INTERNATIONAL STANDARD

ISO/IEC 14496-12

Fifth edition
2015-12-15

AMENDMENT 2
2018-02

Information technology — Coding of audio-visual objects —

Part 12: **ISO base media file format**

AMENDMENT 2: Support for image file format

Technologies de l'information — Codage des objets audiovisuels —

Partie 12: Format ISO de base pour les fichiers médias

AMENDEMENT 2: Support pour fichiers au format image

Reference number
ISO/IEC 14496-12:2015/Amd.2:2018(E)

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

Published in Switzerland

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO 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).

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

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

For an explanation on the 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.

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

A list of all parts in the ISO 14496 series can be found on the ISO website.

Information technology — Coding of audio-visual objects —

Part 12: ISO base media file format

AMENDMENT 2: Support for image file format

3.1

Add term entry and renumber accordingly so that all terms and definitions appear in alphabetical order:

3.1.1

thumbnail image

smaller-resolution representation of an image

8.3.3.3

Add the following additional `reference_type` values for `TrackReferenceBox` in 8.3.3.3:

'`thmb`': this track contains thumbnail images for the referenced track. A thumbnail track shall not be linked to another thumbnail track with the '`thmb`' item reference.

'`auxl`': this track contains auxiliary media for the indicated track (e.g. depth map or alpha plane for video).

NOTE 1 A track with reference type '`auxl`' might have a coding dependency; its use is clarified by specifications that use it.

NOTE 2 When multiple track references would describe an auxiliary video track, derived specifications might constrain or recommend which track references are used. For example, derived specifications might constrain or recommend whether to use '`vdep`' or '`auxl`' or both for auxiliary depth video track.

8.6.6.2

Replace

```
aligned(8) class EditListBox extends FullBox('elst', version, 0) {
   unsigned int(32)   entry_count;
   for (i=1; i <= entry_count; i++) {
      if (version==1) {
         unsigned int(64) segment_duration;
         int(64) media_time;
      } else { // version==0
         unsigned int(32) segment_duration;
         int(32)   media_time;
      }
      int(16) media_rate_integer;
      int(16) media_rate_fraction = 0;
   }
}
```

with

```
aligned(8) class EditListBox extends FullBox('elst', version, flags) {
   unsigned int(32)   entry_count;
   for (i=1; i <= entry_count; i++) {
      if (version==1) {
         unsigned int(64) segment_duration;
         int(64) media_time;
      } else { // version==0
         unsigned int(32) segment_duration;
         int(32)   media_time;
      }
      int(16) media_rate_integer;
      int(16) media_rate_fraction = 0;
   }
}
```


8.6.6.3

Add after the definition of “version”:

> `flags` specifies repetition of the edit list as follows. (`flags` & 1) equal to 0 specifies that the edit list is not repeated, while (`flags` & 1) equal to 1 specifies that the edit list is repeated. The values of `flags` greater than 1 are reserved. When an `EditListBox` indicates the playback of zero or one samples, (`flags` & 1) shall be equal to 0.

NOTE When the edit list is repeated, media at time 0 resulting from the edit list follows immediately the media having the largest time resulting from the edit list. In other words, the edit list is repeated seamlessly.

8.11.6.1

Add to the end of 8.11.6.1:

The `flags` field of `ItemInfoEntry` with `version` greater than or equal to 2 is specified as follows:

(`flags` & 1) equal to 1 indicates that the item is not intended to be a part of the presentation,
(`flags` & 1) equal to 0 indicates that the item is intended to be a part of the presentation.

8.11.6.2

Replace

```
aligned(8) class ItemInfoEntry
   extends FullBox('infe', version, 0) {
```

with

```
aligned(8) class ItemInfoEntry
   extends FullBox('infe', version, flags) {
```

8.11.14

Add the following as 8.11.14, renumbering as needed:

8.11.14 Item Properties Box

8.11.14.1 Definition

Box Type: `'iprp'`

Container: `MetaBox ('meta')`

Mandatory: No

Quantity: Zero or one

The `ItemPropertiesBox` enables the association of any item with an ordered set of item properties. Item properties are small data records.

The `ItemPropertiesBox` consists of two parts: `ItemPropertyContainerBox` that contains an implicitly indexed list of item properties, and one or more `ItemPropertyAssociationBox`(es) that associate items with item properties.

Each item property is a `Box` or `FullBox`. The `boxtype` of the item property specifies the property type. The `FreeSpaceBox` may occur in the `ItemPropertyContainerBox`; it has no meaning, and should not be associated with any item.

Each property association may be marked as either essential or non-essential. A reader shall not process an item that is associated with a property that is not recognized or not supported by the reader and that is marked as essential to the item. A reader may ignore an associated item property that is marked non-essential to the item.

Specifications deriving from this specification may specify property types and the respective item property box definitions as well as constraints and requirements for the property associations.

When defining item properties, it is recommended that they be small. When large data records need to be associated with an item, a separate item and item reference are more suitable.

Each ItemPropertyAssociationBox shall be ordered by increasing item_ID, and there shall be at most one occurrence of a given item_ID, in the set of ItemPropertyAssociationBox boxes. The version 0 should be used unless 32-bit item_ID values are needed; similarly, flags should be equal to 0 unless there are more than 127 properties in the ItemPropertyContainerBox. There shall be at most one ItemPropertyAssociationBox with a given pair of values of version and flags.

8.11.14.2 Syntax

```
aligned(8) class ItemProperty(property_type)
   extends Box(property_type)
{
}


aligned(8) class ItemFullProperty(property_type, version, flags)
   extends FullBox(property_type, version, flags)
{
}


aligned(8) class ItemPropertyContainerBox
   extends Box('ipco')
{
   Box properties[];    // boxes derived from
      // ItemProperty or ItemFullProperty or FreeSpaceBox(es)
      // to fill the box
}

aligned(8) class ItemPropertyAssociationBox
   extends FullBox('ipma', version, flags)

{
   unsigned int(32) entry_count;
   for(i = 0; i < entry_count; i++) {
      if (version < 1)
         unsigned int(16)   item_ID;
      else
         unsigned int(32)   item_ID;
      unsigned int(8) association_count;
      for (i=0; i<association_count; i++) {
         bit(1) essential;
         if (flags & 1)
            unsigned int(15) property_index;
         else
            unsigned int(7) property_index;
      }
   }
}


aligned(8) class ItemPropertiesBox
      extends Box('iprp') {
   ItemPropertyContainerBox property_container;
   ItemPropertyAssociationBox association[];
}
```

8.11.14.3 Semantics

`item_ID` identifies the item with which properties are associated

`essential` when set to 1 indicates that the associated property is essential to the item, otherwise it is non-essential

`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`.

8.18

Add the following as 8.18, renumbering as needed:

8.18 Entity grouping

8.18.1 General

An entity group is a grouping of items, which may also group tracks. The entities in an entity group share a particular characteristic or have a particular relationship, as indicated by the grouping type.

Entity groups are indicated in `GroupsListBox`. Entity groups specified in `GroupsListBox` of a file-level `MetaBox` refer to tracks or file-level items. Entity groups specified in `GroupsListBox` of a movie-level `MetaBox` refer to movie-level items. Entity groups specified in `GroupsListBox` of a track-level `MetaBox` refer to track-level items of that track.

`GroupsListBox` contains `EntityToGroupBoxes`, each specifying one entity group.

8.18.2 Groups List box

8.18.2.1 Definition

Box Type: `'grpl'`

Container: `MetaBox` that is not contained in `AdditionalMetadataContainerBox`

Mandatory: No

Quantity: Zero or One

The `GroupsListBox` includes the entity groups specified for the file. This box contains a set of full boxes, each called an `EntityToGroupBox`, with four-character codes denoting a defined grouping type.

The `GroupsListBox` shall not be present in `AdditionalMetadataContainerBox`.

When `GroupsListBox` is present in a file-level `MetaBox`, there shall be no `item_ID` value in `ItemInfoBox` in any file-level `MetaBox` that is equal to the `track_ID` value in any `TrackHeaderBox`.

8.18.2.2 Syntax

```
aligned(8) class GroupsListBox extends Box('grpl') {
}
```