INTERNATIONAL STANDARD

ISO/IEC 23000-2

Second edition
2008-01-15

Information technology — Multimedia application format (MPEG-A) — Part 2: MPEG music player application format

Technologies de l'information — Format pour application multimédia (MPEG-A) —

Partie 2: Format pour application musicienne MPEG

Reference number
ISO/IEC 23000-2:2008(E)

© ISO/IEC 2008

Contents

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.

ISO/IEC 23000-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This second edition cancels and replaces the first edition (ISO/IEC 23000-2:2006), which has been technically revised.

ISO/IEC 23000 consists of the following parts, under the general title *Information technology — Multimedia application format (MPEG-A)*:

— *Part 1: Purpose for multimedia application formats*

— *Part 2: MPEG music player application format*

— *Part 3: MPEG photo player application format*

— *Part 4: Musical slide show application format (MAF)*

— *Part 5: Media streaming player*

— *Part 7: Open release (MAF)*

— *Part 8: Portable video player MAF*

— *Part 9: Multimedia application format for digital multimedia broadcasting*

Introduction

MPEG has developed a number of standards, all of which strive to serve the needs of consumers and industry. Among those are MPEG-4, a next-generation suite of standards for media compression, and MPEG-7, a suite of standards for meta-data representation. MPEG-4 specifies what MPEG expects to be another very successful specification, the MPEG-4 File Format, while MPEG-7 specifies not only signal-derived meta-data, but also archival meta-data such as Artist, Album and Song Title.

As such, MPEG-4 and MPEG-7 represent an ideal environment to support the current "MP3 music library" user experience, and, moreover, to extend that experience in new directions.

Firstly, this part of ISO/IEC 23000 shows how to carry MP3 information (music and meta-data) within the MPEG-4 and MPEG-7 framework. Moving MP3 into the MPEG-4 world supports, as a baseline, everything that users know and expect, but offers the capability to deliver a much richer music experience with components of MPEG-4, MPEG-7 and MPEG-21 at our disposal.

Secondly, this part of ISO/IEC 23000 builds on the music player and extends it to the Protected Music Player for both mp4 and mp21 file types including (1) mp4 file as protected content format with fixed encryption, without key management components and (2) protected mp21 file with flexible tool selection and key management components.

This part of ISO/IEC 23000 contains conformance and reference software.

Information technology — Multimedia application format (MPEG-A) —

Part 2: MPEG music player application format

1 Scope

This part of ISO/IEC 23000 presents a basic architecture for constructing an annotated music library. It defines a simple file format for songs and a file format for albums and playlists. A conformant player application has to support all these specified file formats.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 11172-3:1993, *Information technology — Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s — Part 3: Audio*

ISO/IEC 13818-3:1998, *Information technology — Generic coding of moving pictures and associated audio information — Part 3: Audio*

ISO/IEC 14496-2:2004, *Information technology — Coding of audio-visual objects — Part 2: Visual*

ISO/IEC 14496-3:2005, *Information technology — Coding of audio-visual objects — Part 3: Audio*

ISO/IEC 14496-12:2005, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 14496-13:2004, *Information technology — Coding of audio-visual objects — Part 13: Intellectual Property Management and Protection (IPMP) extensions*

ISO/IEC 14496-14:2003, *Information technology — Coding of audio-visual objects — Part 14: MP4 file format*

ISO/IEC 15938-5:2003, *Information technology — Multimedia content description interface — Part 5: Multimedia description schemes*

ISO/IEC TR 15938-8:2002, *Information technology — Multimedia content description interface — Part 8: Extraction and use of MPEG-7 descriptions*

ISO/IEC 21000-2:2005, *Information technology — Multimedia framework (MPEG-21) — Part 2: Digital Item Declaration*

ISO/IEC 21000-4:2006, *Information technology — Multimedia framework (MPEG-21) — Part 4: Intellectual Property Management and Protection Components*

ISO/IEC 21000-5:2004, *Information technology — Multimedia framework (MPEG-21) — Part 5: Rights Expression Language*

ISO/IEC 21000-9:2005, *Information technology — Multimedia framework (MPEG-21) — Part 9: File Format*

3 Overview of MPEG Standards

3.1 MPEG-1 Layer III

ISO/IEC 11172-3:1993 specifies MPEG-1 Audio. From that specification, MPEG-1 Layer III (or MP3) is one of the most widely deployed MPEG audio standards ever. Its wide appeal is due to both its good compression performance and its simplicity of implementation. The vast majority of compressed music archives use MP3 encoding.

One aspect of the simplicity of Layer III is that it specifies a self-synchronizing transport, making it amenable to both storage in a computer file and transmission over a channel without byte framing. In the context of transmission channels, Layer III can operate over a constant-rate isocronous link, and has constant-rate headers (as does Layer I and II). However Layer III is an instantaneously-variable-rate coder, which adapts to the constant-rate channel by using a “bit buffer” and “back pointers.” Each of the headers signals the start of another block of audio signal, however due to the Layer III syntax, the data associated with that next block of audio signal may be in a prior segment of the bitstream, pointed to by the back pointer (see Figure 1, specifically the curved arrow pointing to main_data_begin). We note that this is in contrast to the MPEG-4 view of data stream segmentation, in which one Access Unit contains all information necessary to decode one segment of audio.

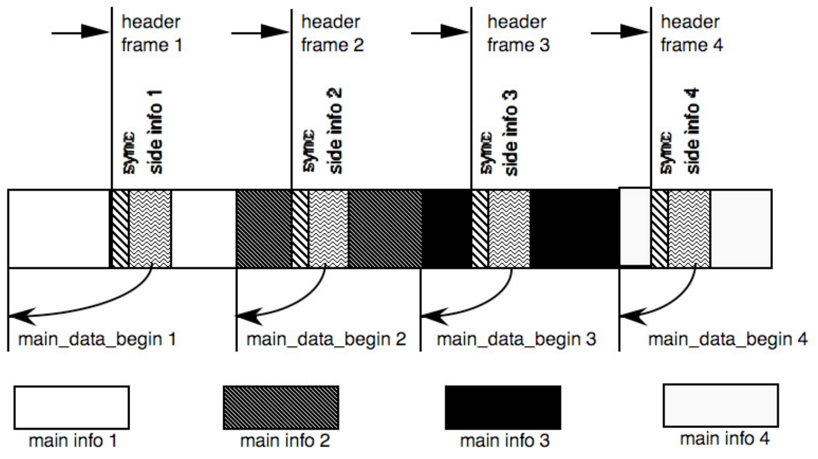

Figure 1 — Layer III bitstream organization

3.2 MPEG-4 “MPEG-1/2 Audio in MPEG-4”

ISO/IEC 14496-3:2005 subpart 9 “MPEG-1/2 Audio in MPEG-4” specifies a method for segmenting and formatting Layer III bitstreams into MPEG-4 Access Units, and therefore is often referred to as “MP3onMP4”. This consists primarily of re-arranging the compressed data associated with a given header such that it follows the header. This typically results in new segments that are no longer of constant length, but that is perfectly in accordance with the definition of MPEG-4 Access Units. See example in Figure 2.

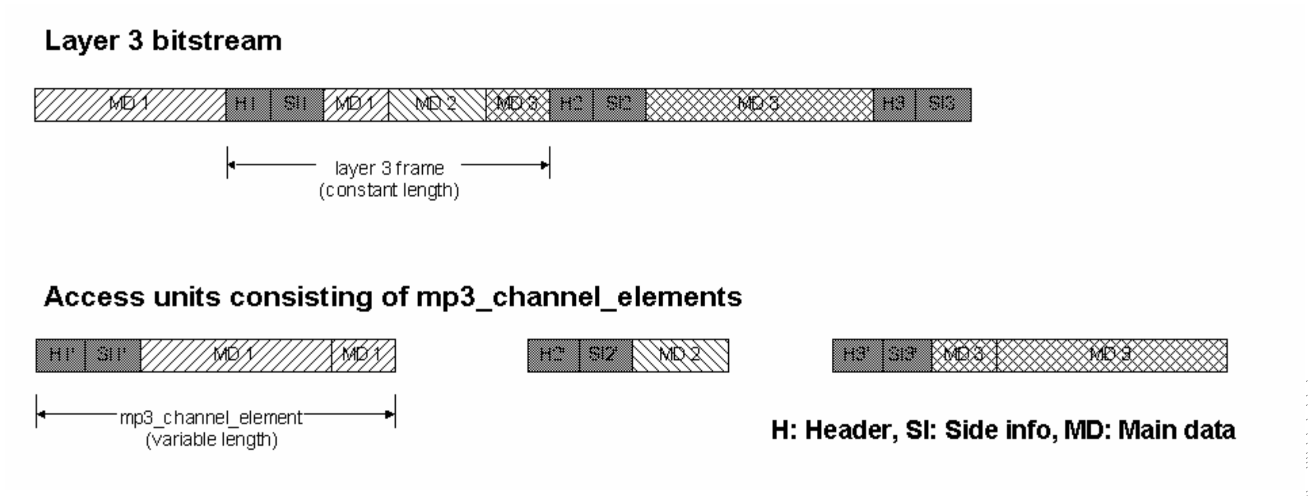

Figure 2 — Converting an MPEG-1/2 Layer 3 bitstream into mp3_channel_elements

3.3 ISO Base Media File Format

ISO/IEC 14496-12:2005 specifies the ISO Base Media File Format that is designed to contain timed media information for a presentation in a flexible, extensible format that facilitates interchange, management, editing, and presentation of the media. The ISO Base Media File Format is a base format for media file formats. In particular, the MPEG-4 file format derives from this base file format.

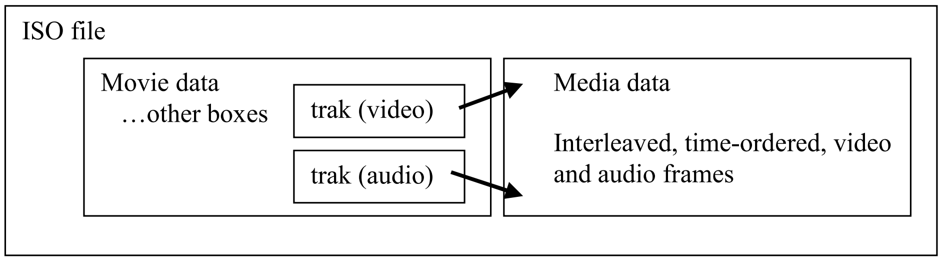

Figure 3 — Example of a simple ISO file used for interchange, containing two streams

The file structure is object-oriented as shown in Figure 3 which means that a file can be decomposed into constituent objects very simply, and the structure of the objects inferred directly from their type. The file format is designed to be independent of any particular network protocol while enabling efficient support for them in general.

3.4 MPEG-4 File Format

ISO/IEC 14496-12:2005 and ISO/IEC 14496-14:2003 together specify the MPEG-4 File Format. This supports storage of compressed audio data (e.g. MP3onMP4) in tracks. It also provides support for metadata in the form of 'meta' boxes at the File, Movie and Track level. This allows support for static (un-timed) meta-data. Figure 4 schematically illustrates the location of these un-timed MPEG-7 Metadata boxes. 4.6 provides details as to when the Metadata boxes at each level are used.

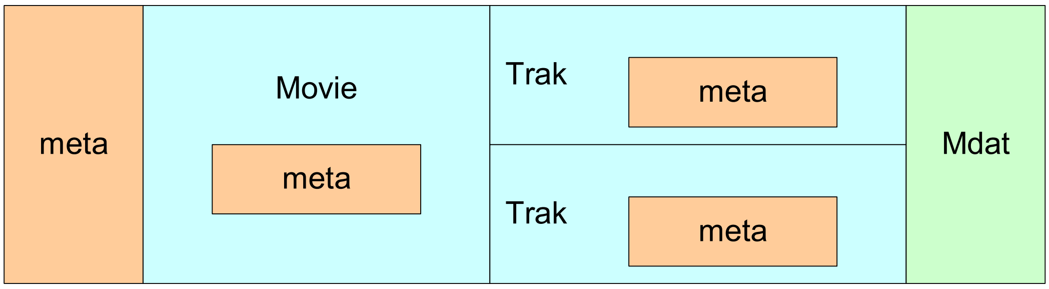

Figure 4 — Support of Static un-timed Metadata in ISO/MP4 Files

3.5 MPEG-21 File Format

ISO/IEC 21000-9:2005 specifies the MPEG-21 File Format. It uses the file-level 'meta' box of the ISO Base Media File Format to store an MPEG-21 Digital Item Declaration. The resources are either included in the file's mdat box or are externally referenced. The 'iloc' and the 'iinf' boxes in the meta box are used to address the resources. Figure 5 shows a simple example of an MPEG-21 file.

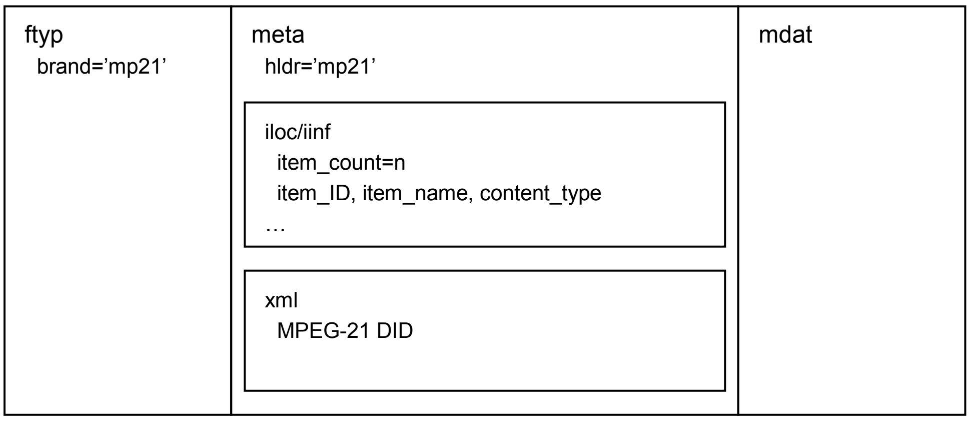

Figure 5 — Basic structure of an MPEG-21 File

3.6 MPEG-7 Multi-Media Description Scheme

ISO/IEC 15938-5:2003, the Multimedia Description Scheme (MDS) specifies all non-Visual and non-Audio specific metadata (e.g. Artist, Title, Date) in the MPEG-7 standard. As such it is able to represent all of the information found in the ID3v1 [3] metadata tagging format, as well as the corresponding information in the ID3v2 format, with its most popular version ID3v2.3 [4].

4 Song File Format

4.1 Introduction

This clause defines a format that contains a single music track with associated meta-data and a single still image containing cover art.

4.2 Audio track

The audio track contains encoded audio data according to ISO/IEC 11172-3 Layer III ("MP3") in the form of MPEG-4 audio samples.

ISO/IEC 11172-3 Layer III ("MP3") specifies a music compression scheme that results in a sequence of bits, forming a bitstream file. In contrast, ISO/IEC 14496-3:2005 specifies a music compression scheme that results in a sequence of packets (Access Units) that contain all data corresponding to one fraction of playout time. Access Units and their corresponding timing information is stored directly into the MPEG-4 File Format, specified in ISO/IEC 14496-14:2003. An MP3 bitstream can be translated into a series of MP3 Access Units using the "MP3onMP4" conversion as specified in ISO/IEC 14496-3:2005 subpart 9. This conversion allows to use mp3 encoded audio data like audio data encoded with any MPEG-4 audio codec type (e.g. AAC) within an MPEG-4 system, e.g. store the MP3 access units to an MPEG-4 file.

4.3 Meta-data

The MPEG-4 file format supports the storage of meta-data associated to a data track. Associated meta-data describing the audio track, like artist or song name, is expressed in MPEG-7 nomenclature, as specified in ISO/IEC 15938-5:2003. MP3 bitstream files can contain associated meta-data, typically ID3 tags [3,4]. The specific mapping from ID3 v1.1 tags and the corresponding ID3 v2.3 frames to MPEG-7 meta-data is show in Table 1. Parenthetical comments under Artist clarify that MPEG-7 is able to make a distinction between Artist as a *person* and Artist as a *group name*.

Table 1 — Mapping from ID3 v1.1 and ID3 v2.3 Tags to MPEG-7

ID3 v1	ID3 v2.3 Frame	Description	MPEG-7 Path
Artist	TOPE *(Original artist / performer)*	Artist performing the song	CreationInformation/Creation/Creator[Role/@href="urn:mpeg:mpeg7:RoleCS:2001:PERFORMER"]/Agent[@xsi:type="PersonType"]/Name/{FamilyName, GivenName} *(Artist Name)* CreationInformation/Creation/Creator[Role/@href="urn:mpeg:mpeg7:RoleCS:2001:PERFORMER"]/Agent[@xsi:type="PersonGroupType"]/Name *(Group Name)*
Album	TALB *(Album / Movie / Show title)*	Title of the album	CreationInformation/Creation/Title[@type="albumTitle"]
Song Title	TIT2 *(Title / Songname / Content description)*	Title of the song	CreationInformation/Creation/Title[@type="songTitle"]
Year	TORY (*Original release year)*	Year of the recording	CreationInformation/CreationCoordinates/Date/TimePoint (*Recording date.*)
Comment	COMM	Any comment of any length	CreationInformation/Creation/Abstract/FreeTextAnnotation
Track	TRCK *(Track number / Position in set)*	CD track number of song	Semantics/SemanticBase[@xsi:type="SemanticStateType"]/AttributeValuePair
Genre	TCON *(Content type)*	ID 3 V1.1 Genre ID 3 V2 Genre (4)(Eurodisco)	CreationInformation/Classification/Genre[@href="urn:id3:v1:4"] CreationInformation/Classification/Genre[@href="urn:id3:v1:4"]/Term[@termID="urn:id3:v2:Eurodisco"] CreationInformation/Classification/Genre[@href="urn:id3:v1:4"] CreationInformation/Classification/Genre[@type="secondary"][@href="urn:id3:v2:Eurodisco"]

MPEG-7 Path notation is a shorthand for the full XML notation, and an example of the correspondence between MPEG-7 Path and XML notation is shown in Annex A.

4.4 File Conversion

This subclause describes a lossless, reversible conversion of a standard mp3 bitstream file into an MPEG-4 file according to this Music Player Application Format. An MP3 bitstream file (containing mp3 audio frames and an ID3 tag) is converted using two modules, as shown in Figure 6.

The first module translates an MP3 bitstream into a series of MP3 Access Units. This is accomplished by the MP3onMP4 formatter, specified in ISO/IEC 14496-3:2005 subpart 9. The Access Units are stored into one audio track of an MPEG-4 File.

The second module extracts the meta-data information from the input file's ID3 tag and expresses it as MPEG-7 descriptor according to 4.3. This MPEG-7 meta-data is stored - together with the optional JPEG image for cover art - into the corresponding meta-box of the audio track.

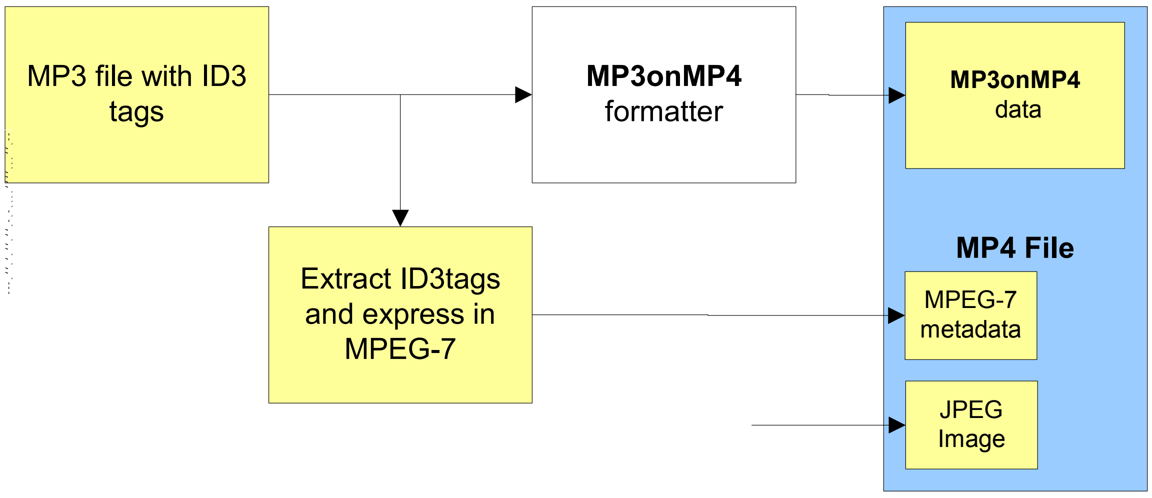

Figure 6 — Encoder System Architecture

4.5 Playback

Playback consists of

— extracting the meta-data from the MPEG-4 file and displaying it on a suitable visual interface.

— extracting the MP3onMP4 data from the MPEG-4 file, filtering it with very light-weight de-formatting operation, and playing it through a "classic" MP3 decoder.

In practice, it may be that the MP3onMP4 data is played by an "MP3onMP4 decoder," consisting of the concatenation of the MP3onMP4 deformatter and the MP3 decoder.

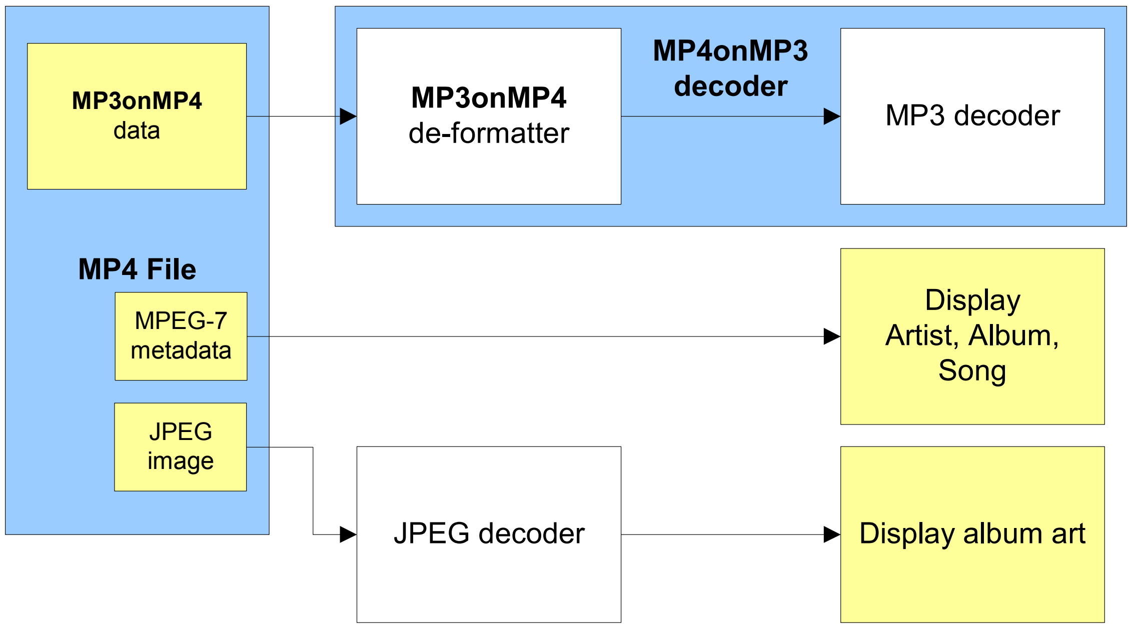

Figure 7 — Player System Architecture

4.6 File Structure

This subclause specifies the basic song file format of the Music Player Application Format in detail. The structure is based on the Playback model described above, however present a detailed view of the internals of the File Format (based on ISO/IEC 14496-12:2005), and highlight what resources a Playback application requires in order to decode the file.

For this structure (and the structures of the following Clause 5) the following information is given:

— File Example – a visual example of the file, showing the important boxes for playback (some boxes have been omitted to minimise complexity).

— File Type – A top level handler which indicates the type of file format that the structure uses. Hence if a Playback application supports either the major-brand and/or the compatible-brands *ftype* box field, it can decode this structure.

— Meta data Handler Type – MAF currently supports file level (*ftyp*) meta type handlers of mp7t and mp21.

— Resource Lookup/Playback – this is an additional section to the Playback model in order to highlight which ISO Base File Format boxes are used to decode the file.

— Notes – additional information that may aid understanding of the structure.

Additionally the following common notes apply to all structures:

NOTE The extent_count=1 for all items in the *iloc* box in each Resource Lookup/Playback section.

This subclause defines the basic structure of the Music Player Application Format specification: a MPEG-4 file with a single music track with optional JPEG image and MPEG-7 meta-data. The MPEG-7 Meta Data and optional JPEG Image are implicitly associated with the MP3 data by using the *trak* level *meta* box in the same track that contains the MP3 data.

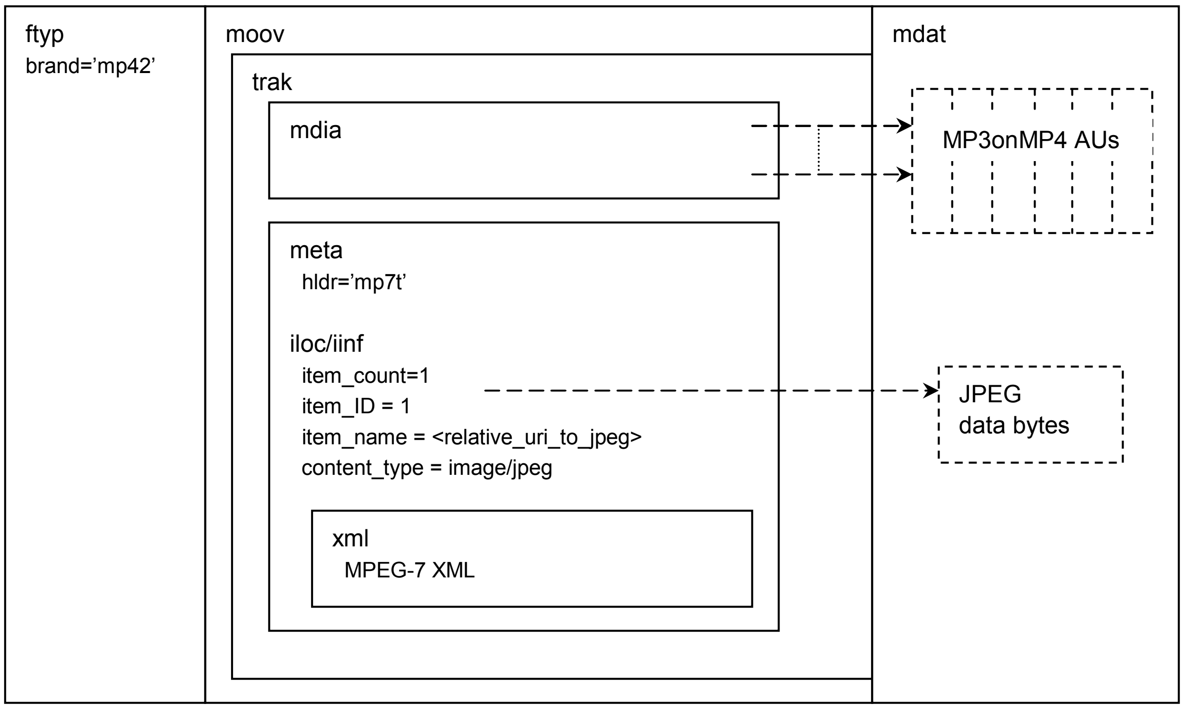

Figure 8 — Example of file structure for mp4 song file containing one audio track with MP3onMP4 audio, MPEG-7 meta-data and JPEG image

File Type: major-brand = 'mp42', compatible-brands = 'mp42', 'iso2', 'MMU2'.

Meta-data Handler Type: MPEG-7 Text (mp7t) at track level

Resource Lookup and Playback:

— A mp4 MAF application uses the *mdia* box and subsequent child boxes of the sample description to find and decode the MP3 data (stored as MP3onMP4 Access Units [14496-3:2005]).

— To present the optional JPEG Image, the application uses a combination of *iloc* and *iinf* boxes as shown below:

```
for(all items in the iloc) {
  if(iinf->content_type == image/jpeg) {
    //locate image using
          iloc->extent_offset
          iloc->extent_length
  }
}
```

NOTE 1 This structure contains MPEG-7 XML meta data pertaining to the mandatory track inside the mdat.

NOTE 2 Here the *meta* box inside the *trak* box is used to link the MP3 and the JPEG together. The coupling with the use of this box will suffice for single track MAFs.

5 Album and Playlist Format for the Music Player

5.1 Introduction

This clause describes an extension to the core song file format. It allows to create complete album files as well as playlist files that references external song files. The MPEG-21 File Format [ISO/IEC 21000-9:2005] and the MPEG-21 Digital Item Declaration (DID) [ISO/IEC 21000-2] is used to enable this functionality.

5.2 Single Track Album

The explanation of the structure starts with the special case of an album that contains only one single song.

An MPEG-21 file contains an MPEG-21 DID as entry point and a single, hidden mp4 song file that is built according to the previous Clause 4. The DID and the mp21 meta box are used to identify and locate the resources. The relationship between the mp4 song file, its MPEG-7 meta data and the optional JPEG image is held in the DID to give applications quick access to the meta-data of the album files (artist, title, cover image,…) without parsing into the hidden mp4 song file(s).

File Type: major-brand = ‘mp21’, compatible-brands = ‘mp21’, ‘MMU2’.

Meta-data Handler Type: MPEG-21 (mp21) at file level

Resource Lookup and Playback:

— The structure uses a parent *meta* box at the file level to refer to the hidden mp4 file inside the *mdat* box and directly to items (MPEG-7 meta data and JPEG image) inside the hidden *mp4 file*.

— A MAF application shall decode this file using the DID as an entry point to the resource mapping technique as described in the following stub code:

```
for (all Resources/Statement elements in a DID) {
        //Get ref attribute
       if( iinf-> item_name = ref->val && iinf->content_type = ref->mimeType) {
               //use extent_offset and extent_length  from iloc to find the //chunk of bytes
        }
}
```

— To enable this resource mapping, the DID Resource element must refer directly to the MPEG-7 and JPEG resource in the hidden *mp4 file*. This means the offset and length for these items in the *iloc* box is worked out directly from the file level *meta* box, where any complications of having to decode offsets and lengths inside the hidden mp4 file *(moov->trak->meta)* are avoided.

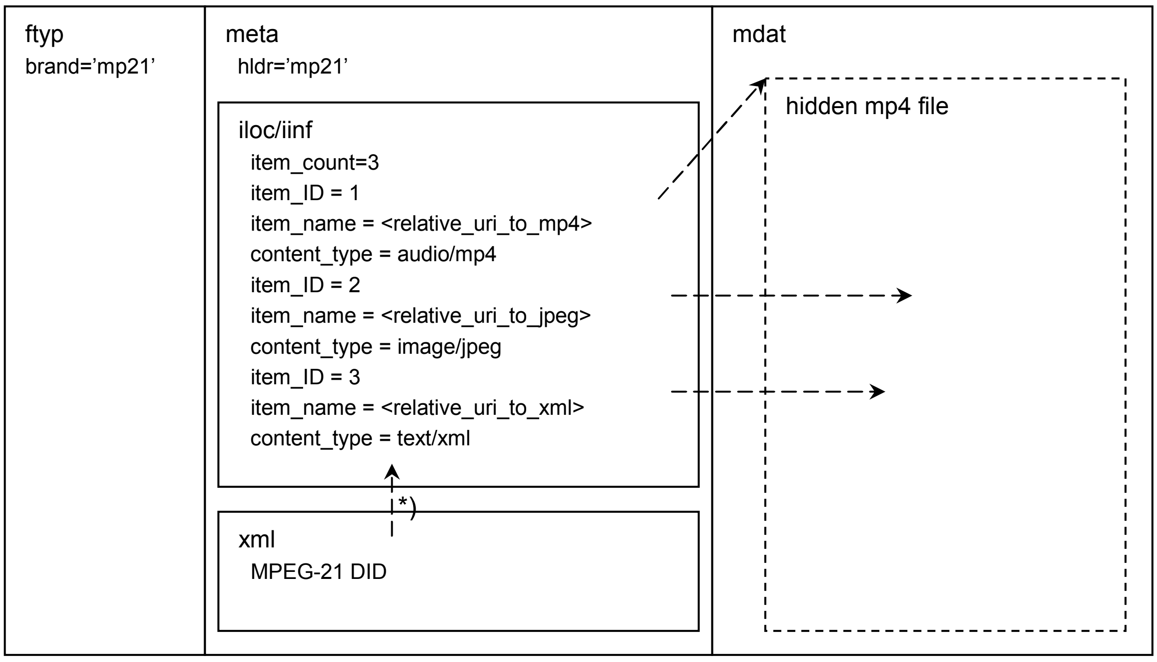

Figure 9 — Example of file structure for mp21 file containing one hidden mp4 song file
*) Arrows indicate that there is a resource mapping process from the DID to the hidden mp4 file

NOTE 1 This Figure outlines how the MPEG-7 meta data can be kept encapsulated inside the hidden *mp4 file* in the *mdat* box, as opposed to inside the DID. To do this the DID Statement element has been used (<Statement mimeType="text/xml" ref="someURI">). Since the DID Resource element and the Statement have a common "ref" attribute, the algorithm in the Resource Lookup/Playback can be applied to this DID Statement element.

NOTE 2 The hidden *mp4 file* is structured as described in Clause 4 containing the music track with MP3onMP4 audio data, MPEG-7 meta-data and optional JPEG data. The hidden *mp4 file* can be separated from the structure presented in this clause to form a stand alone file as described in Clause 4. The MPEG-7 meta-data associated with the music track can be decoded applying the technique in Clause 4.

NOTE 3 Essentially this structure reiterates the Playback model. In order for a MAF application to handle MPEG-21 it has to contain two mandatory items and a possible optional item, consisting of:

— mp4 file with MP3onMP4 audio track (item_ID = 1) [Mandatory]

— JPEG image (item_ID =2) [Optional]

— MPEG-7 meta-data (item_ID = 3) [Mandatory]

5.3 Multiple Track Album

This structure generalizes the structure defined above by including more than one hidden mp4 file. Multiple hidden mp4 files are included in the mp21's mdat box to enable complete music album files with several music tracks.

File Type: major-brand = 'mp21', compatible-brands = 'mp21', 'MMU2'.

Meta-data Handler Type: MPEG-21 (mp21) at file level

Resource Lookup and Playback:

— Even though this structure has multiple music tracks, the processing is identical to that of the structure in 5.2.

— Again the file level *meta* box contains all information to find the information pertaining to each MP3onMP4, its MPEG-7 meta-data and possibly its JPEG image. The Statement element mapping also applies to the multi track structure.

NOTE Refer to Clause 4 for internals of the hidden *mp4* files.

EXAMPLE An example DID which would be in the xml box inside the file level meta box can be found in Annex B

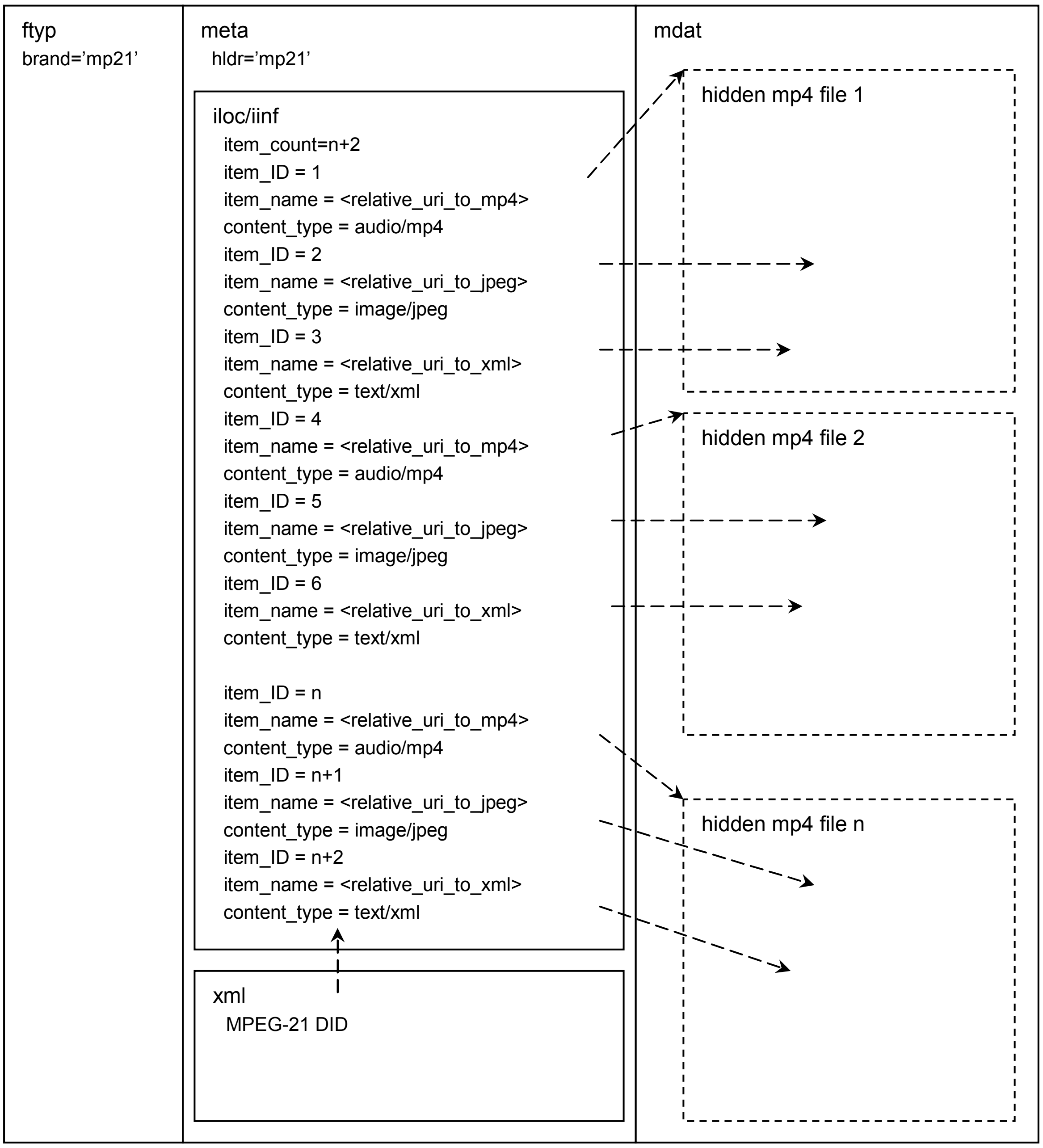
ftyp
brand='mp21'
meta
hldr='mp21'
iloc/iinf
item_count=n+2
item_ID = 1
item_name = <relative_uri_to_mp4>
content_type = audio/mp4
item_ID = 2
item_name = <relative_uri_to_jpeg>
content_type = image/jpeg
item_ID = 3
item_name = <relative_uri_to_xml>
content_type = text/xml
item_ID = 4
item_name = <relative_uri_to_mp4>
content_type = audio/mp4
item_ID = 5
item_name = <relative_uri_to_jpeg>
content_type = image/jpeg
item_ID = 6
item_name = <relative_uri_to_xml>
content_type = text/xml
item_ID = n
item_name = <relative_uri_to_mp4>
content_type = audio/mp4
item_ID = n+1
item_name = <relative_uri_to_jpeg>
content_type = image/jpeg
item_ID = n+2
item_name = <relative_uri_to_xml>
content_type = text/xml
xml
MPEG-21 DID
mdat
hidden mp4 file 1
hidden mp4 file 2
hidden mp4 file n

Figure 10 — Example of file structure for mp21 album file containing several hidden mp4 song file

5.4 Playlist

The playlist format is using the same top level structure as the album, but does not include the songs as hidden mp4 files, but references them as external files. The mp21 playlist file contains only the meta box and acts as “parent” of one or more separate mp4 “child” song files.

The dinf/dref box is used for linkage of the mp21 playlist file and the separate mp4 song files.

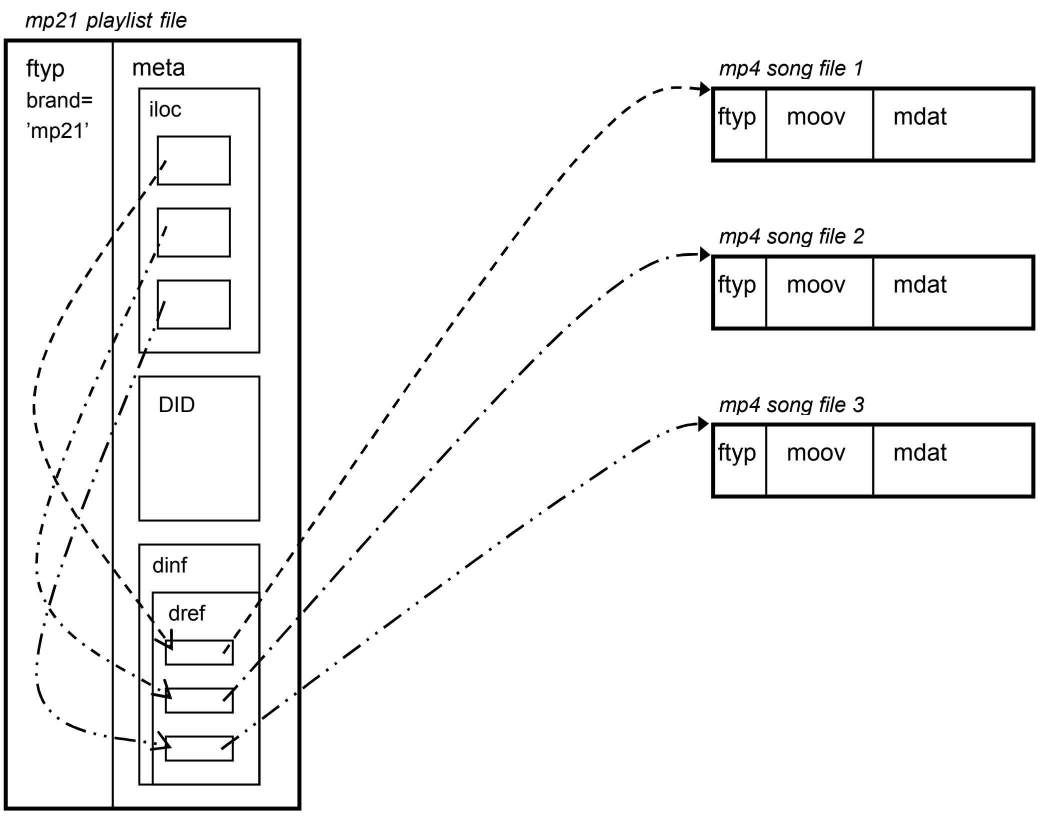

Figure 11 — Linkage of mp21 playlist file referencing external mp4 song files

The 'iloc' box contains for every item a *data_reference_index* starting with '1' (an index of '0' is used for internal hidden files as described for the album above). This index selects the entry from the 'dref' box inside the 'meta' box that corresponds to this item. The 'dref' box in turn contains a list of *DataEntryUrnBox* entries with URL and URN strings in UTF-8 form.

The following walkthrough explains how to find the mp4 song file for one entry in the "playlist" DID:

— Choose song from DID

— Use itemID of this song to select the corresponding item locator in the 'iloc' box

— Get dref-index from iloc entry for this item

— Go to 'dref' box and search for data entry with this index

— Read URL and URN from the *DataEntryUrnBox* ' urn' of the selected data entry

— Use URL to locate the mp4 song file

6 Scope of the Protected Music Player Application Format

The “Protected Music Player” described in the following Clauses 7 to 10 builds on the Music Player as described above. It adds content protection for mp4 song files, explains a default encryption for the song files and adds protection to the mp21 album and playlist files with flexible protection tool selection and key management components.

The following cases are possible in addition to the unprotected case described in Clauses 3 to 5:

a) Protected content files in mp4 file format, without Key Management components, with the default AES-128 encryption tool and MPEG-4 IPMP-X signalling in the IPMPInfoBox

b) Protected files with flexible tool selection and Key Management components (MPEG-21 IPMP and REL) using the mp21 file format with embedded mp4 content files

c) Protected mp21 file with Key Management components (MPEG-21 IPMP and REL) but without embedded mp4 content file (variation of (b)) that functions as a “license file” for an external protected mp4 content file (a)

Optional separation of protected content and license supports a broad range of "governed content scenarios" including “super distribution of protected content” and “subscription models”. The following Figure 12 illustrates the different cases and gives some examples.

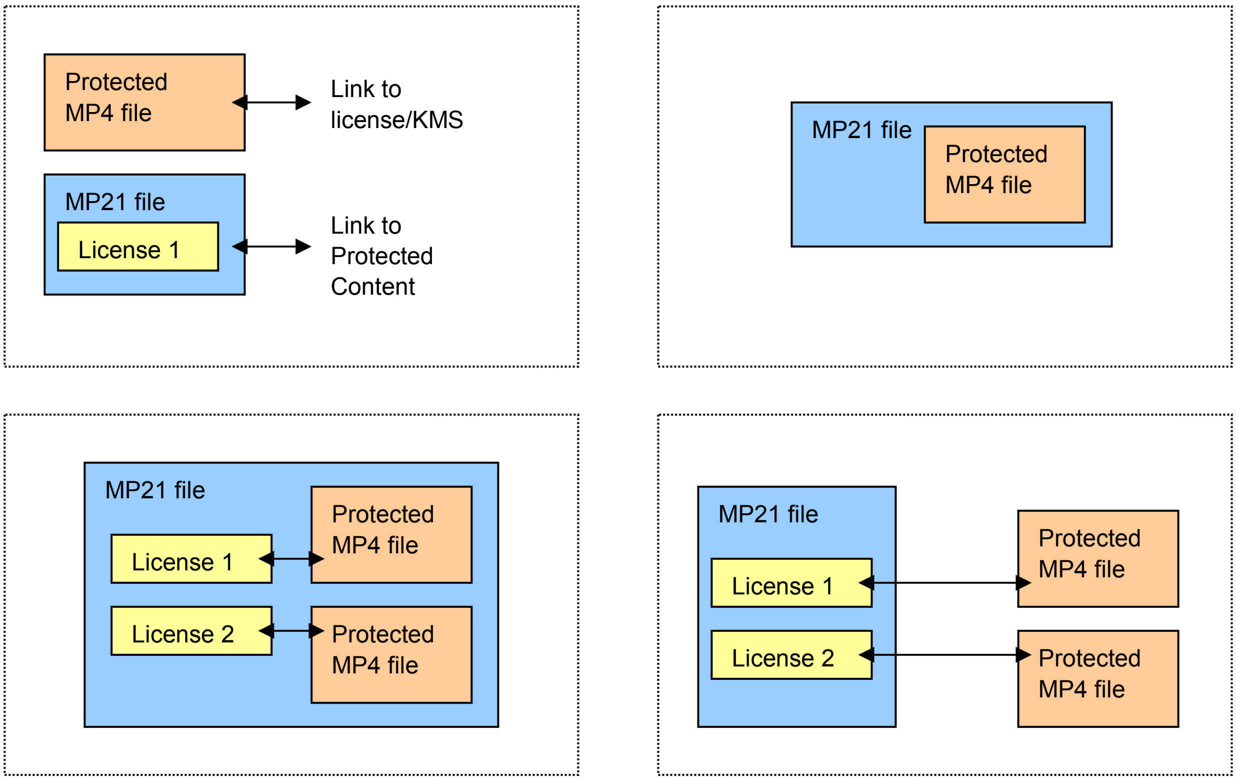

Figure 12 — Examples illustrating the different cases for the relationship of mp4 and mp21 files

7 Overview of Basic Standards for Protection

7.1 Protection and MPEG-4 File Formats

The ISO Base Media File Format ISO/IEC 14496-12:2005 offers basic box structures to signal content protection within the sample description of the *trak* box, as well as within the *meta* box. This boxes are used to signal the protection method and all other protection related parameters.

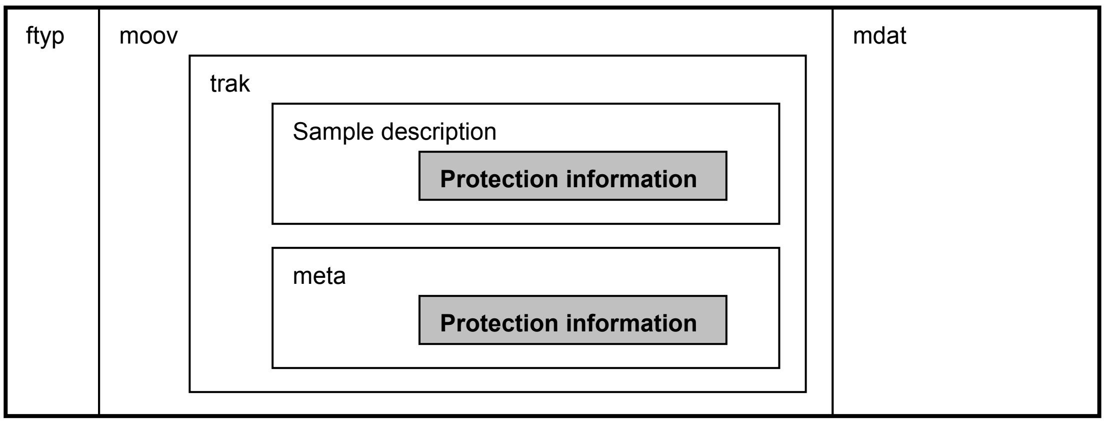

Figure 13 — Protection information inside of mp4 files

7.2 AES-128 encryption

The Advanced Encryption Standard (AES) is an open standard symmetric encryption algorithm. It is a thoroughly evaluated and tested encryption algorithm that is widely accepted as a good choice for many applications. It is the result of an open competition started by NIST to find a successor to DES. Detailed information about AES can be found on the Official AES-Website of the National Institute of Standards and Technology (NIST) [1].

Examples for the usage of AES128 are:

- AACS (Advanced Access Content System): Protection of audio-visual data on high density optical disc formats (HD-DVD, Blu-Ray)
- IPsec, SSH, IEEE 802.11i: protection of general IP (Internet Protocol) based communication

AES uses a symmetric block cipher with variable block length and a key-length of 128, 192 or 256 bit (AES-128, AES-192 and AES-256). It is easy to implement in either hardware or software and requires only low resource consumption in regards of CPU, memory and code length. It is resistant to all known methods of cryptoanalysis and worldwide royalty free.

7.3 MPEG-4 IPMP-X

ISO/IEC 14496-13 “MPEG-4 IPMP Extension” provides tool renewability, which protects MPEG-4 contents against security breakdown. The flexibility allows the use of various protection tools, as well as decoding tools. MPEG-4 IPMP Extension key elements which are used for protecting the music player format are described below.

— IPMP Tools: IPMP tools are modules that perform (one or more) IPMP functions such as authentication, decryption, watermarking, etc. A given IPMP Tool may coordinate other IPMP Tools. Each IPMP Tool has a unique IPMP Tool ID that identifies a Tool in an unambiguous way, at the presentation level or at a universal level.

- IPMP Descriptors: The IPMP Descriptors are used to denote the IPMP Tool that is used to protect the object. An independent registration authority (RA) is used so any party can register its own IPMP Tool and identify this without collisions.

- IPMP Tool List: IPMP Tool list carries the information of the tools required by the terminal to consume the content. This mechanism enables the terminal to select and manage the tools or retrieve them when the tools are missing.

7.4 MPEG-21 IPMP Base Profile

ISO/IEC 21000-4 specifies the MPEG-21 IPMP Base Profile that is aimed at supporting use cases in widespread use in the area of commercial content distribution. The Base Profile purposely provides a limited scope in order to facilitate the implementation in devices with limited computational/storage capabilities. It provides sufficient functionality to support current and emerging practices for distribution of commercial content, with a special focus on entertainment content such as movies and music, while reducing the requirements on end devices (e.g. footprint, memory usage, computational power, storage).

The MPEG-21 IPMP Base Profile adopts and in some cases restricts the MPEG-21 IPMP Components specification.

7.4.1 IPMP Digital Item Declaration Language

The MPEG-21 IPMP Base Profile includes all the elements in the IPMP DIDL schema. A set of these elements are a representation of the DID model that allows for inclusion of governance information as defined in ISO/IEC 21000-4.

7.4.2 IPMP General Info Descriptor

The IPMP General Info Descriptor in the MPEG-21 IPMP Base Profile restricts the corresponding element defined in ISO/IEC 21000-4 as follows:

- The ToolList shall contain at most one instance of ToolDescription (i.e. cardinality is 1).

- The ToolDescription element shall include the IPMPToolID element and optionally the Remote element, i.e. this Base Profile provides no support for Inline tool.

- The tool is assumed to be ready-to-be-used on the terminal; hence there is no need to carry the ConfigurationSettings element.

- The LicenseCollection element can contain any number of RightsDescriptor elements (in case there are multiple assets in the digital item), although in most usage instances a single RightsDescriptor element is likely to be used. The RightsDescriptor in the Base Profile excludes the possibility of having an IPMPInfoDescriptor child.

7.4.3 IPMP Info Descriptor

The IPMP Info Descriptor in the MPEG-21 IPMP Base Profile restricts the corresponding elements defined in ISO/IEC 21000-4 as follows:

- The Tool element shall have no attributes (since there is at most one Tool, order is no longer relevant).
- There is no support to describe the tool in the Info Descriptor. The tool will only refer to the one that has been defined in the IPMP General Info Descriptor; hence a Tool shall include a ToolRef element.
- There is no license information place holder under the IPMP Info Descriptor as all the necessary license shall be put under LicenseCollection in the IPMP General Info Descriptor.
- There is no support for hierarchical protection to the InitializationSettings element.

7.5 MPEG-21 REL MAM Profile

The MPEG-21 REL MAM Profile (Mobile And optical Media Profile) is meant to be used in simple multimedia application domains. This simple applications may sometimes but not always run on limited devices/systems. To this end, the MAM Profile restricts the MPEG-21 Rights Expression Language as specified in ISO/IEC 21000-5 which includes REL Core, REL Standard Extension, and REL Multimedia Extension.

8 Protection of mp4 files

8.1 Introduction

This clause describes signalling of content protection in mp4 files. The scope is to only signal the protection method (like encryption) and the associated parameters. License or Key Management System (KMS) related signalling (like usage rules using a rights expression language or storage and transport of the protected encryption key or any other IPMP signalling necessary for a complete DRM) are out of scope for this clause. Only linkage information to an external KMS may be signalled.

All changes that are necessary to protect the mp4 file take place within the box structure of the mp4 file. The process to hide an mp4 file within an mp21 file is not affected. A protected mp4 audio file differs in three ways from the clear text mp4 audio file:

- The sample description four-cc code is changed to “enca”.
- The “ProtectionSchemeInfoBox” is added to the sample description and contains all signalling information in sub-boxes.
- The audio samples are transformed.

8.2 Signalling

On the mp4 file format level, the following information has to be signalled:

- If the content of a track is actually protected
- Which protection scheme is used
- All parameters related to the protection scheme
- KMS related Linkage information :
  - Content ID to uniquely identify the content and associate content and license
  - KMS URI as contact information for the License Issuer (e.g. to get key and rights information)

The ISO Base Media File Format defines the basic boxes to signal content protection (ISO/IEC 14496-12:2005, Subclause 8.45). The *ProtectionSchemeInfoBox* is part of the sample description and contains all necessary information in the sub-boxes *OriginalFormatBox*, *SchemeTypeBox, SchemeInformationBox* and *IPMPInfoBox*. All other boxes of the file are unmodified.

```
aligned(8) class ProtectionSchemeInfoBox(fmt) extends Box('sinf') {
   OriginalFormatBox(fmt) original_format;
   IPMPInfoBox IPMP_descriptors;
   SchemeTypeBox scheme_type; //optional
   SchemeInformationBox scheme_type_info; //optional
}
```

This specification uses the *IPMPInfoBox*, so it is not optional in the scope of the Protected Music Player. The *IPMPInfoBox* carries an MPEG-4 IPMP-X descriptor to signal the protection method including all parameters for the chosen protection method as well as linkage information to a KMS. Annex D describes an alternative signalling of the same values using the *SchemeTypeBox* and *SchemeInformationBox*.

8.2.1 Sample Entry Type

To enable mp4 file readers to identify protected tracks, the four-cc of the sample description is replaced with a four-cc indicating protection encapsulation: “enca” is used for protected (i.e. encrypted) audio tracks. This is done to prevent accidental treatment of protected data as if it were un-protected. The four-character-code of the original un-transformed sample description is stored in the *OriginalFormatBox* “frma”. In case of MPEG-4 audio (and MP3onMP4), this is “mp4a”.

```
aligned(8) class OriginalFormatBox(codingname) extends Box ('frma') {
   unsigned int(32) data_format = codingname;  // format of decrypted,
                                                   encoded data
}
```

8.2.2 IPMPInfoBox

The *IPMPInfoBox* contains one MPEG-4 IPMP-X Descriptor (as defined in ISO/IEC 14496-13) which documents the protection applied to the track. As no full MPEG-4 Systems is used (no complete object descriptor in a separate OD track), the IPMP Descriptor is carried directly in the *IPMPInfoBox*.

The *IPMP_Descriptor* has an *IPMP_ToolID*, which identifies the required IPMP tool for protection (see also next clause). The *IPMP_Descriptor* carries MPEG-4 IPMP-X information for one or more IPMP Tool instances.

Syntax:

```
aligned (8) class IPMPInfoBox extends FullBox(‘imif’, 0, 0){
   IPMP_Descriptor   ipmp_descriptor;
}
```

Semantics:

`ipmp_descriptor` is an MPEG-4 IPMP-X descriptor as defined in ISO/IEC 14496-13

8.2.2.1 IPMP_Descriptor

The MPEG-4 IPMP-X *IPMP_Descriptor* specifies the following information related to content protection:

- Protection method (e.g. encryption using AES 128-CTR)
- Protection granularity (all data, fragment, selected access units)
- All parameters related to the protection method used

In the scope of the Protected Music Player, the IPMP-X descriptor inside the mp4 file is not used to carry Key Management System (KMS) related information, like usage rules expressed in an REL. Only a reference to a key management system may be included. If KMS related information should be included, MPEG-21 IPMP is used inside the DID in the mp21 file (see Clause 9).

Syntax:

```
class IPMP_Descriptor() extends BaseDescriptor : bit(8) tag = IPMP_DescrTag
{
    bit(8) IPMP_DescriptorID;
    unsigned int(16) IPMPS_Type;
    if (IPMP_DescriptorID == 0xFF && IPMPS_Type == 0xFFFF){
        bit(16) IPMP_DescriptorIDEx;
        bit(128) IPMP_ToolID;
        bit(8) controlPointCode;
        if (controlPointCode > 0x00)
            bit(8) sequenceCode;
        IPMP_Data_BaseClass IPMPX_data[];
    }
    else if (IPMPS_Type == 0)
        bit(8) URLString[sizeOfInstance-3];
    else
        bit(8) IPMP_data[sizeOfInstance-3];
}
```

Semantics:

`IPMP_DescriptorID`	the value of this is set to `0xFF` (escape code)
`IPMPS_Type`	IPMPS_Type is set to `0xFFFF`, to signal that an initialization setting value is needed to be carried
`IPMP_DescriptorIDEx`	contains the actual descriptor ID; the value of this is set to `0x0001`
`IPMP_ToolID`	ID of the IPMP tool used for protection, as signaled in the IPMP_ToolListDescriptor (see 8.2.3.1)
`controlPointCode`	contains the controlPointCode
`sequenceCode`	contains the sequencePointCode
`IPMPX_data`	contains all data for the IPMP tool signaled in IPMP_ToolID.

The following Figure 14 illustrates the location of the IPMP-X descriptor inside of the track's sample description.

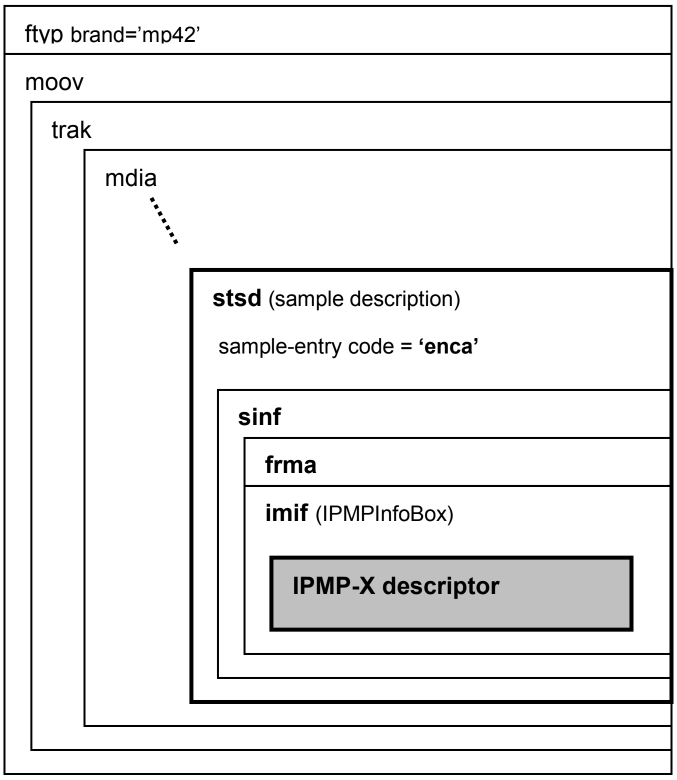

Figure 14 — Location of the IPMP-X descriptor inside the sample description

8.2.3 IPMPControlBox

IPMP Tool list information is carried in the IPMPControlBox *ipmc* at 'moov' level [4].

Syntax:

```
aligned(8) class IPMPControlBox extends FullBox('ipmc', 0, flags) {
   IPMP_ToolListDescriptor toollist;
   int(8)  no_of_IPMPDescriptors;
   IPMP_Descriptor ipmp_desc[no_of_IPMPDescriptors];
}
```

Semantics:

`toollist` is an IPMP_ToolListDescriptor

`no_of_IPMPDEscriptors` is a count of the size of the array that contains `ipmp_desc`. For usage in protected music player MAF, the value is set to 0.

`ipmp_desc[]` is an array of IPMP descriptors. For usage in protected music player MAF, no ipmp_desc is carried here.

NOTE In the scope of this specification the IPMP-X descriptor itself is carried in the sinf box (see 8.2.2) and not in the ipmc box in *ipmp_desc[]*, so the *no_of_IPMPDescriptors* is always set to zero.

8.2.3.1 IPMP_ToolListDescriptor

IPMP_ToolListDescriptor is defined in ISO/IEC 14496-13:

Syntax:

```
class IPMP_ToolListDescriptor extends BaseDescriptor :
  bit(8) tag= IPMP_ToolsListDescrTag
{
  IPMP_Tool ipmpTool[0 .. 255];
}
class IPMP_Tool extends BaseDescriptor :
  bit(8) tag= IPMP_ToolTag
{
  bit(128)  IPMP_ToolID;
  bit(1)    isAltGroup;
  bit(1)    isParametric;
  const bit(6)     reserved=0b0000.00;

  if(isAltGroup){
    bit(8)    numAlternates;
    bit(128)  specificToolID[numAlternates];
  }
  if(isParametric)
    IPMP_ParamtericDescription   toolParamDesc;
  ByteArray ToolURL[];
}
```

Semantics:

`IPMP_ToolID`	the value of this field identifies a specific protection tool.
`isAltGroup`	the value of this field is 0.
`isParametric`	the value of this field is 0.
`ToolURL`	the value of this field gives a pointer to the location of the specific protection tool

8.3 JPEG and MPEG-7 meta-data

The JPEG cover image and the MPEG-7 meta-data, both stored in the *meta* box at track level, can be protected in the same way as the audio samples. The *ProtectionSchemeInformationBox*, including all sub-boxes and descriptors as described above, may be added to the *ItemProtectionBox* “ipro” inside the *meta* box as well. The following Figure 15 illustrates the location of the IPMP-X descriptor inside of the meta box.

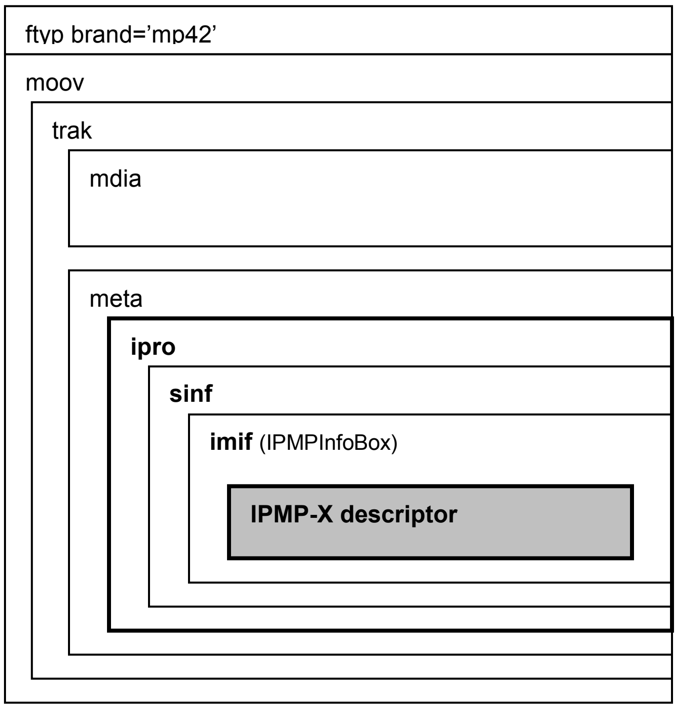

Figure 15 — Location of the IPMP-X descriptor inside the meta box

8.4 MPEG-21 DID album

In Clause 5 of the “Music Player Application Format”, the MPEG-21 File Format with an MPEG-21 DID is used for an album or playlist functionality. The MPEG-21 file may contain several mp4 files hidden in the mp21’s mdat box or references external mp4 files. The DID is used as a “table of contents” for these songs (= hidden mp4 files) of the album (= the complete mp21 file).

This functionality is not affected by the content protection. As described above, the protected mp4 file is still a valid mp4 file where the pointer from the “iloc” box of the mp21 “meta” box points to. To identify protected and unprotected mp4 files, the file reader needs to follow this pointer and parse the “moov” box of the hidden mp4 file to the sample description of the audio track and read the sample entry four-cc code. If this is “enca” the audio track is protected (i.e. encrypted), if it is “mp4a” it is clear text encoded audio data.

Protected and unprotected songs may also be mixed in one album.

9 Protection of mp21 files using MPEG-21 IPMP and MPEG-21 REL

9.1 Introduction

This clause describes signalling of protection and license information in mp21 files. The scope is to signal protection related information beyond the information signalled in the IPMP-X descriptor in the mp4 file. Also, license or Key Management System (KMS) related information can be signalled (like usage rules using a rights expression language or storage and transport of the protected encryption key or any other IPMP signalling necessary for a complete DRM).

If the embedded or associated mp4 file contains an IPMP-X descriptor for the "content protection signalling" (as described in Clause 8), it will not be removed (see Figure 16). The mp4 song files are not modified when importing them into an mp21 album. The MPEG-21 IPMP_INFO only carries the additional information.

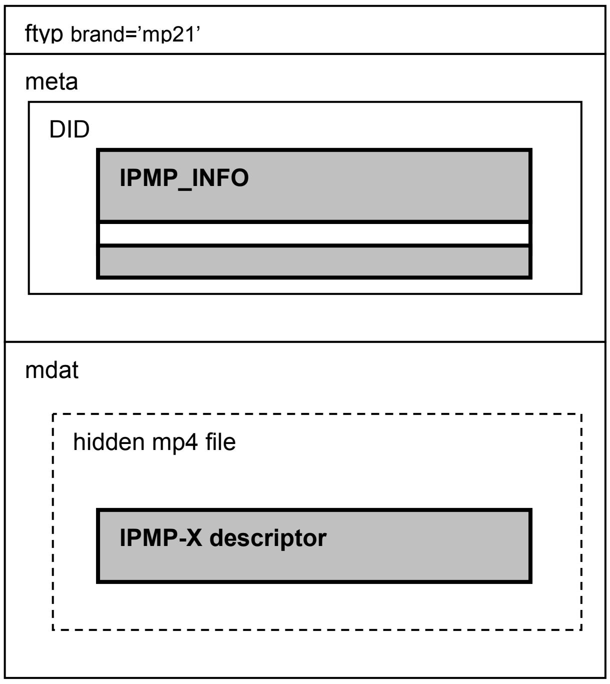

Figure 16 — The encryption info remains inside the hidden mp4 file; only add additional protection info is signalled in the MPEG-21 IPMP_INFO

This clause shows how the protection can be applied to mp21 files of the Music Player MAF:

- Single track MAF with MPEG-21 meta data and file type.
- Multiple tracks MAF with MPEG-21 meta data and file type.

The following sub clauses show the location where the description of the protection is placed. The examples of the description are listed in Annex C.

9.2 Protection of single track mp21 files

When a single hidden mp4 file is embedded in an mp21 file, the IPMP information is signalled in the form of XML metadata description (as MPEG-21 IPMP Base Profile original form). The protection description is carried in the 'meta' box at file level using MPEG-21 DID and MPEG-21 IPMP_DIDL.

Figure 17 shows an illustration of the approach. The IPMPDIDL meta data contains two major parts. Note that the structure described below is not an exhausted one, more additional DID elements may exist:

— Descriptor that contains IPMPGeneralInfo. It is recommended that this Descriptor is defined at the beginning of the IPMPDIDL meta data. The IPMPGeneralInfo contains:

  — ToolList, as defined in MPEG-21 IPMP Base Profile.

  — Container for licenses. The license information is described by MPEG-21 REL MAM Profile.

— An Item element that model the structure of the Protected Music Player MAF content. The Item shall contain at least three children Container elements. Each Container carries a Resource element for each sub resource of the Protected Music Player MAF (mp4 file with MP3onMP4 audio track, JPEG image and MPEG-7 meta data). If the sub resource is protected, the Resource element shall have an IPMPInfo element that describes the protection mechanism.

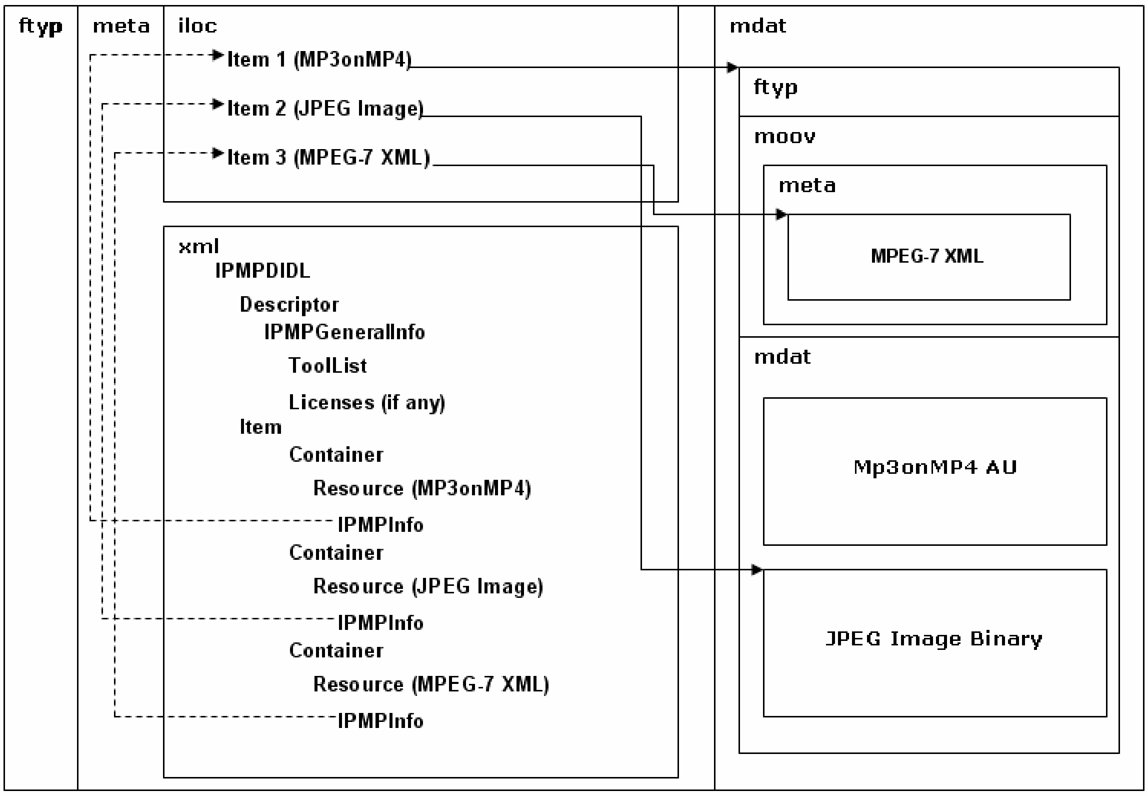

Figure 17 — Protecting single track Music MAF with IPMP support

General note: ISO/IEC 21000-4 provides more detailed information about IPMPDIDL, IPMPGeneralInfo, and IPMPInfo.

9.3 Protection of mp21 album files with multiple tracks

The protection mechanism for a multiple tracks file is similar to the case of a single track file with MPEG-21 meta data and file type. In the multiple tracks case the same approach is applied as for mp21 files with one embedded hidden mp4 file. However, the structure of the digital item in the DIDL/IPMPDIDL has one more level.

Figure 18 shows the illustration for the case of protecting a multiple tracks mp21 album file. Note that the structure of IPMPDIDL metadata now has several Item elements. Each Item element is associated with one hidden mp4 file in the 'mdat' box. The details of Item element are similar to the one described in 9.2.

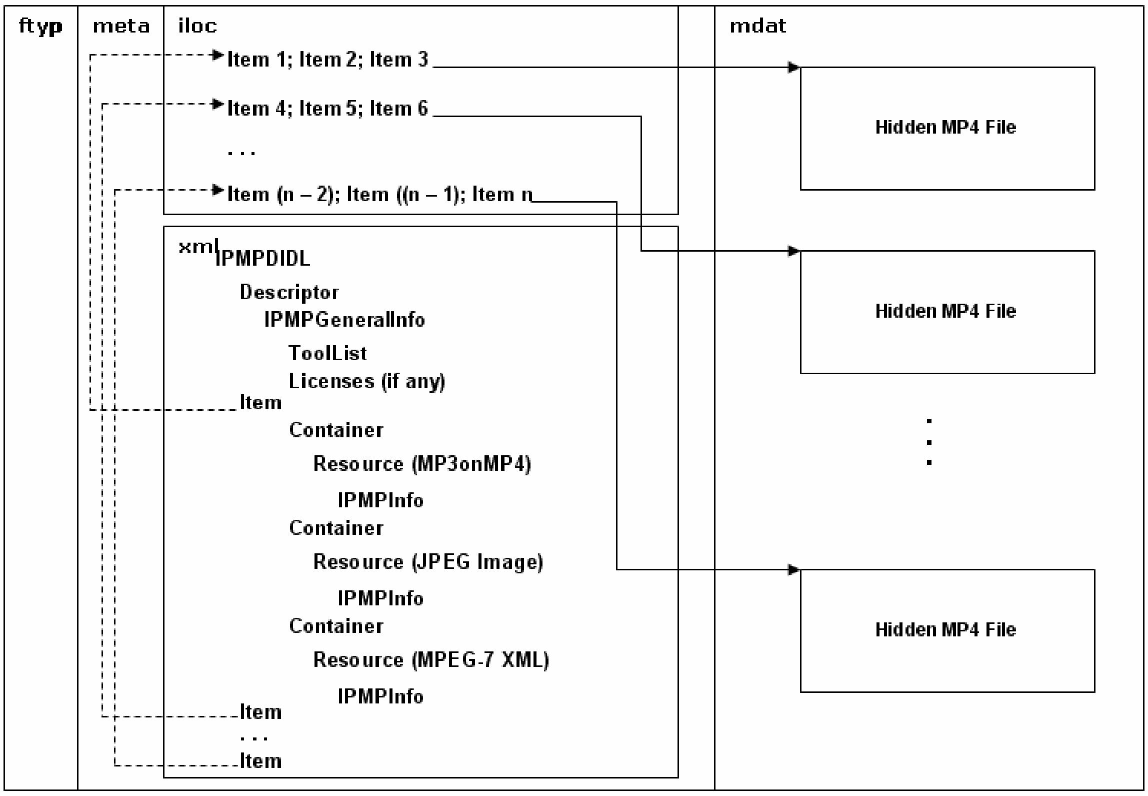

Figure 18 — Protecting multiple tracks Music MAF with IPMP support

10 Content encryption using AES-128

10.1 Introduction

This clause describes signalling and content transformation using the default encryption method AES-128. The signalling for AES-128 encryption is described using MPEG-4 IPMP-X in mp4 files.

During the encryption process, the access units are read from the ‘mdat’ box, transformed and stored back to the ‘mdat’ box. To enable continued random-access also for encrypted access units, an encryption header is added to each sample.

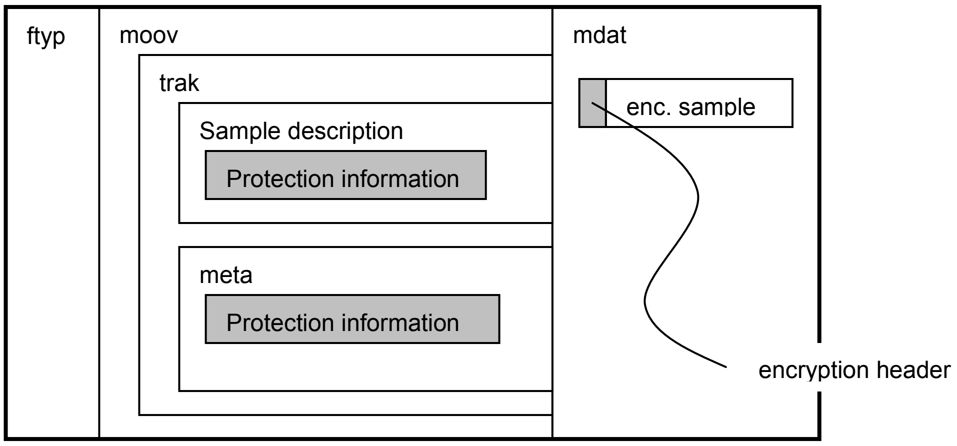

Figure 19 — Protection information inside an mp4 file and encrypted samples

10.2 MPEG-4 IPMP-X signalling of AES-128 encryption

MPEG-4 IPMP-X signaling in mp4 files is described in general in 7.3. This clause builds on this description and derives a specific IPMP-X tool and IPMP-X descriptor for AES-128 from it.

The following information has to be signaled in the IPMP-X descriptor:

- encryption method (e.g. AES-128 CTR)
- encryption granularity (all data, selected access units)
- initialization vector length
- key indicator length

10.2.1 IPMP_ToolListDescriptor for AES-128

The following Table 2 fully defines the IPMP-X tool for AES encryption. To align with the ISMACryp specification [2], the IPMP_ToolID `0x4953` is used for the AES encryption tool.

Table 2 — IPMP-X Tool definition for AES-128 encryption

Descriptor Name			
Field No.	**Size in Bits**	**Field Name**	**Value**
1	8	IPMP_ToolListDescTag	0x60
2	16	Descriptor size	0x13
		IPMP_Tool	
3	8	IPMP_ToolTag	0x61
4	8	Descriptor size	0x11
5	128	IPMP_ToolID	**0x4953**
6	1	isAltGroup	0
7	1	isParametric	0
8	6	reserved	0b0000.00

10.2.2 IPMP_Descriptor for AES-128

The following Table 3 fully defines the IPMP_descriptor to signal protection with the AES-128 decryption tool. Inside, AESCryp_Data is extended from the IPMP-X IPMP_Data_BaseClass with the DataTag `0xD0` to carry AESCryp decryption parameters.

Table 3 — IPMP-X descriptor for AES-128 encryption tool

Descriptor Name			
Field No.	**Size in Bits**	**Field Name**	**Value**
		IPMP_Descriptor	
1	8	IPMP_Descriptor tag	11
2	8	descriptor size	34
3	8	IPMP_DescriptorID	0xFF
4	16	IPMPS_Type	0xFFFF
5	16	IPMP_DescriptorIDEx	0x0001
6	128	IPMP_ToolID	**0x4953**
7	8	ControlPointCode	0x01 (between the decode buffer and the decoder)
8	8	SequenceCode	0x80
		AESCryp_Data	
9	8	AESCryp_DataTag	0xD0
10	8	data size	9
11	8	Version	0x01
12	32	dataID	**0x4953**
13	8	Crypto-suite	Identifies encryption and authentication transforms
14	8	IV-length	Byte length of the initialization vector
15	1	Selective-encryption	0 or 1 (see below)
16	7	Reserved	MUST be zero
17	8	Key-indicator-length	Byte length of the key indicator

All possible values for each parameter of AESCryp_Data and the default values are defined in Table 4.

Table 4 — parameters for IPMP-X descriptor

DESCRIPTOR	**DEFINED VALUES**	**DEFAULT**
CryptoSuite	1..255	1 (AES_CTR_128)
IVLength	1..8	4
SelectiveEncryption	0: all AUs encrypted 1: encryption of selected AUs	0
KeyIndicatorLength	0..255	0

10.3 Encrypting the audio samples

The samples (Access Units) are encrypted using AES-128 in Counter Mode (AES-128 CTR) with a 128-bit counter, key, and a 128-bit keystream block.

The 128-bit key (that is provided by the key management system) is used as input to the AES algorithm to form a so called “keystream” of pseudo-random blocks. This keystream is XOR-ed with the plain sample data for encryption. For decryption, it is XOR-ed with the encrypted data.

A unique part of the keystream is used for each audio sample. The initialisation vector IV acts as a byte offset counter to signal the start position in the key stream for decryption.

An additional salt value as random start offset into the keystream may be signalled via the KMS.

For a detailed description of the AES algorithm see [1], for a description of the process of encryption and decrypting Access Units using AES-128 CTR see also Clause 10 of [2].

10.3.1 Access Unit header

Every Access Unit (sample) is accessible independently in mp4 files. For random access, the decoding process can be started at every sync sample, which is e.g. necessary for fast forward or backward. For audio tracks, every sample is a sync sample.

Random access has to be possible also for encrypted tracks. To enable starting the decrypting process at each sample, the following encryption header is added before each sample:

Syntax:

```
aligned(8) class AESEncryptedSample {
   if (selective_encryption == 1) {// from the sample description
      bit(1)   sample_is_encrypted;
      bit(7)   reserved;      // must be zero
   }
   else sample_is_encrypted = 1;

   if (sample_is_encrypted==1) {
       unsigned int(8 * IV_length)             IV;
       unsigned int(8 * key_indicator_length)  key_indicator;

    }
    unsigned int(8)  data[];   // encrypted media data, to end of sample
```

Semantics:

-	`sample_is_encrypted`	flag that indicates if this AU is encrypted or not (only present if selective encryption is switched on)
-	`IV`	Initialisation Vector = byte counter offset; use keystream from this position on for decryption of AU
-	`Key_indicator`	in case of key rotation enabled (`key_indicator_length =! 0`) indicates which key has to be used for decrypting this AU

10.4 Encryption of JPEG and MPEG-7 meta-data

The JPEG cover image and the MPEG-7 meta-data that is stored in the *meta* box at track level can be protected in the same way as the audio samples.

The JPEG data and the MPEG-7 data inside the “xml” resp. “bxml” box is encrypted like one audio sample. As there is only one single block of data, the encryption header should be configured with selective_encryption and key rotation switched off (with key_indicator_length=0).

11 Usage of File Format Brands

The 'ftyp' box of the ISO Base Media File Format contains a list of "brands" that are used as identifiers in the file format. To enable player applications to easily identify files which are compliant to this MAF specification, specific brand identifiers are defined. These brands are used in the compatible-brands list in addition to other appropriate brand types, like "iso2", "mp42" or "mp21".

One brand is defined for every conformance point described in Clause 12. These brands are "four character codes" of the type "MM?2", with "MM" for "Music Player MAF" and "2" for "2nd edition" of the Music Player MAF.

The three defined brands are:

— "MMU2": Music Player MAF 2nd edition without protection (=unprotected), as defined in Clauses 3, 4 and 5

— "MMD2": Music Player MAF 2nd edition with default protection, as described in Clauses 8 and 10

— "MMF2": Music Player MAF 2nd edition with flexible protection, as described in Clause 9

12 Conformance

12.1 Introduction

Conformance points and set of test-sequences are defined. Conformance is specified for each conformance point in the Music Player Application Format. For each conformance point, the specific conformance procedure and criterion for each media type refers to the conformance specification for that media type.

12.2 File structures

Conformant files shall be readable by the reference software player. The structure of generated files shall be equivalent to the structure of one of the conformance files for the Music Player Application Format described in the following tables.

12.2.1 Unprotected music player

Filename	Short description
mmu2_11.mp4 mmu2_12.mp4	MPEG-4 file with MP3onMP4 audio track and track level meta data box containing MPEG-7 meta data and JPEG image
mmu2_13.m21	MPEG-21 album file containing mmp2_11.mp4 as one single hidden mp4 file
mmu2_14.m21	MPEG-21 album file containing two hidden mp4 files: mmp2_11.mp4, mmp2_12.mp4
mmu2_15.m21	MPEG-21 playlist file referencing one external mp4 file mmp2_11.mp4
mmu2_16.m21	MPEG-21 playlist file referencing two external mp4 files: mmp2_12.mp4, mmp2_12.mp4

12.2.2 Protected music player with default protection

Filename	Short description
mmd2_21.mp4 mmd2_22.mp4	MPEG-4 file containing protected audio track and meta data box using the default encryption method; the two files are the protected versions of 11 and 12.mp4
mmd2_23.m21	MPEG-21 album file containing mmp2_21.mp4 as one single hidden protected mp4 file
mmd2_24.m21	MPEG-21 album file containing two hidden mp4 files: mmp2_21.mp4 as protected file and mmp2_11.mp4 as unprotected file

12.2.3 Protected music player with flexible protection

Filename	Short description
mmf2_31.m21	MPEG21 file containing one hidden mp4 file that is protected using MPEG-21 IPMP (or IPMP/REL) (unprotected input file for protection is: 11.mp4)
mmf2_32.m21	MPEG-21 file containing mmp2_21.mp4 as hidden mp4 file. The mp4 file is protected using the default encryption (see above) and additional MPEG-21 IPMP/REL tools defined in the MPEG-21 DIDL
mmf2_33.m21	MPEG-21 file containing two hidden mp4 files that are protected using MPEG-21 IPMP/REL (unprotected input files for protection are: 11.mp4 and 12.mp4)
mmf2_34.m21	MPEG-21 file referencing external protected mp4 file mmp2_21.mp4 The MPEG-21 file contains additional protection using MPEG-21 IPMP and REL
mmf2_35.m21	MPEG-21 file referencing external mp4 file. The mp4 file is the extracted hidden mp4 from mmp2_31.m21. It is protected using MPEG-21 IPMP and REL as described in the MPEG-21 file.

12.3 Music Player device conformance points

12.3.1 Unprotected music player

Conformant devices shall be able to read the conformance files of 12.2.1. The output of the devices shall be the same as the output of the reference software player.

12.3.2 Protected music player using the default protection

Conformant devices shall be able to read the conformance files of 12.2.2. The output of the devices shall be the same as the output of the reference software player.

12.3.3 Protected music player using flexible protection

Conformant devices shall be able to read the conformance files of 12.2.3. The output of the devices shall be the same as the output of the reference software player.

13 Reference Software

13.1 Player architecture

The software architecture for the Music Player MAF application is shown in Figure 20. Basically, it consists of three parts: box parser, un-protector and resource playback. The box parser will be invoked first, parsing all necessary information stored inside the boxes, in case of protected resources, the un-protector will be used to decrypt the encrypted protected resource(s), while the resource playback will play the MP3 audio data and display its corresponding JPEG image data and MPEG-7 metadata based on the parsed information. To reduce the memory usage, the resource will be loaded to the memory only during the playback.

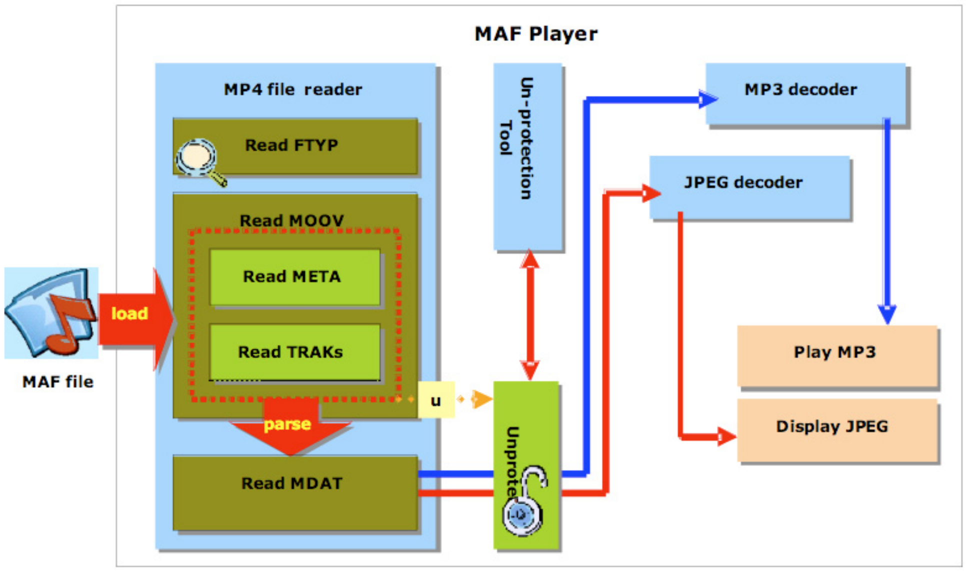

Figure 20 — Software architecture of the Music Player MAF application

The following table gives an overview of all header-files and cpp-files used for the player application. External header-files are marked in *italic*.

File name	**Usage**
fmod.h, fmod.hpp	FMOD SoundSystem header files
fmod_errors.h	FMOD SoundSystem error-handler file
xfile.h, ximabmp.h, ximacfg.h, ximadef.h, ximage.h, ximaiter.h, ximajpg.h, xiofile.h, xmemfile.h	These files contain classes for CxImage library
res/xptheme.xml	Windows XP skin for MFC
icon.ico	Icon for dialog box
MP4Movies.h, MP4OSMacros.h, ISOMovies.h	libisomedia library header files
Rijndael.cpp, XORProtectionTool.cpp, ProtectionTool.cpp	Protection tool class files
Rijndael.h, XORProtectionTool.h, ProtectionTool.h	Protection tool header files

13.2 Player functionalities

Music Player MAF files (protected and unprotected) are parsed in the following way:

1) In case of an mp21 file, parse the file-level meta box to get the primary data, i.e. the IPMP DIDL metadata

2) In case of an mp21 file, parse the file-level meta box to get the first item, i.e. the hidden MP4 file

3) Parse the (hidden) mp4 file to get the resources: MP3 audio, JPEG image and MPEG-7 metadata

13.2.1 IPMP protection

In case of mp21 files using flexible protection tools the IPMP DIDL meta-data is parsed to obtain the identifier for the protection tool used inside the file. The un-protection tool will be invoked just once when a protected item is found during the third procedure. The IPMP DIDL meta-data is parsed using MSXML library, following the schema of IPMP base profile.

This application uses several protection tools to unprotect protected resource(s): an XOR algorithm and AES algorithm (Rijndael). The protection tool is applied to each access units of the protected audio data.

13.3 MP3onMP4 Access Unit translator

The module mafmp.c currently builds to a small application that opens and read access units from an MPEG-4 file, optionally decrypts them (if they were encrypted with AES-128 CTR) and translates them from MP3onMP4 format to MPEG-2 Layer 3 format. These mp3 frames are then saved to an mp3 bitstream file. This functionality will be integrated in the player application.

Filename	**Description**
mafmp.c	MP3onMP4 bitstream decoder and AES-128 CTR decrypter
Makefile	Makefile for compilation and linking
Readme.txt	Informational file

13.4 File format

The reference software library “libisomediafile” is used for reading/parsing MPEG-4 and MPEG-21 files.

13.5 External dependencies

External libraries are used for functionalities that are beyond the core scope of this specification to be able to build a complete application.

13.5.1 MP3 and JPEG decoder

To decode the MP3 audio and JPEG image data the following external libraries are used:

— The FMOD SoundSystem library by Fireflight Technologies is used to decode the MP3 audio. This library is available as freeware (for non-commercial usage) in http://www.fmod.org.

— The CxImage library by Davide Pizzolato is used to decode JPEG data. This library is open source and can be licensed under the zlib license, available at http://www.xdp.it/cximage.htm.

13.5.2 XML Parser

To parse the XML metadata in the player, the MSXML SDK is used. It is available at http://www.microsoft.com/downloads/.

13.5.3 AES-128 CTR decrypter

The “libcrypto” from the “OpenSSL” open source project is used for AES-128 CTR decryption. It is available at http://www.openssl.org.

Annex A
(informative)

Example of ID3 information mapped to MPEG-7

ID3 V1.1	Value
Song Title	If Ever You Were Mine
Album Title	Celtic Legacy
Artist	Natalie MacMaster
Year	1995
Comment	AG# 3B8308D8
Track	05
Genre	80 (Folk)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ID3 V1.1 Example -->
<Mpeg7   xmlns="urn:mpeg:mpeg7:schema:2001"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 C:\mpeg7\is\Mpeg7-2001.xsd">

    <Description xsi:type="CreationDescriptionType">
        <CreationInformation id="track-05">
            <Creation>
                <!-- ID3 Song Title -->
                <Title type="songTitle">If Ever You Were Mine</Title>
                <!-- ID3 Album Title -->
                <Title type="albumTitle">Celtic Legacy</Title>
                <!-- ID3 Comment -->
                <Abstract>
                 <FreeTextAnnotation>AG# 3B8308D8</FreeTextAnnotation>
                </Abstract>
                <!-- ID3 Artist -->
                <Creator>
                 <Role href="urn:mpeg:mpeg7:RoleCS:2001:PERFORMER"/>
                 <Agent xsi:type="PersonType">
                    <Name>
                        <FamilyName>MacMaster</FamilyName>
                        <GivenName>Natalie</GivenName>
                    </Name>
                 </Agent>
                </Creator>
                <!-- ID3 Year -->
                <CreationCoordinates>
                 <Date><TimePoint>1995</TimePoint></Date>
                </CreationCoordinates>
            </Creation>
            <!-- ID3 Genre  (80 = Folk) -->
            <Classification>
                <Genre href=" urn:id3:cs:ID3genreCS:v1:80"><Name>Folk</Name></Genre>
            </Classification>
        </CreationInformation>
    </Description>
    <Description xsi:type="SemanticDescriptionType">
```

```
        <Semantics>
            <SemanticBase xsi:type="SemanticStateType">
                <!-- ID3 Track —>
                <AttributeValuePair>
                    <Attribute>
                        <TermUse href="urn:mpeg:maf:cs :musicplayer:CollectionElementsCS:2007:assetNum"/>
                    </Attribute>
                    <IntegerValue>6</IntegerValue>
                </AttributeValuePair>
                <!-- ID3v2 TRCK /12-->
                <AttributeValuePair>
                    <Attribute>
                        <TermUse href="urn:mpeg:maf:cs :musicplayer:CollectionElementsCS:2007:assetTot"/>
                    </Attribute>
                    <IntegerValue>12</IntegerValue>
                </AttributeValuePair>
                <!-- ID3v2 TPOS 1/2-->
                <AttributeValuePair>
                    <Attribute>
                        <TermUse href="urn:mpeg:maf:cs :musicplayer:CollectionElementsCS:2007:volumeNum"/>
                    </Attribute>
                    <IntegerValue>1</IntegerValue>
                </AttributeValuePair>
                <AttributeValuePair>
                    <Attribute>
                        <TermUse href="urn:mpeg:maf:cs :musicplayer:CollectionElementsCS:2007:volumeTot"/>
                    </Attribute>
                    <IntegerValue>2</IntegerValue>
                </AttributeValuePair>
            </SemanticBase>
        </Semantics>
    </Description>
</Mpeg7>
```

Annex B
(informative)

Example of DID in MAF

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:01-DIDL-NS"
  xmlns:Mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Item id="SOME_ALBUM">
    <Item id="track-1">
      <Descriptor>
        <Statement mimeType="text/plain">Track 1 Description</Statement>
      </Descriptor>
      <Descriptor>
        <Component>
          <Resource mimeType="image/jpeg" ref="cover_of_song1.jpg"/>
          <Component>
            <Descriptor>
              <Statement mimeType="text/plain">Track 1 Specific Description</Statement>
            </Descriptor>
            <Resource mimeType="audio/mp4" ref="song1.mp4"/>
          </Component>
          <Descriptor>
            <Statement mimeType="text/xml" ref="description_of_song1xml"/>
          </Descriptor>
        </Component>
      </Descriptor>
    </Item>
    <Item id="track-2">
      <Descriptor>
        <Statement mimeType="text/plain">Track 2 Description</Statement>
      </Descriptor>
      <Descriptor>
        <Component>
          <!--JPEG not attached-->
           <Component>
            <Descriptor>
              <Statement mimeType="text/plain">Track 2 Specific Description</Statement>
            </Descriptor>
            <Resource mimeType="audio/mp4" ref="song2.mp4"/>
          </Component>
          <Descriptor>
            <Statement mimeType="text/xml" ref="description_of_song2.xml"/>
          </Descriptor>
        </Component>
      </Descriptor>
    </Item>
   <Item id="track-3">
      <Descriptor>
        <Statement mimeType="text/plain">Track 3 Description</Statement>
      </Descriptor>
      <Descriptor>
        <Component>
          <Resource mimeType="image/jpeg" ref="samplepics/cover_of_song3.jpg"/>
          <Component>
            <Descriptor>
```

Note that this Track does not have a corresponding JPEG cover image

```
                        <Statement mimeType="text/plain">Track 3 Specific Description</Statement>
                    </Descriptor>
                    <Resource mimeType="audio/mpeg" ref="song3.mp4"/>
                </Component>
                <Descriptor>
                    <Statement mimeType="text/xml" ref="description_of_song3.xml"/>
                </Descriptor>
            </Component>
        </Descriptor>
    </Item>
  </Item>
</DIDL>
```

Annex C
(informative)

Examples of MPEG-21 Protection Metadata

Below are examples of Protected Music Player definition base on the IPMP and REL profiles.

Table C1 shows an instance of IPMP/REL description that applies protection and governance to single track music. In this example, an encryption tool "urn:IPMPToolsServer:ToolEnc005-3485" (localId = 1) is used to protect resource identified by id IPMPId001 which is the audio data. The parameter to execute the tool is carried by the InitializationSettings element. The protected data itself is stored in the 'mdat' box (see file format structure in Clause 5). The information of the protected data in the 'mdat' is shown in the ipmpdidl:Contents element [#mp (/byte(2000, 5000000))] which means the media pointer started at byte 2000 with length 5.000.000 bytes.

The license information which grant a permission to play the protected mp3 resource (id: IPMPId001) is stored under the ipmpdidl:LicenseCollection element.

Table C.1 — Example of protection to single track music player MAF

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:ipmpdidl="urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-BASE-NS"
xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-REL-
MX-NS rel-mx.xsd
urn:mpeg:mpeg21:2004:01-IPMPINFO-BASE-NS IPMPGeneralInfo-Profilev0.4.xsd
urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS IPMPDIDL.xsd
urn:mpeg:mpeg21:2002:01-DII-NS dii.xsd
urn:mpeg:mpeg21:2002:02-DIDL-NS DIDL.xsd ">
    <Container>
        <Descriptor>
            <Statement mimeType="text/xml">
                <ipmpinfo:IPMPGeneralInfoDescriptor>
                    <ipmpinfo:ToolList>
                        <ipmpinfo:ToolDescription localID=" a">

    <ipmpinfo:IPMPToolID>urn:mpegRA:mpeg21:IPMP:ABC005:77:29</ipmpinfo:IPMPToolID>
                            <ipmpinfo:Remote ref="urn:IPMPToolsServer:ToolEnc005-3485"/>
                        </ipmpinfo:ToolDescription>
                    </ipmpinfo:ToolList>
                </ipmpinfo:IPMPGeneralInfoDescriptor>
            </Statement>
        </Descriptor>
        <Item id="MySong">
            <Component>
                <Resource mimeType="image/jpeg" ref="samplemp3s/MySongCover.jpeg"/>
            </Component>
            <Component>
                <Resource mimeType="application/ipmp">
                    <ipmpdidl:ProtectedAsset mimeType="audio/mp3">
                        <ipmpdidl:Identifier>
                            <dii:Identifier>IPMPId001</dii:Identifier>
                        </ipmpdidl:Identifier>
                        <ipmpdidl:Info>
```

```
                                    <ipmpinfo:IPMPInfoDescriptor>
                                        <ipmpinfo:Tool>
                                            <ipmpinfo:ToolRef localidref="1"/>
                                            <ipmpinfo:InitializationSettings>
                                                <ipmpinfo:InitializationData>
                                                    <key_length>4</key_length>
                                                    <operating_mode>PBE</operating_mode>
                                                    <encrypt_format>Based4</encrypt_format>
                                                    <initialization_vector>12sdsrer</initialization_vector>
                                                    <padding_scheme>PCK#5</padding_scheme>
                                                </ipmpinfo:InitializationData>
                                            </ipmpinfo:InitializationSettings>
                                        </ipmpinfo:Tool>
                                    </ipmpinfo:IPMPInfoDescriptor>
                                </ipmpdidl:Info>
                                <ipmpdidl:Contents ref="="#mp (/byte(2000, 5000000))"/>
                            </ipmpdidl:ProtectedAsset>
                        </Resource>
                    </Component>
                </Item>
        </Container>
</DIDL>
```

Table C2 shows description for protection of multiple music tracks. As defined by the IPMP Base Profile, one protection tool is used to protect the resources (Song@Track1 and Song@Track2).

Licenses that grant access to play both songs are listed under the ipmpdidl:LicenseCollection element.

Table C.2 — Example of protection to multiple tracks music player MAF

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:ipmpdidl="urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-BASE-NS"
xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-REL-MX-NS rel-mx.xsd
urn:mpeg:mpeg21:2004:01-IPMPINFO-BASE-NS IPMPGeneralInfo-Profilev0.4.xsd
urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS IPMPDIDL.xsd
urn:mpeg:mpeg21:2002:01-DII-NS dii.xsd
urn:mpeg:mpeg21:2002:02-DIDL-NS DIDL.xsd ">
    <Container>
        <Descriptor>
            <Statement mimeType="text/xml">
                <ipmpinfo:IPMPGeneralInfoDescriptor>
                    <ipmpinfo:ToolList>
                        <ipmpinfo:ToolDescription localID="1">

    <ipmpinfo:IPMPToolID>urn:mpegRA:mpeg21:IPMP:XYZ005:77:29</ipmpinfo:IPMPToolID>
                        </ipmpinfo:ToolDescription>
                    </ipmpinfo:ToolList>
                    <ipmpinfo:LicenseCollection>
                        <ipmpinfo:RightsDescriptor>
                            <ipmpinfo:License>
                                <r:license>
                                    <r:grant>
                                <mx:play/>
                                <mx:diReference>
                                    <mx:identifier>IPMPId001</mx:identifier>
                                </mx:diReference>
                                <r:allConditions>
                                    <r:validityInterval>
                                        <r:notBefore>2006-01-01T00:00:00</r:notBefore>
                                        <r:notAfter>2006-12-31T12:59:59</r:notAfter>
                                    </r:validityInterval>
                                </r:allConditions>
                            </r:grant>
                            <r:grant>
                                <mx:play/>
                                <mx:diReference>
                                    <mx:identifier>IPMPId002</mx:identifier>
                                </mx:diReference>
                                <r:allConditions>
                                    <r:validityInterval>
                                        <r:notBefore>2006-01-01T00:00:00</r:notBefore>
                                        <r:notAfter>2006-12-31T12:59:59</r:notAfter>
                                    </r:validityInterval>
                                </r:allConditions>
                            </r:grant>
                        </r:license>
                    </ipmpinfo:License>
                </ipmpinfo:RightsDescriptor>
            </ipmpinfo:LicenseCollection>
        </ipmpinfo:IPMPGeneralInfoDescriptor>
    </Statement>
</Descriptor>
<Item id="Song@Track1">
    <Component>
        <Resource mimeType="image/jpeg" ref="samplemp3s/Track_1_Image.jpeg"/>
    </Component>
    <Component>
        <Resource>
            <ipmpdidl:ProtectedAsset mimeType="audio/mp3">
                <ipmpdidl:Identifier>
                    <dii:Identifier>IPMPId001</dii:Identifier>
                </ipmpdidl:Identifier>
                <ipmpdidl:Info>
                    <ipmpinfo:IPMPInfoDescriptor>
                        <ipmpinfo:Tool>
                            <ipmpinfo:ToolRef localidref="1"/>
                        </ipmpinfo:Tool>
                    </ipmpinfo:IPMPInfoDescriptor>
                </ipmpdidl:Info>
                <ipmpdidl:Contents ref="#mp (/byte(2000, 5000000))"/>
            </ipmpdidl:ProtectedAsset>
        </Resource>
    </Component>
</Item>
<Item id="Song@Track2">
    <Component>
        <Resource mimeType="image/jpeg" ref="samplemp3s/Track_2_Image.jpeg"/>
    </Component>
    <Component>
        <Resource>
            <ipmpdidl:ProtectedAsset mimeType="audio/mp3">
                <ipmpdidl:Identifier>
                    <dii:Identifier>IPMPId002</dii:Identifier>
                </ipmpdidl:Identifier>
                <ipmpdidl:Info>
```

```
                              <ipmpinfo:IPMPInfoDescriptor>
                                  <ipmpinfo:Tool>
                                      <ipmpinfo:ToolRef localidref="1"/>
                                  </ipmpinfo:Tool>
                              </ipmpinfo:IPMPInfoDescriptor>
                          </ipmpdidl:Info>
                          <ipmpdidl:Contents ref="#mp (/byte(5002000, 4500000))"/>
                      </ipmpdidl:ProtectedAsset>
                  </Resource>
              </Component>
          </Item>
      </Container>
</DIDL>
```

Table C3 shows how the integrity of the protection description is ensured by signing the description and attaching the signature information under dsig:Signature element. In this example, dsig:Signature elements under IPMPGeneralInfoDescriptor and IPMPInfoDescriptor contain the signature information of their parent elements.

Table C.3 — Example of ensuring integrity of the protection description

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"
xmlns:dii="urn:mpeg:mpeg21:2002:01-DII-NS"
xmlns:ipmpdidl="urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS"
xmlns:ipmpinfo="urn:mpeg:mpeg21:2004:01-IPMPINFO-BASE-NS"
xmlns:mx="urn:mpeg:mpeg21:2003:01-REL-MX-NS"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:mpeg:mpeg21:2003:01-REL-MX-NS rel-mx.xsd
urn:mpeg:mpeg21:2004:01-IPMPINFO-BASE-NS IPMPGeneralInfo-Profilev0.4.xsd
urn:mpeg:mpeg21:2004:01-IPMPDIDL-NS IPMPDIDL.xsd
urn:mpeg:mpeg21:2002:01-DII-NS dii.xsd
urn:mpeg:mpeg21:2002:02-DIDL-NS DIDL.xsd ">
    <Container>
        <Descriptor>
            <Statement mimeType="text/xml">
                <ipmpinfo:IPMPGeneralInfoDescriptor>
                    <ipmpinfo:ToolList>
                        <ipmpinfo:ToolDescription localID="1">

    <ipmpinfo:IPMPToolID>urn:mpegRA:mpeg21:IPMP:ZZZ005:77:29</ipmpinfo:IPMPToolID>
                        </ipmpinfo:ToolDescription>
                    </ipmpinfo:ToolList>
                    <ipmpinfo:LicenseCollection>
                        <ipmpinfo:RightsDescriptor>
                            <ipmpinfo:License>
                                <r:license>
                                    <r:grant>
                                        <mx:play/>
                                        <mx:diReference>
                                            <mx:identifier>IPMPId001</mx:identifier>
                                        </mx:diReference>
                                        <r:allConditions>
                                            <r:validityInterval>
                                                <r:notBefore>2006-01-01T00:00:00</r:notBefore>
                                                <r:notAfter>2006-12-31T12:59:59</r:notAfter>
                                            </r:validityInterval>
                                        </r:allConditions>
                                    </r:grant>
                                </r:license>
```

```
                     </ipmpinfo:License>
                  </ipmpinfo:RightsDescriptor>
               </ipmpinfo:LicenseCollection>
               <dsig:Signature Id="1">
                  <dsig:SignedInfo>
                     <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
                     <dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
                     <dsig:Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
                        <dsig:Transforms>
                           <dsig:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
                        </dsig:Transforms>
                        <dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                        <dsig:DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</dsig:DigestValue>
                     </dsig:Reference>
                  </dsig:SignedInfo>
                  <dsig:SignatureValue>MC0CFFrVLtRlk=...</dsig:SignatureValue>
               </dsig:Signature>
            </ipmpinfo:IPMPGeneralInfoDescriptor>
         </Statement>
      </Descriptor>
      <Item id="MySong">
         <Component>
            <Resource mimeType="image/jpeg" ref="samplemp3s/MySongCover.jpeg"/>
         </Component>
         <Component>
            <Resource>
               <ipmpdidl:ProtectedAsset mimeType="audio/mp3">
                  <ipmpdidl:Identifier>
                     <dii:Identifier>IPMPId001</dii:Identifier>
                  </ipmpdidl:Identifier>
                  <ipmpdidl:Info>
                     <ipmpinfo:IPMPInfoDescriptor>
                        <ipmpinfo:Tool>
                           <ipmpinfo:ToolRef localidref="1"/>
                        </ipmpinfo:Tool>
                        <dsig:Signature Id="2">
                           <dsig:SignedInfo>
                              <dsig:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-
xml-c14n-20010315"/>
                              <dsig:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
                              <dsig:Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-
20000126/">
                                 <dsig:Transforms>
                                    <dsig:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-
c14n-20010315"/>
                                 </dsig:Transforms>
                                 <dsig:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                                 <dsig:DigestValue>rqfdsvuiojbew9$#$fda4321hjjl</dsig:DigestValue>
                              </dsig:Reference>
                           </dsig:SignedInfo>
                           <dsig:SignatureValue>fdafdasfdasrrcvarew2432dfdaf</dsig:SignatureValue>
                        </dsig:Signature>
                     </ipmpinfo:IPMPInfoDescriptor>
                  </ipmpdidl:Info>
                  <ipmpdidl:Contents ref="#mp (/byte(2000, 5000000))"/>
               </ipmpdidl:ProtectedAsset>
            </Resource>
         </Component>
      </Item>
   </Container>
</DIDL>
```

Annex D
(informative)

Alternative protection signalling in mp4 files

D.1 Introduction

This clause describes an alternative signalling of the encryption parameters using the *SchemeTypeBox* according to the ISMACryp specification [2]. This signalling is a complete equivalent of the parameters used in the IPMP-X descriptor as defined in Clauses 5 and 7. The optional *SchemeTypeBox* may be included in parallel to the *IPMPInfoBox* in 'sinf', so that both signalling schemes can coexist in one file.

Subsequently, a way to link to external Key Management Systems (e.g. OMA DRM 2.0 [6] or MPEG-21 based) within this signalling scheme is described.

D.2 Interoperability with ISMACryp

D.2.1 Encryption Scheme

The *SchemeTypeBox* identifies the protection scheme by type and version. A file reader may get additional information about the scheme using the optional "Scheme_URI". This is an absolute URI formed as a null-terminated string in UTF-8 characters.

```
aligned(8) class SchemeTypeBox extends FullBox('schm', 0, flags) {
   unsigned int(32) scheme_type; // 4CC identifying the scheme
   unsigned int(32) scheme_version; // scheme version
   if (flags & 0x000001) {
      unsigned int(8) scheme_uri[]; // browser uri
   }
}
```

The "Scheme_type" is a four-cc code that defines the protection scheme. To align with ISMACryp, "iAEC" is used as scheme with a "Scheme_version" of "1".

The *Scheme Information Box* is a container box for information the encryption system needs and is completely owned by the scheme signalled in "Scheme_type". For "iAEC" it contains two kinds of information: information necessary for the decrypting process and information to connect the protected content to a specific Key Management System.

```
aligned(8) class SchemeInformationBox extends Box('schi') {
   Box ISMAKMSBox [];
   Box ISMASampleFormatBox [];
   Box ISMACrypSaltBoxBox [];  // optional
   Box data []; // any other boxes
}
```

The *ISMASampleFormatBox* and the *ISMAKMSBox* are described in detail below. The optional *ISMACrypSaltBox* may be used to additionally convey a salt key to be used in encryption.

```
aligned(8) class ISMACrypSaltBox extends Box('iSLT') {
   unsigned int (64)
}
```

D.2.1.1 Sample Format Information

The encryption scheme supports selective encryption (some samples are stored as clear text to allow e.g. pre-listening) and random access to every sample. To allow this functionality, an encryption header is added to each sample (see Clause 10). To minimize the amount of additional bits, the encryption header is configurable using the ISMASampleFormatBox “iSFM”. In this box the selective-encryption indicator is switched on or off and the size of the key indicator and the initialization vector (IV) are configured.

```
aligned(8) class ISMASampleFormatBox extends FullBox('iSFM', 0, 0) {
   bit(1)   selective_encryption;
   bit(7)   reserved;                // MUST be zero
   unsinged int(8)   key_indicator_length;
   unsigned int(8)   IV_length;
}
```

If selective_encryption is zero, all samples of the track are encrypted and the “sample_is_encrypted” flag is not present in the encryption_header.

If key_indicator_length is zero, only one key is used for the track and no key_indicator field is present in the encryption header.

D.2.1.2 Key Management System Information

Some information is necessary to tell the mp4 file reader how to use the protected content, e.g. where to get the encryption key and the rights description. This information is about how to identify and connect to a specific Key Management System. The actual rights description, key transport, etc. is out of scope.

The KMS_URI contains an identifier where a file reader may get additional information how to connect the KMS. The KMS_ID and KMS_version are used to identify the KMS.

```
aligned(8) class ISMAKMSBox extends FullBox('iKMS', version, 0) {
   if (version==0) {
      string   KMS_URI;       // the KMS URI
   } else { // version ==1
      unsigned int(32)  KMS_ID;      // 4CC identifying the KMS
      unsigned int(32)  KMS_version; // KMS version
      string            kms_URI;     // the KMS URI
}
```

Within the scheme information box ('schi'), additional boxes may be added by the KMS. Box types starting with the letter 'k' are defined by and reserved to the KMS identified by the KMS_URI resp. the KMS_ID.

D.3 Linkage to external Key Management Systems

A variety of Key Management Systems can be used with the protected content format building block from very simple systems with potentially limited security (e.g. you get the key after registration on a Web Site) up to full fledged DRM systems.

D.3.1 OMA DRM 2.0

As described above, the *SchemeInformationBox* is a container box used to carry encryption scheme and KMS specific information. In addition to the two boxes *ISMAKMSBox* and *ISMASampleFormatBox*, it may contain any other boxes of any type and format used by a specific KMS, thus it can include OMA specific boxes.

To use OMA DRM KMS [6], the *SchemeInformationBox* includes exactly (see also ISMACryp1.1 Annex E [2]):

— the *ISMAKMSBox* "iKMS" with

  — version = 1

  — KMS ID = iOMA

  — KMS version = 0x00000200

  — KMS URI = OMA DRM v2 right issuer URI

— the OMA DRM Common Headers Box "ohdr" [5] that specifies the encryption scheme and its parameters and provides information about the Rights Issuer as well

— the *ISMASampleFormatBox* "iSFM".

With this information the OMA DRM v2.0 Right Object Acquisition Protocol (ROAP) [6] can be launched to acquire the license that includes the content encryption key and the rights information.

D.3.2 MPEG-21

To use a MPEG-21 based DRM KMS, the *SchemeInformationBox* includes:

— the *ISMAKMSBox* "iKMS" with

  — version = 1

  — KMS ID = iM21

  — KMS version = 0x00000001

  — KMS URI = MPEG-21 license URI

— if *the* MPEG-21 IPMP and REL information is included in the top level meta box in the same file, a box should be added here that contains information to direct the file reader to the meta box

— *the ISMASampleFormatBox* "iSFM".

The linkage in the other direction from MPEG-21 IPMP to the content format is out of scope here.

Annex E
(informative)

Music Player Device Behaviour

E.1 Introduction

This clause briefly describes all steps a music player application has to perform to successfully play a file. The description is given in the form of walkthroughs.

E.2 Unprotected music player

Music players need the following functionality:

- For MPEG-4 song files:
  - read from mp4 files:
    - MP3onMP4 access units from audio track
    - MPEG-7 meta data from track level meta box
    - JPEG image
  - process MP3onMP4 access units reformatting them to mp3 frames and decode mp3 frames (or do both in one step)
  - parse and decode MPEG-7 meta-data
  - decode JPEG image
- For MPEG-21 album and playlist files, additionally:
  - Read from mp21 files:
    - MPEG-21 DID
    - iloc/iinf information
    - MPEG-7 meta-data and JPEG image
  - Parse MPEG-21 DID
  - For album files:
    - Parse iloc/iinf boxes to locate included resources
    - Process hidden mp4 file like stand-alone mp4 file as described above
    - Process MPEG-7 meta-data and JPEG image

  - For playlist files:
    - Parse iloc/iinf and dinf/dref boxes to locate external resources
    - Process external mp4 file as described above
    - Process MPEG-7 meta-data and JPEG image

E.3 Protected music player using the default protection

Music players using the default protection need the following functionality:

- For MPEG-4 song files:
  - read from mp4 files:
    - protected MP3onMP4 access units from audio track
    - protected MPEG-7 meta data and JPEG from track level meta box
    - MPEG-4 IPMP-X encryption signalisation in SampleDescription
  - Parse encryption signalisation
  - decrypt audio access units
  - decrypt MPEG-7 meta-data and JPEG image
  - process MP3onMP4 access units, MPEG-7 meta-data and JPEG image as described above for the unprotected case
- For MPEG-21 album and playlist files, additionally:
  - As described above

E.4 Protected music player using flexible protection

Music players using flexible protection need the following functionality:

- For MPEG-21 files:
  - Read from mp21 files:
    - MPEG-21 DIDL, IPMP and REL
    - Hidden mp4 file using the iloc/iinf information, if resources are internal, else parse dref box for URL to external mp4 file
    - MPEG-7 meta-data and JPEG image
  - Parse MPEG-21 DIDL
    - Read IPMP info related to protection tool information

- Process hidden/referenced MPEG-4 song files:
  - read from mp4 files:
    - protected MP3onMP4 access units from audio track
    - protected MPEG-7 meta data and JPEG from track level meta box
  - de-protect audio access units
  - de-protect MPEG-7 meta-data and JPEG image
  - process MP3onMP4 access units, MPEG-7 meta-data and JPEG image as described above for the unprotected case

Bibliography

[1] NIST, FIPS 197, Advanced Encryption Standard (AES), http://csrc.nist.gov/CryptoToolkit/Encryption.html#aAES

[2] ISMA Encryption and Authentication Specification 1.1, July 2006 http://www.isma.tv/specreq.nsf/SpecRequest

[3] http://www.id3.org/id3v1.html

[4] http://www.id3.org/id3v2.3.0

[5] OMA DRM Content Format Version 2.0, OMA-TS-DRM-DCF-V2_0-20060303-A, March 2006 http://www.openmobilealliance.org/release_program/drm_v2_0.html

[6] OMA DRM Specification Version 2.0, OMA-TS-DRM-DRM-V2_0-20060303-A, March 2006 http://www.openmobilealliance.org/release_program/drm_v2_0.html

ICS 35.040

Price based on 46 pages