INTERNATIONAL STANDARD

ISO/IEC 14496-12

Second edition
2005-04-01

Corrected version
2005-10-01

AMENDMENT 2
2008-02-01

Information technology — Coding of audio-visual objects —

Part 12: ISO base media file format

AMENDMENT 2: Hint track format for ALC/LCT and FLUTE transmission and multiple meta box support

Technologies de l'information — Codage des objets audiovisuels —
Partie 12: Format ISO de base pour les fichiers médias

AMENDEMENT 2: Format de piste optimisé pour transmission ALC/LCT et FLUTE et support de boîte "meta" multiple

Reference number
ISO/IEC 14496-12:2005/Amd.2:2008(E)

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

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.

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 2 to ISO/IEC 14496-12:2005 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

It provides an update for the ISO base media file format by specifying server extensions for file delivery over ALC/LCT and FLUTE, instructions for streaming servers and support for multiple meta boxes.

Information technology — Coding of audio-visual objects —

Part 12:
ISO base media file format

AMENDMENT 2: Hint track format for ALC/LCT and FLUTE transmission and multiple meta box support

Replace Clause 3 with the following:

3 Terms, definitions and abbreviated terms

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

3.1 Terms and definitions

3.1.1
Box
object-oriented building block defined by a unique type identifier and length (called 'atom' in some specifications, including the first definition of MP4)

3.1.2
Chunk
contiguous set of samples for one track

3.1.3
Container Box
box whose sole purpose is to contain and group a set of related boxes

3.1.4
Hint Track
special track which does not contain media data. Instead it contains instructions for packaging one or more tracks into a streaming channel

3.1.5
Hinter
tool that is run on a file containing only media, to add one or more hint tracks to the file and so facilitate streaming

3.1.6
Movie Box
container box whose sub-boxes define the metadata for a presentation ('moov')

3.1.7
Media Data Box
container box which can hold the actual media data for a presentation ('mdat')

3.1.8
ISO Base Media File
name of the file format described in this specification

3.1.9
Presentation
one or more motion sequences (q.v.), possibly combined with audio

3.1.10
Sample
In non-hint tracks, a sample is an individual frame of video, a series of video frames in decoding order, or a compressed section of audio in decoding order. In hint tracks, a sample defines the formation of one or more streaming packets. No two samples within a track may share the same time-stamp.

3.1.11
Sample Description
structure which defines and describes the format of some number of samples in a track

3.1.12
Sample Table
packed directory for the timing and physical layout of the samples in a track

3.1.13
Track
Collection of related samples (q.v.) in an ISO base media file. For media data, a track corresponds to a sequence of images or sampled audio. For hint tracks, a track corresponds to a streaming channel.

3.2 Abbreviated terms

ALC	Asynchronous Layered Coding
FD	File Delivery
FDT	File Delivery Table
FEC	Forward Error Correction
FLUTE	File Delivery over Unidirectional Transport
IANA	Internet Assigned Numbers Authority
LCT	Layered Coding Transport
MBMS	Multimedia Broadcast/Multicast Service

In 4.2 Object Structure, after "The fields in the objects are stored with the most significant byte first, commonly known as network byte order or big-endian format.", insert the following:

When fields smaller than a byte are defined, or fields span a byte boundary, the bits are assigned from the most significant bits in each byte to the least significant. For example, a field of two bits followed by a field of six bits has the two bits in the high order bits of the byte.

In 4.3.1, add the following at the end of the subclause:

All file format brands defined in this specification are included in Annex E with a summary of which features they require.

In 6.2.3, add the following entries at the end of Table 1 (correctly cross-referenced):

	fiin						8.46.2	*file delivery item information*
		paen					8.46.2	*partition entry*
			fpar				8.46.3	*file partition*
			fecr				8.46.4	*FEC reservoir*
		segr					8.46.5	*file delivery session group*
		gitn					8.46.6	*group id to name*
		tsel					8.48.2	*track selection*
meco							8.44.9	*additional metadata container*
	mere						8.44.10	*metabox relation*

In 8.6.3, add the following reference_type to the end of the list:

- 'hind' this track depends on the referenced hint track, i.e., it should only be used if the referenced hint track is used.

In 8.9.3, replace:

`handler_type` when present in a meta box, contains an appropriate value to indicate the format of the meta box contents

with:

`handler_type` when present in a meta box, contains an appropriate value to indicate the format of the meta box contents. The value `'null'` can be used in the primary meta box to indicate that it is merely being used to hold resources.

In 8.44.1.1, replace:

"Box type: … Zero or one", i.e. the first four lines at the beginning,

with:

Box Type: 'meta'
Container: File, Movie Box ('moov'), Track Box ('trak'), or Additional Metadata Container Box ('meco')
Mandatory: No
Quantity: Zero or one (in File, 'moov', and 'trak'), One or more (in 'meco')

In 8.44.1.1 Definition, replace:

At most one meta box may occur at each of the file level, movie level, or track level.

with:

At most one meta box may occur at each of the file level, movie level, or track level, unless they are contained in an additional metadata container box ('meco').

In 8.44.6.1, insert the following text at the end of the subclause:

Two 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.

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.

In 8.44.6.2, replace the entire text with the following:

```
aligned(8) class ItemInfoExtension(unsigned int(32) extension_type)
{
}

aligned(8) class FDItemInfoExtension() extends ItemInfoExtension ('fdel')
{
   string            content_location;
   string            content_MD5;
   unsigned int(64)  content_length;
   unsigned int(64)  transfer_length;
   unsigned int(8)   entry_count;
   for (i=1; i <= entry_count; i++)
      unsigned int(32)  group_id;
}

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
   }
}

aligned(8) class ItemInfoBox
      extends FullBox('iinf', version = 0, 0) {
   unsigned int(16) entry_count;
   ItemInfoEntry[ entry_count ]     item_infos;
}
```


In 8.44.6.3, replace the entire text with the following:

`item_id` contains either 0 for the primary resource (e.g., the XML contained in an 'xml ' box) or the ID of the item for which the following information is defined.

`item_protection_index` contains either 0 for an unprotected item, or the one-based index into the item protection box defining the protection applied to this item (the first box in the item protection box has the index 1).

`item_name` is a null-terminated string in UTF-8 characters containing a symbolic name of the item (source file for file delivery transmissions).

`content_type` is a null-terminated string in UTF-8 characters with the MIME type of the item. If the item is content encoded (see below), then the content type refers to the item after content decoding.

`content_encoding` is an optional null-terminated string in UTF-8 characters used to indicate that the binary file is encoded and needs to be decoded before interpreted. The values are as defined for Content-Encoding for HTTP/1.1. Some possible values are “gzip”, “compress” and “deflate”. An empty string indicates no content encoding. Note that the item is stored after the content encoding has been applied.

`extension_type` is a is a printable four-character code that identifies the extension fields of version 1 w.r.t. version 0 of the Item information entry.

`content_location` is a null-terminated string in UTF-8 characters containing the URI of the file as defined in HTTP/1.1 (RFC 2616).

`content_MD5` is a null-terminated string in UTF-8 characters containing an MD5 digest of the file. See HTTP/1.1 (RFC 2616) and RFC 1864.

`content_length` gives the total length (in bytes) of the (un-encoded) file.

`transfer_length` gives the total length (in bytes) of the (encoded) file. Note that transfer length is equal to content length if no content encoding is applied (see above).

`entry_count` provides a count of the number of entries in the following array.

`group_ID` indicates a file group to which the file item (source file) belongs. See 3GPP TS 26.346 for more details on file groups.

Add the following subclauses before 8.45:

8.44.9 Additional Metadata Container Box

8.44.9.1 Definition

Box Type: ‘meco’
Container: File, Movie Box (‘moov’), or Track Box (‘trak’)
Mandatory: No
Quantity: Zero or one

The additional metadata container box includes one or more meta boxes. It can be carried at the top level of the file, in the Movie Box (‘moov’), or in the Track Box (‘trak’) and shall only be present if it is accompanied by a meta box in the same container. A meta box that is not contained in the additional metadata container box is the preferred (primary) meta box. Meta boxes in the additional metadata container box complement or give alternative metadata information. The usage of multiple meta boxes may be desirable when, e.g., a single handler is not capable of processing all metadata. All meta boxes at a certain level, including the preferred one and those contained in the additional metadata container box, must have different handler types.

A meta box contained in an additional metadata container box shall contain a primary Item box or the primary data box required by the handler (e.g., an XML Box). It shall not include boxes or syntax elements concerning items other than the primary item indicated by the present primary item box or XML box. URLs in a meta box contained in an additional metadata container box are relative to the context of the preferred meta box.

8.44.9.2 Syntax

```
aligned(8) class AdditionalMetadataContainerBox extends Box('meco') {
}
```

8.44.10 Metabox Relation Box

8.44.10.1 Definition

Box Type: 'mere'
Container: Additional Metadata Container Box ('meco')
Mandatory: No
Quantity: Zero or more

The metabox relation box indicates a relation between two meta boxes at the same level, i.e., the top level of the file, the Movie Box, or Track Box. The relation between two meta boxes is unspecified if there is no metabox relation box for those meta boxes. Meta boxes are referenced by specifying their handler types.

8.44.10.2 Syntax

```
aligned(8) class MetaboxRelationBox
   extends FullBox('mere', version=0, 0) {
   unsigned int(32)  first_metabox_handler_type;
   unsigned int(32)  second_metabox_handler_type;
   unsigned int(8)   metabox_relation;
}
```

8.44.10.3 Sematics

`first_metabox_handler_type` indicates the first meta box to be related.
`second_metabox_handler_type` indicates the second meta box to be related.
`metabox_relation` indicates the relation between the two meta boxes. The following values are defined:

1 the relationship between the boxes is unknown (which is the default when this box is not present);

2 the two boxes are semantically un-related (e.g., one is presentation, the other annotation);

3 the two boxes are semantically related but complementary (e.g., two disjoint sets of meta-data expressed in two different meta-data systems);

4 the two boxes are semantically related but overlap (e.g., two sets of meta-data neither of which is a subset of the other); neither is 'preferred' to the other;

5 the two boxes are semantically related but the second is a proper subset or weaker version of the first; the first is preferred;

6 the two boxes are semantically related and equivalent (e.g., two essentially identical sets of meta-data expressed in two different meta-data systems).

Add the following subclauses before Clause 9:

8.46 File Delivery Format Extensions

8.46.1 Introduction

Files intended for transmission over ALC/LCT or FLUTE are stored as items in a top-level meta box ('meta'). The item location box ('iloc') specifies the actual storage location of each item within the container file as well as the file size of each item. File name, content type (MIME type), etc., of each item are provided by version 1 of the item information box ('iinf').

Pre-computed FEC reservoirs are stored as additional items in the meta box. If a source file is split into several source blocks, FEC reservoirs for each source block are stored as separate items. The relationship between FEC reservoirs and original source items is recorded in the partition entry box ('paen') located in the FD item information box ('fiin').

See Clause 11 for more details on the usage of the file delivery format.

8.46.2 FD Item Information Box

8.46.2.1 Definition

Box Type: 'fiin'
Container: Meta Box ('meta')
Mandatory: No
Quantity: Zero or one

The FD item information box is optional, although it is mandatory for files using FD hint tracks. It provides information on the partitioning of source files and how FD hint tracks are combined into FD sessions. Each partition entry provides details on a particular file partitioning, FEC encoding and associated FEC reservoirs. It is possible to provide multiple entries for one source file (identified by its item ID) if alternative FEC encoding schemes or partitionings are used in the file. All partition entries are implicitly numbered and the first entry has number 1.

8.46.2.2 Syntax


```
aligned(8) class PartitionEntry extends Box('paen') {
   FilePartitionBox  blocks_and_symbols;
   FECReservoirBox   FEC_symbol_locations; //optional
}

aligned(8) class FDItemInformationBox
      extends FullBox('fiin', version = 0, 0) {
   unsigned int(16)  entry_count;
   PartitionEntry    partition_entries[ entry_count ];
   FDSessionGroupBox session_info;     //optional
   GroupIdToNameBox  group_id_to_name; //optional
}
```


8.46.2.3 Semantics

`entry_count` provides a count of the number of entries in the following array.

The semantics of the boxes are described where the boxes are documented.

8.46.3 File Partition Box

8.46.3.1 Definition

Box Type: 'fpar'
Container: Partition Entry ('paen')
Mandatory: Yes
Quantity: Exactly one

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').

8.46.3.2 Syntax

```
aligned(8) class FilePartitionBox
      extends FullBox('fpar', version = 0, 0) {
   unsigned int(16)  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;
   unsigned int(16)  entry_count;
   for (i=1; i <= entry_count; i++) {
      unsigned int(16)  block_count;
      unsigned int(32)  block_size;
   }
}
```

8.46.3.3 Semantics

`item_ID` references the item in the item location box ('iloc') that the file partitioning applies to.

`packet_payload_size` gives the target ALC/LCT or FLUTE packet payload size of the partitioning algorithm. Note that UDP packet payloads are larger, as they also contain ALC/LCT or FLUTE headers.

`FEC_encoding_ID` identifies the FEC encoding scheme and is subject to IANA registration (see RFC 3452). Note that i) value zero corresponds to the "Compact No-Code FEC scheme" also known as "Null-FEC" (RFC 3695); ii) value one corresponds to the "MBMS FEC" (3GPP TS 26.346); iii) for values in the range of 0 to 127, inclusive, the FEC scheme is Fully-Specified, whereas for values in the range of 128 to 255, inclusive, the FEC scheme is Under-Specified.

`FEC_instance_ID` provides a more specific identification of the FEC encoder being used for an Under-Specified FEC scheme. This value should be set to zero for Fully-Specified FEC schemes and shall be ignored when parsing a file with `FEC_encoding_ID` in the range of 0 to 127, inclusive. `FEC_instance_ID` is scoped by the `FEC_encoding_ID`. See RFC 3452 for further details.

`max_source_block_length` gives the maximum number of source symbols per source block.

`encoding_symbol_length` gives the size (in bytes) of one encoding symbol. All encoding symbols of one item have the same length, except the last symbol which may be shorter.

`max_number_of_encoding_symbols` gives the maximum number of encoding symbols that can be generated for a source block for those FEC schemes in which the maximum number of encoding symbols is relevant, such as FEC encoding ID 129 defined in RFC 3452. For those FEC schemes in which the maximum number of encoding symbols is not relevant, the semantics of this field is unspecified.

`scheme_specific_info` is a base64-encoded null-terminated string of the scheme-specific object transfer information (FEC-OTI-Scheme-Specific-Info). The definition of the information depends on the FEC encoding ID.

`entry_count` gives the number of entries in the list of (block_count, block_size) pairs that provides a partitioning of the source file. Starting from the beginning of the file, each entry indicates how the next segment of the file is divided into source blocks and source symbols.

`block_count` indicates the number of consecutive source blocks of size block_size.

`block_size` indicates the size of a block (in bytes). A block_size that is not a multiple of the encoding_symbol_length symbol size indicates that the last source symbol includes padding that is not stored in the item.