If edits from the referred track are used, file readers may need to remap the edit list durations from the timescale of the referred movie to the timescale of the referring movie, if these timescales differ.

8.3.7.2.2 Syntax

class ExternalTrackLocationBox extends FullBox ('extl', version=0, flags)  
{  
 unsigned int(32) referenced\_track\_ID;  
 unsigned int(32) referenced\_handler\_type;   
 unsigned int(32) media\_timescale;   
 utf8string location;   
}

8.3.7.2.3 Semantics

referenced\_track\_ID indicates the identifier (track\_ID) of the external track in the referred file. If value is 0, this indicates that the referenced track is the first TrackBox present in the MovieBox of the referred file. The external track shall be declared through a TrackBox, i.e. recursively referencing external track is forbidden. The external track can use external data references or not; this can be constrained by derived specifications.

referenced\_handler\_type indicates the handler (media) type of the track, and shall be equal to the handler type of the external track in the referred file.

media\_timescale indicates the timescale used to express edit list contained in this external track. Value may be 0 when no edit list is declared in the ExternalTrackBox, or a different value expressing a preferred timescale in case of future insertion of an edit list. Otherwise (an edit list is declared in the ExternalTrackBox), value shall not be 0.

NOTE: this value may be different from the timescale in the MediaHeader in the external track.

location indicates the location of the referred file as a URN or URN, depending on the flags EXTERNAL\_TRACK\_URN.

8.3.7.3 External Track Processing Model

A file reader processes an external track as follows:

* Identify whether the referring file can be processed (brands, track handler types): this follows the same process as for files with no external tracks
* Identify whether it should take the track into consideration: this follows the same rules as for regular tracks, e.g. looking at user preferences, groups, etc …
* If an external track is selected for processing, the referred file is loaded. The external track is marked as invalid if any of the following is true:
  + the location described is invalid
  + the file and/or track cannot be processed by the reader due to brand requirements in ftyp or ttyp
  + the TrackBox corresponding to the external track cannot be found in the referred file,
  + the external track handler type does not match the handler type in ExternalTrackLocationBox
  + the ExternalTrackBox contains a TrackTypeBox with unsupported brands

If an external track is invalid, file readers may reject the file or present only a subset of the external tracks that are valid, as they would usually do for files with no external tracks,

* Otherwise (external track is valid), the processing of the external track is equivalent to processing the track using a second file reader, but using track groups and references defined in the referring file; this implies that global movie structure of the referred file, such as trex, pssh… may be required to process the external file.

‘Meta’ at file or moov level and ‘udta’ at moov level in the refered file(s) shall be ignored.

[Ed. Note: Could the location be designed in such a way that changing its value would not require size changes of the moov ?]

# Change to ‘kind’ box containers

*Replace the following table in clause 8.10.4.1:*

|  |  |
| --- | --- |
| Box Type: | 'kind' |
| Container: | AudioElementBox, PreselectionGroupBox, or  UserDataBox of the corresponding TrackBox |
| Mandatory: | No |
| Quantity: | Zero or more |

*with:*

|  |  |
| --- | --- |
| Box Type: | 'kind' |
| Container: | AudioElementBox, PreselectionGroupBox, or  UserDataBox |
| Mandatory: | No |
| Quantity: | Zero or more |

# Sidx change

*Change the syntax from:*

aligned(8) class SegmentIndexBox extends FullBox('sidx', version, 0)

{

unsigned int(32) reference\_ID;

unsigned int(32) timescale;

if (version==0) {

unsigned int(32) earliest\_presentation\_time;

unsigned int(32) first\_offset;

}

else {

unsigned int(64) earliest\_presentation\_time;

unsigned int(64) first\_offset;

}

unsigned int(16) reserved = 0;

unsigned int(16) reference\_count;

for (i=1; i <= reference\_count; i++)

{

bit (1) reference\_type;

unsigned int(31) referenced\_size;

unsigned int(32) subsegment\_duration;

bit(1) starts\_with\_SAP;

unsigned int(3) SAP\_type;

unsigned int(28) SAP\_delta\_time;

}

}

*to:*

aligned(8) class SegmentIndexBox extends FullBox('sidx', version, 0) {

unsigned int(32) reference\_ID;

unsigned int(32) timescale;

if (version == 0) {

unsigned int(32) earliest\_presentation\_time;

unsigned int(32) first\_offset;

}

else if (version == 1) {

unsigned int(64) earliest\_presentation\_time;

unsigned int(64) first\_offset;

}

else if (version == 2) {

unsigned int(64) earliest\_presentation\_time;

unsigned int(64) offset\_anchor;

unsigned int(64) first\_offset;

}

unsigned int(16) reserved = 0;

unsigned int(16) reference\_count;

for(i=1; i <= reference\_count; i++) {

bit (1) reference\_type;

unsigned int(31) referenced\_size;

unsigned int(32) subsegment\_duration;

bit(1) starts\_with\_SAP;

unsigned int(3) SAP\_type;

unsigned int(28) SAP\_delta\_time;

}

}

*Change the paragraph:*

In the file containing the SegmentIndexBox, the anchor point for a SegmentIndexBox is the first byte after that box. If there are two files, the anchor point in the media file is the beginning of the top-level segment (i.e. the beginning of the segment file if each segment is stored in a separate file). The material in the file containing media (which may also be the file that contains the SegmentIndexBoxes) starts at the indicated offset from the anchor point. If there are two files, the material in the index file starts at the anchor point, i.e. immediately following the SegmentIndexBox.

*to:*

In the file containing the SegmentIndexBox, the anchor point for a SegmentIndexBox is the first byte after that box if version 2 SegmentIndexBox not used, otherwise the anchor point is set by offset\_anchor. If there are two files and version 2 of SegmentIndexBox is not used, the anchor point in the media file is the beginning of the top-level segment (i.e. the beginning of the segment file if each segment is stored in a separate file). Otherwise, the anchor point is set by offset\_anchor. The material in the file containing media (which may also be the file that contains the SegmentIndexBoxes) starts at the indicated offset from the anchor point. If there are two files, the material in the index file starts at the anchor point.

*add semantics for offset\_anchor:*

offset\_anchor is a distance in bytes, in the file containing the media, from the start of the file. first\_offset determines the distance to the indexed material from offset\_anchor.

*Change the bullet point from:*

* Segment index boxes shall be placed before subsegment material they document, that is, before any MovieFragmentBox of the documented material of the subsegment;

*to:*

* Segment index boxes shall be placed before subsegment material they document, that is, before any MovieFragmentBox of the documented material of the subsegment, unless sidx version 2 is used;

# Unif clarifications

From BoG recommendation:

Unif clarifications (Cyril)

https://git.mpeg.expert/MPEG/Systems/FileFormat/isobmff/-/issues/301

# FileTypeBox updates

*Change part of the definition in clause 5.2.1 from:*

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.

The minor version is informative only. It does not appear for compatible-brands, and is not used to determine the conformance of a file to a standard. It may allow more precise identification of the major specification, for inspection, debugging, or improved decoding.

*to:*

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 major\_brand and the compatible\_brands); the major\_brand may be repeated in the compatible\_brands list. If only a single brand needs to be signaled, the compatible brands list may be empty. 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.

The minor version is informative only. It does not appear for compatible-brands, and is not used to determine the conformance of a file to a standard. It may allow more precise identification of the major specification, for inspection, debugging, or improved decoding. The interpretation of the minor version is major-brand specific. The semantics of the 32 bits of the minor\_version field may be re-defined by the specification defining the major brand value, for example using these 32 bits as flags.

# Track fragment decode time box clarifications

*Replace the following text from subclause 8.8.12.3*

baseMediaDecodeTime is an integer equal to the sum of the decode durations of all earlier samples in the media, expressed in the media's timescale. It does not include the samples added in the enclosing track fragment.

*with:*

baseMediaDecodeTime is the absolute decoding timestamp, measured on the decoding timeline, of the first sample in decoding order in the track fragment, expressed in the media's timescale. The value of baseMediaDecodeTime shall be greater than or equal to the sum of the sample durations of all the samples of this track that precede this track fragment in decoding order.

# Clarify behavior for end of box data

*Add the following definition to clause 3:*

**3.1.XX  
content box**

box that is not a container box

*Replace the following text from subclause 4.2.2:*

Boxes specified in this document may be extended but such extensions are reserved for future use by ISO/IEC. Syntax may be added at the end of a box derived from FullBox and an already specified version value may be kept, if it is not essential to parse such added syntax. When a parser has not reached the end of a box derived from FullBox as defined by the values of the size or largesize field (as appropriate) but does not recognize the remaining syntax elements, it shall ignore and skip the remaining of the box.

*with:*

Content boxes specified in this document may be extended but such extensions are reserved for future use by ISO/IEC. When it is not essential to parse a syntax extension of a content box, the syntax extension may be added at the end of the content box. When a content box is derived from FullBox and it is not essential to parse a syntax extension that is added at the end of the content box, an already specified version value of the content box may be kept. When a parser has not reached the end of a content box as defined by the values of the size or largesize field (as appropriate) but does not recognize the remaining syntax elements, it shall ignore and skip the remaining of the content box.

# Other improvements

*Replace the following text from subclause 12.1.5.1:*

If colour information is supplied in both this box, and also in the video bitstream, this box takes precedence, and over-rides the information in the bitstream.

*with:*

The colour information supplied in both this box and in the video bitstream should match. If it is not the case, this box takes precedence, and over-rides the information in the bitstream.

*replace the following text in Table 2:*

data reference box, declares source(s) of metadata items

*with:*

data reference box, declares source(s) of items

# Changes from Defect Report

*Replace Annex E.2 with the following:*

**E.2 The 'isom' brand**

**E.2.1 Requirements on files**

The type 'isom' (ISO base media file) is defined in this clause, as identifying files that conform to the first version of ISO base media file format.

The boxes listed in Table E.1 are required in a file conforming to the this brand. The Version column in Table E.1 specifies the version values allowed by this brand. Other version values shall not be present. A '-' in the Version column indicates that the box is derived from Box and does not contain a version field. The Additional requirements column in Table E.1 specifies additional requirements on files conforming to this brand.

**Table E.1: Required boxes in a file conforming to the 'isom' brand**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | **Version** | **Additional requirements** |
| ftyp |  |  |  |  |  | - |  |
| moov |  |  |  |  |  | - |  |
|  | mvhd |  |  |  |  | 0, 1 |  |
|  | trak |  |  |  |  | - | There shall be at least one media track present that is constrained as specified in this table. |
|  |  | tkhd |  |  |  | 0, 1 |  |
|  |  | mdia |  |  |  | - |  |
|  |  |  | mdhd |  |  | 0, 1 |  |
|  |  |  | hdlr |  |  | 0 |  |
|  |  |  | minf |  |  | - | Exactly one media header box shall be present within the 'minf' box. |
|  |  |  |  | dinf |  | - |  |
|  |  |  |  |  | dref | 0 | Each DataEntryBox within the DataReferenceBox shall be either a DataEntryUrnBox or a DataEntryUrlBox. |
|  |  |  |  | stbl |  | - |  |
|  |  |  |  |  | stts | 0 |  |
|  |  |  |  |  | stsd | 0 |  |
|  |  |  |  |  | stsz | 0 | Either SampleSizeBox ('stsz') or CompactSampleSizeBox ('stz2') shall be present within the 'stbl' box. |
|  |  |  |  |  | stz2 | 0 | Either SampleSizeBox ('stsz') or CompactSampleSizeBox ('stz2') shall be present within the 'stbl' box. |
|  |  |  |  |  | stsc | 0 |  |
|  |  |  |  |  | stco | 0 | Either ChunkOffsetBox ('stco') or ChunkLargeOffsetBox ('co64') shall be present within the 'stbl' box. |
|  |  |  |  |  | co64 | 0 | Either ChunkOffsetBox ('stco') or ChunkLargeOffsetBox ('co64') shall be present within the 'stbl' box. |

~~NOTE 1 The default-base-is-moof flag (~~**~~Error! Reference source not found.~~**~~) cannot be set where a file is marked with this brand.~~

When a file is marked with this brand and includes a TrackFragmentHeaderBox, the default-base-is-moof flag shall not be set in tf\_flags of the TrackFragmentHeaderBox.

NOTE~~2~~ Some requirements of the TrackHeaderBox do not apply to this brand; see subclause **Error! Reference source not found.**.

Under this brand and its derivatives the media\_rate in the EditListBox is restricted such that the fraction shall have the value 0 and the integer shall have the value 0 or 1.

**E.2.2 Requirements on readers**

Support for the structural boxes according to Table E.2 is required. The Version column in Table E.2 specifies the version values that shall be supported by the readers of this brand. A '-' in the Version column indicates that the box is derived from Box and does not contain a version field.

**Table E.2: Boxes required to be supported in readers of ~~under~~ the 'isom' brand**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | **Version** | **Box description** |
| ftyp |  |  |  |  |  |  | - | *file type and compatibility* |
| moov |  |  |  |  |  |  | - | *container for all the meta-data* |
|  | mvhd |  |  |  |  |  | 0, 1 | *movie header, overall declarations* |
|  | trak |  |  |  |  |  | - | *container for an individual track or stream* |
|  |  | tkhd |  |  |  |  | 0, 1 | *track header, overall information about the track* |
|  |  | tref |  |  |  |  | - | *track reference container* |
|  |  | edts |  |  |  |  | - | *edit list container* |
|  |  |  | elst |  |  |  | 0, 1 | *an edit list* |
|  |  | mdia |  |  |  |  | - | *container for the media information in a track* |
|  |  |  | mdhd |  |  |  | 0, 1 | *media header, overall information about the media* |
|  |  |  | hdlr |  |  |  | 0 | *handler, at this level, the media (handler) type* |
|  |  |  | minf |  |  |  | - | *media information container* |
|  |  |  |  | vmhd |  |  | 0 | *video media header, overall information (video track only)* |
|  |  |  |  | smhd |  |  | 0 | *sound media header, overall information (sound track only)* |
|  |  |  |  | hmhd |  |  | 0 | *hint media header, overall information (hint track only)* |
|  |  |  |  | <mpeg> |  |  |  | *mpeg stream headers* |
|  |  |  |  | dinf |  |  | - | *data information box, container* |
|  |  |  |  |  | dref |  | 0 | *data reference box, declares source(s) of media in track* |
|  |  |  |  |  |  | url | 0 | *URL data entry box* |
|  |  |  |  |  |  | urn | 0 | *URN data entry box* |
|  |  |  |  | stbl |  |  | - | *sample table box, container for the time/space map* |
|  |  |  |  |  | stts |  | 0 | *(decoding) time-to-sample* |
|  |  |  |  |  | ctts |  | 0 | *composition time-to-sample table* |
|  |  |  |  |  | stss |  | 0 | *sync (key, I-frame) sample map* |
|  |  |  |  |  | stsd |  | 0 | *sample description box (codec types, initialization etc.)* |
|  |  |  |  |  | stsz |  | 0 | *sample sizes (framing)* |
|  |  |  |  |  | stz2 |  | 0 | *compact sample sizes (framing)* |
|  |  |  |  |  | stsc |  | 0 | *sample-to-chunk, partial data-offset information* |
|  |  |  |  |  | stco |  | 0 | *chunk offset, partial data-offset information* |
|  |  |  |  |  | co64 |  | 0 | *64-bit chunk offset* |
|  |  |  |  |  | stsh |  | 0 | *shadow sync* |
|  |  |  |  |  | padb |  | 0 | *sample padding bits* |
|  |  |  |  |  | stdp |  | 0 | *degradation priority* |
|  |  | udta |  |  |  |  | - | *user-data, copyright etc.* |
|  | mvex |  |  |  |  |  | - | *movie extends box* |
|  |  | mehd |  |  |  |  | 0, 1 | *movie extends header box* |
|  |  | trex |  |  |  |  | 0 | *track extends defaults* |
|  | udta |  |  |  |  |  | - | *user-data, copyright etc.* |
| mdat |  |  |  |  |  |  | - | *Media data container* |
| free |  |  |  |  |  |  | - | *free space* |
| skip |  |  |  |  |  |  | - | *free space* |
| ~~udta~~ |  |  |  |  |  |  |  | *~~user-data, copyright etc.~~* |
| moof |  |  |  |  |  |  | - | *movie fragment* |
|  | mfhd |  |  |  |  |  | 0 | *movie fragment header* |
|  | traf |  |  |  |  |  | - | *track fragment* |
|  |  | tfhd |  |  |  |  | 0 | *track fragment header* |
|  |  | trun |  |  |  |  | 0 | *track fragment run* |
| mfra |  |  |  |  |  |  | - | *movie fragment random access* |
|  | tfra |  |  |  |  |  | 0, 1 | *track fragment random access* |
|  | mfro |  |  |  |  |  | 0 | *movie fragment random access offset* |

Hint tracks shall be recognized, and in hint tracks, RTP protocol hint tracks.

The following syntax elements within default\_sample\_flags of TrackExtendsBox and TrackFragmentHeaderBox and within sample\_flags and first\_sample\_flags of TrackRunBox shall be supported: sample\_padding\_value, sample\_is\_non\_sync\_sample, sample\_degradation\_priority.

The following flags of TrackFragmentHeaderBox shall be supported:

base-data-offset-present

sample-description-index-present

default-sample-duration-present

default-sample-size-present

default-sample-flags-present

duration-is-empty

The following flags of TrackRunBox shall be supported:

data-offset-present

first-sample-flags-present

sample-duration-present

sample-size-present

sample-flags-present

sample-composition-time-offsets-present

~~Support for only version 0 of the CompositionOffsetBox is required; version 1 support is not required.~~

~~Support for only version 0 of the TrackRunBox is required; version 1 support is not required.~~

*replace the following paragraph from Annex A.11:*

— For video, it is suggested to form track fragments so that the first sample of a track fragment can be marked as a sync sample or a SAP sample. In the case of gradual decoder refresh, a SAP sample of type 4 and the corresponding random access recovery point are stored in the same movie fragment. For audio, samples having the closest presentation time for every video random accessible sample are stored as the first sample of each TrackFragmentBox. Hence, the first samples of each media in the MovieFragmentBox have the approximately equal presentation times.

*with:*

— For video, it is suggested to form track fragments so that the first sample of a track fragment can be marked as a sync sample or a SAP sample. In the case of gradual decoder refresh, a SAP sample of type 4 and the corresponding random access recovery point are stored in the same movie fragment. For audio, samples having the closest presentation time for every video random access point sample are stored as the first sample of each TrackFragmentBox. Hence, the first samples of each media in the MovieFragmentBox have the approximately equal presentation times.

*Replace the subclause 8.8.11.2 with the following:*

aligned(8) class MovieFragmentRandomAccessOffsetBox extends FullBox('mfro', 0, 0)

{

unsigned int(32) parent\_size;

}

*Replace the subclause 8.4.5.2.2 with the following:*

aligned(8) class NullMediaHeaderBox extends FullBox('nmhd', 0, 0)

{}

*Replace the subclause 6.2 with the following:*

An overall view of the normal encapsulation structure is provided in the following informative Table 2. In the event of a conflict between this table and the prose, the prose prevails. The order of boxes within its container is not necessarily indicated in Table 2.

The table shows those boxes that may occur at the top-level in the left-most column; indentation is used to show possible containment. Thus, for example, a TrackHeaderBox ('tkhd') is found in a TrackBox ('trak'), which is found in a MovieBox ('moov').

Boxes using an extended type may be placed in a wide variety of containers, not just the top level.

Table 2 — Box types, structure and cross-reference

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ftyp |  |  |  |  |  |  | 5.2 | file type and compatibility |
| otyp |  |  |  |  |  |  | 8.16.5 | original file-type |
| pdin |  |  |  |  |  |  | 8.1.3 | progressive download information |
| moov |  |  |  |  |  |  | 8.2.1 | container for all the structure-data |
|  | mvhd |  |  |  |  |  | 8.2.2 | movie header, overall declarations |
|  | meta |  |  |  |  |  | 8.11.1 | metadata |
|  | trak |  |  |  |  |  | 8.3.1 | container for an individual track or stream |
|  |  | ttyp |  |  |  |  | 8.3.5 | track type |
|  |  | tkhd |  |  |  |  | 8.3.2 | track header, overall information about the track |
|  |  | tref |  |  |  |  | 8.3.3 | track reference container |
|  |  | trgr |  |  |  |  | 8.3.4 | track grouping indication |
|  |  |  | msrc |  |  |  | 8.3.4.4.1 | multi-source presentation track group type box |
|  |  |  | ster |  |  |  | 8.3.4.4.2 | stereoscopic pair track group type box |
|  |  | edts |  |  |  |  | 8.6.5 | edit list container |
|  |  |  | elst |  |  |  | 8.6.6 | an edit list |
|  |  | meta |  |  |  |  | 8.11.1 | metadata |
|  |  | mdia |  |  |  |  | 8.4.1 | container for the media information in a track |
|  |  |  | mdhd |  |  |  | 8.4.2 | media header, overall information about the media |
|  |  |  | hdlr |  |  |  | 8.4.3 | handler, declares the media (handler) type |
|  |  |  | elng |  |  |  | 8.4.6 | extended language tag |
|  |  |  | minf |  |  |  | 8.4.4 | media information container |
|  |  |  |  | vmhd |  |  | 12.1.2 | video media header, overall information (video track only) |
|  |  |  |  | smhd |  |  | 12.2.2 | sound media header, overall information (sound track only) |
|  |  |  |  | hmhd |  |  | 12.4.3 | hint media header, overall information (hint track only) |
|  |  |  |  | sthd |  |  | 12.6.2 | subtitle media header, overall information (subtitle track only) |
|  |  |  |  | nmhd |  |  | 8.4.5.2 | Null media header, overall information (some tracks only) |
|  |  |  |  | dinf |  |  | 8.7.1 | data information box, container |
|  |  |  |  |  | dref |  | 8.7.2 | data reference box, declares source(s) of media data in track |
|  |  |  |  | stbl |  |  | 8.5.1 | sample table box, container for the time/space map |
|  |  |  |  |  | stsd |  | 8.5.2 | sample description box (codec types, initialization etc.) |
|  |  |  |  |  | stts |  | 8.6.1.2 | (decoding) time-to-sample |
|  |  |  |  |  | ctts |  | 8.6.1.3 | (composition) time to sample |
|  |  |  |  |  | cslg |  | 8.6.1.4 | composition to decode timeline mapping |
|  |  |  |  |  | stsc |  | 8.7.4 | sample-to-chunk, partial data-offset information |
|  |  |  |  |  | stsz |  | 8.7.3.2 | sample sizes (framing) |
|  |  |  |  |  | stz2 |  | 8.7.3.3 | compact sample sizes (framing) |
|  |  |  |  |  | stco |  | 8.7.5 | chunk offset, partial data-offset information |
|  |  |  |  |  | co64 |  | 8.7.5 | 64-bit chunk offset |
|  |  |  |  |  | stss |  | 8.6.2 | sync sample table |
|  |  |  |  |  | stsh |  | 8.6.3 | shadow sync sample table |
|  |  |  |  |  | padb |  | 8.7.6 | sample padding bits |
|  |  |  |  |  | stdp |  | 8.5.3 | sample degradation priority |
|  |  |  |  |  | sdtp |  | 8.6.4 | independent and disposable samples |
|  |  |  |  |  | sbgp |  | 8.9.2 | sample-to-group |
|  |  |  |  |  | sgpd |  | 8.9.3 | sample group description |
|  |  |  |  |  | subs |  | 8.7.7 | sub-sample information |
|  |  |  |  |  | saiz |  | 8.7.8 | sample auxiliary information sizes |
|  |  |  |  |  | saio |  | 8.7.9 | sample auxiliary information offsets |
|  |  | udta |  |  |  |  | 8.10.1 | user-data |
|  |  |  | cprt |  |  |  | 8.10.2 | copyright etc. |
|  |  |  | tsel |  |  |  | 8.10.3 | track selection box |
|  |  |  | kind |  |  |  | 8.10.4 | track kind box |
|  |  |  | strk |  |  |  | 8.13.3 | sub track box |
|  |  |  |  | stri |  |  | 8.13.4 | sub track information box |
|  |  |  |  | strd |  |  | 8.13.5 | sub track definition box |
|  |  |  | ludt |  |  |  | 12.2.7 | audio stream loudness |
|  | mvex |  |  |  |  |  | 8.8.1 | movie extends box |
|  |  | mehd |  |  |  |  | 8.8.2 | movie extends header box |
|  |  | trex |  |  |  |  | 8.8.3 | track extends defaults |
|  |  | leva |  |  |  |  | 8.8.13 | level assignment |
|  | udta |  |  |  |  |  | 8.10.1 | user-data |
|  |  | cprt |  |  |  |  | 8.10.2 | copyright etc. |
| moof |  |  |  |  |  |  | 8.8.4 | movie fragment |
|  | mfhd |  |  |  |  |  | 8.8.5 | movie fragment header |
|  | meta |  |  |  |  |  | 8.11.1 | metadata |
|  | traf |  |  |  |  |  | 8.8.6 | track fragment |
|  |  | tfhd |  |  |  |  | 8.8.7 | track fragment header |
|  |  | trun |  |  |  |  | 8.8.8 | track fragment run |
|  |  | sbgp |  |  |  |  | 8.9.2 | sample-to-group |
|  |  | sgpd |  |  |  |  | 8.9.3 | sample group description |
|  |  | subs |  |  |  |  | 8.7.7 | sub-sample information |
|  |  | saiz |  |  |  |  | 8.7.8 | sample auxiliary information sizes |
|  |  | saio |  |  |  |  | 8.7.9 | sample auxiliary information offsets |
|  |  | tfdt |  |  |  |  | 8.8.12 | track fragment decode time |
|  |  | meta |  |  |  |  | 8.11.1 | metadata |
|  |  | udta |  |  |  |  | 8.10.1 | user-data |
|  | udta |  |  |  |  |  | 8.10.1 | user-data |
| mfra |  |  |  |  |  |  | 8.8.9 | movie fragment random access |
|  | tfra |  |  |  |  |  | 8.8.10 | track fragment random access |
|  | mfro |  |  |  |  |  | 8.8.11 | movie fragment random access offset |
| mdat |  |  |  |  |  |  | 8.1.1 | media data container |
| free |  |  |  |  |  |  | 8.1.2 | free space |
| skip |  |  |  |  |  |  | 8.1.2 | free space |
| imda |  |  |  |  |  |  | 8.1.4 | media data container that contains an identifier to be used with data references |
| meta |  |  |  |  |  |  | 8.11.1 | metadata |
|  | hdlr |  |  |  |  |  | 8.4.3 | handler, declares the metadata (handler) type |
|  | dinf |  |  |  |  |  | 8.7.1 | data information box, container |
|  |  | dref |  |  |  |  | 8.7.2 | data reference box, declares source(s) of metadata items |
|  | iloc |  |  |  |  |  | 8.11.3 | item location |
|  | ipro |  |  |  |  |  | 8.11.5 | item protection |
|  |  | sinf |  |  |  |  | 13.4.2 | protection scheme information box |
|  |  |  | frma |  |  |  | 13.4.3 | original format box |
|  |  |  | schm |  |  |  | 13.4.6 | scheme type box |
|  |  |  | schi |  |  |  | 13.4.7 | scheme information box |
|  | iinf |  |  |  |  |  | 8.11.6 | item information |
|  | xml |  |  |  |  |  | 8.11.2 | XML container |
|  | bxml |  |  |  |  |  | 8.11.2 | binary XML container |
|  | pitm |  |  |  |  |  | 8.11.4 | primary item reference |
|  | fiin |  |  |  |  |  | 8.12.2 | file delivery item information |
|  |  | paen |  |  |  |  | 8.12.2 | partition entry |
|  |  |  | fire |  |  |  | 8.12.7 | file reservoir |
|  |  |  | fpar |  |  |  | 8.12.3 | file partition |
|  |  |  | fecr |  |  |  | 8.12.4 | FEC reservoir |
|  |  | segr |  |  |  |  | 8.12.5 | file delivery session group |
|  |  | gitn |  |  |  |  | 8.12.6 | group id to name |
|  | idat |  |  |  |  |  | 8.11.11 | item data |
|  | iref |  |  |  |  |  | 8.11.12 | item reference |
|  | grpl |  |  |  |  |  | 8.15.2 | entities groups list |
|  |  | prsl |  |  |  |  | 8.15.4.1 | preselection group |
|  |  |  | elng |  |  |  | 8.4.6 | extended language tag |
|  |  |  | udta |  |  |  | 8.10.1 | user-data |
|  |  |  | kind |  |  |  | 8.10.4 | track kind |
|  |  |  | labl |  |  |  | 8.10.5 | label |
|  |  |  | chnl |  |  |  | 12.2.4 | channel layout |
|  |  |  | ardi |  |  |  | 12.2.8 | audio rendering indication |
|  |  |  | aelm |  |  |  | 12.2.9 | audio element |
|  |  |  |  | elng |  |  | 8.4.6 | extended language tag |
|  |  |  |  | kind |  |  | 8.10.4 | track kind |
|  |  |  |  | labl |  |  | 8.10.5 | label |
|  |  |  |  | chnl |  |  | 12.2.4 | channel layout |
|  |  |  |  | aedb |  |  | 12.2.10 | audio element description |
|  |  |  |  |  | aepp |  | 12.2.11 | audio element positioning interactivity polar |
|  |  |  |  |  | aepr |  | 12.2.12 | audio element prominence interactivity |
|  |  |  | aesb |  |  |  | 12.2.13 | audio element selection |
|  |  |  |  | labl |  |  | 8.10.5 | label |
|  |  |  |  | aelm |  |  | 12.2.9 | audio element |
|  |  |  |  |  | elng |  | 8.4.6 | extended language tag |
|  |  |  |  |  | kind |  | 8.10.4 | track kind |
|  |  |  |  |  | labl |  | 8.10.5 | label |
|  |  |  |  |  | chnl |  | 12.2.4 | channel layout |
|  |  |  |  |  | aedb |  | 12.2.10 | audio element description |
|  |  |  |  |  |  | aepp | 12.2.11 | audio element positioning interactivity polar |
|  |  |  |  |  |  | aepr | 12.2.12 | audio element prominence interactivity |
|  |  |  |  | aepp |  |  | 12.2.11 | audio element positioning interactivity polar |
|  |  |  |  | aepr |  |  | 12.2.12 | audio element prominence interactivity |
|  |  |  |  | aesd |  |  | 12.2.14 | audio element selection description |
| styp |  |  |  |  |  |  | 8.14.2 | segment type |
| sidx |  |  |  |  |  |  | 8.14.3 | segment index |
| ssix |  |  |  |  |  |  | 8.14.4 | subsegment index |
| prft |  |  |  |  |  |  | 8.14.5 | producer reference time |
| !mov |  |  |  |  |  |  | 8.16.6 | compressed movie box |
| !mof |  |  |  |  |  |  | 8.16.7 | compressed movie fragment box |
| !six |  |  |  |  |  |  | 8.16.8 | compressed segment index box |
| !ssx |  |  |  |  |  |  | 8.16.9 | compressed subsegment index box |

*Replace the subclause 8.2.1.1 with the following:*

|  |  |
| --- | --- |
| Box Type: | 'moov' |
| Container: | File |
| Mandatory: | No |
| Quantity: | Zero or one |

The structure-data for a presentation is stored in the single MovieBox which occurs at the top-level of a file. Normally this box is close to the beginning or end of the file, though this is not required.

*Replace the following text from subclause 8.7.2.1:*

|  |  |
| --- | --- |
| Box Types: | 'url ', 'urn ' |
| Container: | DataReferenceBox |
| Mandatory: | Yes (at least one of 'url ' or 'urn ' shall be present) |
| Quantity: | One or more |

*with:*

|  |  |
| --- | --- |
| Box Types: | 'url ', 'urn ' |
| Container: | DataReferenceBox |
| Mandatory: | No |
| Quantity: | Zero or more |