INTERNATIONAL STANDARD

ISO/IEC 14496-12

Fourth edition
2012-09-15

AMENDMENT 1
2013-09-15

Information technology — Coding of audio-visual objects —

Part 12:
ISO base media file format

AMENDMENT 1: Various enhancements including support for large metadata

Technologies de l'information — Codage des objets audiovisuels —

Partie 12: Format ISO de base pour les fichiers médias

AMENDEMENT 1: Plusieurs améliorations, y compris un support pour métadonnées volumineuses

Reference number
ISO/IEC 14496-12:2012/Amd.1:2013(E)

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

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.

Amendment 1 to ISO/IEC 14496-12:2012 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

Information technology — Coding of audio-visual objects —

Part 12:
ISO base media file format

AMENDMENT 1: Various enhancements including support for large metadata

In 8.6.1.4.1, add Track Extension Properties Box (`'trep'`) as a container box as follows:

Box Type:	`'cslg'`
Container:	Sample Table Box (`'stbl'`) or Track Extension Properties Box (`'trep'`)
Mandatory:	No
Quantity:	Zero or one

Add the following sentence after the sentence starting with "When the Composition to Decode Box is included in the Sample Table Box":

When the Composition to Decode Box is included in the Track Extension Properties Box, it documents the composition and decoding time relations of the samples in all movie fragments following the Movie Box.

In 8.8, add the following Subclauses:

8.8.12 Track Extension Properties Box

8.8.12.1 Definition

Box Type:	`'trep'`
Container:	Movie Extends Box (`'mvex'`)
Mandatory:	No
Quantity:	Zero or more. (Zero or one per track)

This box can be used to document or summarize characteristics of the track in the subsequent movie fragments. It may contain any number of child boxes.

8.8.12.2 Syntax

```
class TrackExtensionPropertiesBox extends FullBox('trep', 0, 0) {
   unsigned int(32) track_id;
   // Any number of boxes may follow
}
```

8.8.12.3 Semantics

`track_id` indicates the track for which the track extension properties are provided in this box.

Add the following Subclauses:

8.8.13 Alternative Startup Sequence Properties Box

8.8.13.1 Definition

Box Type: `'assp'`
Container: Track Extension Properties Box (`'trep'`)
Mandatory: No
Quantity: Zero or one

This box indicates the properties of alternative startup sequence sample groups in the subsequent track fragments of the track indicated in the containing Track Extension Properties box.

Version 0 of the Alternative Startup Sequence Properties box shall be used if version 0 of the Sample to Group box is used for the alternative startup sequence sample grouping. Version 1 of the Alternative Startup Sequence Properties box shall be used if version 1 of the Sample to Group box is used for the alternative startup sequence sample grouping.

8.8.13.2 Syntax

```
class AlternativeStartupSequencePropertiesBox extends FullBox('assp', version, 0)
{
   if (version == 0) {
      signed int(32)     min_initial_alt_startup_offset;
   }
   else if (version == 1) {
      unsigned int(32)  num_entries;
      for (j=1; j <= num_entries; j++) {
         unsigned int(32)  grouping_type_parameter;
         signed int(32)     min_initial_alt_startup_offset;
      }
   }
}
```

8.8.13.3 Semantics

`min_initial_alt_startup_offset`: No value of sample_offset[1] of the referred sample group description entries of the alternative startup sequence sample grouping shall be smaller than min_initial_alt_startup_offset. In version 0 of this box, the alternative startup sequence sample grouping using version 0 of the Sample to Group box is referred to. In version 1 of this box, the alternative startup sequence sample grouping using version 1 of the Sample to Group box is referred to as further constrained by `grouping_type_parameter`.

`num_entries` indicates the number of alternative startup sequence sample groupings documented in this box.

`grouping_type_parameter` indicates which one of the alternative sample groupings this loop entry applies to.

In 8.11.3.1 Definition, replace:

The box starts with three or four values, specifying the size in bytes of the `offset` field, `length` field, `base_offset` field, and, in version 1 of this box, the `extent_index` fields, respectively. These values must be from the set {0, 4, 8}.

with:

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 `extent_index` fields, respectively. These values must be from the set {0, 4, 8}.

Replace:

For maximum compatibility, version 0 of this box should be used in preference to version 1 with construction_method==0, when possible.

with:

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.

In 8.11.3.2 Syntax, replace a definition of "aligned(8) class ItemLocationBox" with:

```
aligned(8) class ItemLocationBox extends FullBox('iloc', version, 0) {
   unsigned int(4)    offset_size;
   unsigned int(4)    length_size;
   unsigned int(4)    base_offset_size;
   if ((version == 1) || (version == 2)) {
      unsigned int(4)    index_size;
   } else {
      unsigned int(4)    reserved;
   }
   if (version < 2) {
      unsigned int(16)   item_count;
   } else if (version == 2) {
      unsigned int(32)   item_count;
   }
   for (i=0; i<item_count; i++) {
      if (version < 2) {
         unsigned int(16)   item_ID;
      } else if (version == 2) {
         unsigned int(32)   item_ID;
      }
      if ((version == 1) || (version == 2)) {
         unsigned int(12)   reserved = 0;
         unsigned int(4)    construction_method;
      }
      unsigned int(16)   data_reference_index;
      unsigned int(base_offset_size*8)   base_offset;
      unsigned int(16)      extent_count;
      for (j=0; j<extent_count; j++) {
         if (((version == 1) || (version == 2)) && (index_size > 0)) {
            unsigned int(index_size*8)   extent_index;
         }
         unsigned int(offset_size*8) extent_offset;
         unsigned int(length_size*8) extent_length;
      }
   }
}
```

In 8.11.4.2, replace a definition of "aligned(8) class PrimaryItemBox" with:

```
aligned(8) class PrimaryItemBox
      extends FullBox('pitm', version, 0) {
   if (version == 0) {
      unsigned int(16)   item_ID;
   } else {
      unsigned int(32)   item_ID;
   }
}
```

In 8.11.4.3, replace:

`item_ID` is the identifier of the primary item *with:*

`item_ID` is the identifier of the primary item. Version 1 should only be used when large `item_ID` values (exceeding 65535) are required or expected to be required.

In 8.11.6.1, replace:

Three versions of the item info entry are defined. Version 1 includes additional information to version 0 as specified by an extension type. For instance, it shall be used with extension type `'fdel'` for items that are referenced by the file partition box (`'fpar'`), which is defined for source file partitionings and applies to file delivery transmissions. Version 2 provides an alternative structure in which metadata item types are indicated by a 32-bit (typically 4-character) registered or defined code; two of these codes are defined to indicate a MIME type or metadata typed by a URI.

If no extension is desired, the box may terminate without the `extension_type` field and the extension; if, in addition, `content_encoding` is not desired, that field also may be absent and the box terminate before it. If an extension is desired without an explicit `content_encoding`, a single null byte, signifying the empty string, must be supplied for the `content_encoding`, before the indication of `extension_type`.

If file delivery item information is needed and a version 2 ItemInfoEntry is used, then the file delivery information is stored (a) as a separate item of type 'fdel') (b) linked by an item reference from the item, to the file delivery information, of type 'fdel'. There must be exactly one such reference if file delivery information is needed.

It is possible that there are valid URI forms for MPEG-7 metadata (e.g. a schema URI with a fragment identifying a particular element), and it may be possible that these structures could be used for MPEG-7. However, there is explicit support for MPEG-7 in ISO base media file format family files, and this explicit support is preferred as it allows, among other things:

a) incremental update of the metadata (logically, I/P coding, in video terms) whereas this draft is 'I-frame only';

b) binarization and thus compaction;

c) the use of multiple schemas.

Therefore, the use of these structures for MPEG-7 is deprecated (and undocumented).

Information on URI forms for some metadata systems can be found in Annex G.

with:

Four versions of the item info entry are defined. Version 1 includes additional information to version 0 as specified by an extension type. For instance, it shall be used with extension type `'fdel'` for items that are referenced by the file partition box (`'fpar'`), which is defined for source file partitionings and applies to file delivery transmissions. Versions 2 and 3 provide an alternative structure in which metadata item types are indicated by a 32-bit (typically 4-character) registered or defined code; two of these codes are defined to indicate a MIME type or metadata typed by a URI. Version 2 supports 16-bit item_ID values, whereas version 3 supports 32-bit item_ID values.

If no extension is desired, the box may terminate without the `extension_type` field and the extension; if, in addition, `content_encoding` is not desired, that field also may be absent and the box terminate before it. If an extension is desired without an explicit `content_encoding`, a single null byte, signifying the empty string, must be supplied for the `content_encoding`, before the indication of `extension_type`.

If file delivery item information is needed and a version 2 or 3 ItemInfoEntry is used, then the file delivery information is stored (a) as a separate item of type 'fdel') (b) linked by an item reference from the item, to the file delivery information, of type ‘fdel’. There must be exactly one such reference if file delivery information is needed.

It is possible that there are valid URI forms for MPEG-7 metadata (e.g. a schema URI with a fragment identifying a particular element), and it may be possible that these structures could be used for MPEG-7. However, there is explicit support for MPEG-7 in ISO base media file format family files, and this explicit support is preferred as it allows, among other things:

a) incremental update of the metadata (logically, I/P coding, in video terms) whereas this draft is ‘I-frame only’;

b) binarization and thus compaction;

c) the use of multiple schemas.

Therefore, the use of these structures for MPEG-7 is deprecated (and undocumented).

Information on URI forms for some metadata systems can be found in Annex G.

Version 1 of `ItemInfoBox` should only be used when support for a large number of `itemInfoEntries` (exceeding 65535) is required or expected to be required.

In 8.11.6.2, replace definitions of "aligned(8) class ItemInfoEntry" and "aligned(8) class ItemInfoBox" with:

```
aligned(8) class ItemInfoEntry
      extends FullBox(‘infe’, version, 0) {
   if ((version == 0) || (version == 1)) {
      unsigned int(16)   item_ID;
      unsigned int(16)   item_protection_index
      string             item_name;
      string             content_type;
      string             content_encoding;   //optional
   }
   if (version == 1) {
      unsigned int(32)   extension_type;     //optional
      ItemInfoExtension(extension_type);     //optional
   }
   if (version >= 2) {
      if (version == 2) {
         unsigned int(16)   item_ID;
      } else if (version == 3) {
         unsigned int(32)   item_ID;
      }
      unsigned int(16)   item_protection_index;
      unsigned int(32)   item_type;
      string            item_name;
      if (item_type==’mime’) {
         string         content_type;
         string         content_encoding;   //optional
      } else if (item_type == ‘uri ‘) {
         string         item_uri_type;
      }
   }
}
aligned(8) class ItemInfoBox
      extends FullBox(‘iinf’, version, 0) {
   if (version == 0) {
      unsigned int(16)  entry_count;
   } else {
      unsigned int(32) entry_count;
   }
   ItemInfoEntry[ entry_count ]     item_infos;
}
```

In 8.11.12.1 Definition, replace:

The item reference box allows the linking of one item to others via typed references. All the references for one item of a specific type are collected into a single item type reference box, whose type is the reference type, and which has a ‘from item ID’ field indicating which item is linked. The items linked to are then represented by an array of ‘to item ID’s. All these single item type reference boxes are then collected into the item reference box. The reference types defined for the track reference box defined in 8.3.3 may be used here if appropriate, or other registered reference types.

with:

The item reference box allows the linking of one item to others via typed references. All the references for one item of a specific type are collected into a single item type reference box, whose type is the reference type, and which has a ‘from item ID’ field indicating which item is linked. The items linked to are then represented by an array of ‘to item ID’s. All these single item type reference boxes are then collected into the item reference box. The reference types defined for the track reference box defined in 8.3.3 may be used here if appropriate, or other registered reference types. Version 1 of `ItemReferenceBox` with `SingleItemReferenceBoxLarge` should only be used when large `from_item_ID` or `to_item_ID` values (exceeding 65535) are required or expected to be required.

Replace 8.11.12.2 Syntax with:

```
aligned(8) class SingleItemTypeReferenceBox(referenceType) extends
Box(referenceType) {
   unsigned int(16) from_item_ID;
   unsigned int(16) reference_count;
   for (j=0; j<reference_count; j++) {
      unsigned int(16) to_item_ID;
   }
}
aligned(8) class SingleItemTypeReferenceBoxLarge(referenceType) extends
Box(referenceType) {
   unsigned int(32) from_item_ID;
   unsigned int(16) reference_count;
   for (j=0; j<reference_count; j++) {
      unsigned int(32) to_item_ID;
   }
}

aligned(8) class ItemReferenceBox extends FullBox('iref', version, 0) {
   if (version==0) {
      SingleItemTypeReferenceBox    references[];
   } else if (version==1) {
      SingleItemTypeReferenceBoxLarge   references[];
   }
}
```

In 8.13.3.1, replace:

The File Partition box identifies the source file and provides a partitioning of that file into source blocks and symbols. Further information about the source file, e.g., filename, content location and group IDs, is contained in the Item Information box (`'iinf'`), where the Item Information entry corresponding to the item ID of the source file is of version 1 and includes a File Delivery Item Information Extension (`'fdel'`).

with:

The File Partition box identifies the source file and provides a partitioning of that file into source blocks and symbols. Further information about the source file, e.g., filename, content location and group IDs, is contained in the Item Information box (`'iinf'`), where the Item Information entry corresponding to the item ID of the source file is of version 1 and includes a File Delivery Item Information Extension (`'fdel'`). Version 1 of `FilePartitionBox` should only be used when support for large `item_ID` or `entry_count` values (exceeding 65535) is required or expected to be required.

In 8.13.3.2, replace a definition of "aligned(8) class FilePartitionBox" with:

```
aligned(8) class FilePartitionBox
      extends FullBox('fpar', version, 0) {
   if (version == 0) {
      unsigned int(16)  item_ID;
   } else {
      unsigned int(32)  item_ID;
   }
   unsigned int(16)  packet_payload_size;
   unsigned int(8)   reserved = 0;
   unsigned int(8)   FEC_encoding_ID;
   unsigned int(16)  FEC_instance_ID;
   unsigned int(16)  max_source_block_length;
   unsigned int(16)  encoding_symbol_length;
   unsigned int(16)  max_number_of_encoding_symbols;
   string            scheme_specific_info;
   if (version == 0) {
      unsigned int(16)  entry_count;
   } else {
      unsigned int(32)  entry_count;
   }
   for (i=1; i <= entry_count; i++) {
      unsigned int(16)  block_count;
      unsigned int(32)  block_size;
   }
}
```

In 8.13.4.1, replace:

The FEC reservoir box associates the source file identified in the file partition box (`'fpar'`) with FEC reservoirs stored as additional items. It contains a list that starts with the first FEC reservoir associated with the first source block of the source file and continues sequentially through the source blocks of the source file.

with:

The FEC reservoir box associates the source file identified in the file partition box (`'fpar'`) with FEC reservoirs stored as additional items. It contains a list that starts with the first FEC reservoir associated with the first source block of the source file and continues sequentially through the source blocks of the source file. Version 1 of `FECReservoirBox` should only be used when support for large `item_ID` values and `entry_count` (exceeding 65535) is required or expected to be required.

In 8.13.4.2, replace a definition of "aligned(8) class FECReservoirBox" with:

```
aligned(8) class FECReservoirBox
      extends FullBox('fecr', version, 0) {
   if (version == 0) {
      unsigned int(16)  entry_count;
   } else {
      unsigned int(32)  entry_count;
   }
   for (i=1; i <= entry_count; i++) {
      if (version == 0) {
         unsigned int(16)  item_ID;
      } else {
         unsigned int(32)  item_ID;
      }
      unsigned int(32)  symbol_count;
   }
}
```

In 8.13.7.1, replace:

The File reservoir box associates the source file identified in the file partition box ('fpar') with File reservoirs stored as additional items. It contains a list that starts with the first File reservoir associated with the first source block of the source file and continues sequentially through the source blocks of the source file.

with:

The File reservoir box associates the source file identified in the file partition box ('fpar') with File reservoirs stored as additional items. It contains a list that starts with the first File reservoir associated with the first source block of the source file and continues sequentially through the source blocks of the source file. Version 1 of `FileReservoirBox` should only be used when support for large `item_ID` or `entry_count` values (exceeding 65535) is required or expected to be required.

In 8.13.7.2, replace a definition of "aligned(8) class FileReservoirBox" with:

```
aligned(8) class FileReservoirBox
      extends FullBox('fire', version, 0) {
   if (version == 0) {
      unsigned int(16)  entry_count;
   } else {
      unsigned int(32)  entry_count;
   }
   for (i=1; i <= entry_count; i++) {
      if (version == 0) {
         unsigned int(16)  item_ID;
      } else {
         unsigned int(32)  item_ID;
      }
      unsigned int(32)  symbol_count;
   }
}
```

Replace 8.12.2 introduction and syntax as follows:

Box Types:	`'frma'`
Container:	Protection Scheme Information Box (`'sinf'`), Restricted Scheme Information Box ('rinf'), or Complete Track Information Box (`'cinf'`)
Mandatory:	Yes when used in a protected sample entry, in a restricted sample entry, or in a sample entry for an incomplete track.
Quantity:	Exactly one.

The Original Format Box `'frma'` contains the four-character-code of the original un-transformed sample description:

```
aligned(8) class OriginalFormatBox(codingname) extends Box ('frma') {
   unsigned int(32)  data_format = codingname;
                     // format of decrypted, encoded data (in case of protection)
                     // or un-transformed sample entry (in case of restriction
                     // and complete track information)
}
```

In 8.16.4.1 insert the following as an extra bullet point before the final note:

- The data ranges corresponding to partial subsegments include both Movie Fragment boxes and Media Data boxes. The first partial subsegment, i.e. the lowest level, will correspond to a Movie Fragment box as well as (parts of) Media Data box(es), whereras subsequent partial subsegments (higher levels) may correspond to (parts of) Media Data box(es) only.

In Clause 8, add the following Subclauses:

8.17 Support for Incomplete Tracks

8.17.1 General

This Subclause documents the sample entry formats for tracks that are incomplete. Incomplete tracks may contain samples that are marked empty or not received using the sample format.

Incomplete tracks may result, for example, when subsegments are received partially according to level assignments and `padding_flag` in the Level Assignment box indicates that the data in a Media Data box that is not received can be replaced by zeros. Consequently, sample data assigned to non-accessed levels is not present, and care should be taken not to attempt to process such samples. However, in partially received subsegments some tracks might remain complete in content while other tracks might be incomplete and only contain data that is included by reference into the complete tracks.

This Subclause specifies support for sample entry formats for incomplete tracks. With this support, readers can detect incomplete tracks from their sample entries and avoid processing such tracks or take the possibility of empty or not received samples into account when processing such tracks.

The support for incomplete tracks is similar to the content protection transformation where sample entries are hidden behind generic sample entries, such as `'encv'` and `'enca'`. Because the format of a sample entry varies with media-type, a different encapsulating four-character-code is used for incomplete tracks of each media type (audio, video, text etc.). They are:

Stream (Track) Type	**Sample-Entry Code**
Video	`icpv`
Audio	`icpa`
Text	`icpt`
System	`icps`
Hint	`icph`
Timed Metadata	`icpm`

Sample data of incomplete tracks may be included into samples of other tracks by reference, and hence an incomplete track should not be removed as long as any track reference points to it.

NOTE – The choice of level by the original recording client may vary over time, and at times represent the complete track. The level is not indicated here, and it is not required that the sample entry change from ‘incomplete’ to ‘complete’ when all levels were, in fact, received, for a period. Note also that the ‘original format’ may have indicated encryption, if partial reception and decryption works for that encryption format.

8.17.2 Transformation

The sample entry for a track that becomes incomplete e.g. through partial reception, should be modified as follows:

1) The four-character-code of the sample entry, e.g. `'avc1'`, is replaced by a new sample entry code `'icpv'` meaning an incomplete track.

2) A Complete Track Information box is added to the sample description, leaving all other boxes unmodified.

3) The original sample entry type, e.g. `'avc1'`, is stored within an Original Format box contained in the Complete Track Information box.