FINAL
DRAFT

AMENDMENT

ISO/IEC 15444-8:2007 FDAM 1

ISO/IEC JTC **1**

Secretariat: **ANSI**

Voting begins on:
2008-02-29

Voting terminates on:
2008-04-29

Information technology — JPEG 2000 image coding system: Secure JPEG 2000

AMENDMENT 1: File format security

Technologies de l'information — Système de codage d'images JPEG 2000: JPEG 2000 sécurisé

AMENDEMENT 1: Sécurité de format de fichier

Please see the administrative notes on page iii

RECIPIENTS OF THIS DRAFT ARE INVITED TO SUBMIT, WITH THEIR COMMENTS, NOTIFICATION OF ANY RELEVANT PATENT RIGHTS OF WHICH THEY ARE AWARE AND TO PROVIDE SUPPORTING DOCUMENTATION.

IN ADDITION TO THEIR EVALUATION AS BEING ACCEPTABLE FOR INDUSTRIAL, TECHNOLOGICAL, COMMERCIAL AND USER PURPOSES, DRAFT INTERNATIONAL STANDARDS MAY ON OCCASION HAVE TO BE CONSIDERED IN THE LIGHT OF THEIR POTENTIAL TO BECOME STANDARDS TO WHICH REFERENCE MAY BE MADE IN NATIONAL REGULATIONS.

Reference number
ISO/IEC 15444-8:2007/FDAM 1:2008(E)

© ISO/IEC 2008

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

Copyright notice

This ISO document is a Draft International Standard and is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

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

Violators may be prosecuted.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

© ISO/IEC 2008 – All rights reserved

In accordance with the provisions of Council Resolution 21/1986, this document is **circulated in the English language only**.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Foreword

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

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

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 15444-8:2007 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information* in collaboration with ITU-T. The identical text is published as ITU-T Rec. T.807 (05/2006)/FDAM 1(E).

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

INTERNATIONAL STANDARD
ISO/IEC 15444-8:2007/FDAM 1 (E)
ITU-T Rec. T.807 (05/2006)/FDAM 1(E)
ITU-T RECOMMENDATION

Information technology – JPEG 2000 image coding system: Secure JPEG 2000

AMENDMENT 1: File format security

1) Clause 2: Normative References

Add the following references:

ISO/IEC 13818-11:2004, *Information technology – Generic coding of moving pictures and associated audio information – Part 11: IPMP on MPEG-2 systems.*

ISO/IEC 15444-4, *Information technology – JPEG 2000 image coding system: Conformance testing*

ISO/IEC 15444-6, *Information technology – JPEG 2000 image coding system: Compound image file format*

ISO/IEC 15444-12, *Information technology – JPEG 2000 image coding system – Part 12: ISO base media file format (technically identical to ISO/IEC 14496-12)*

2) Clause 3: Terms and Definitions

Rewrite the first paragraph as follows (with the changes underlined):

For the purposes of this Recommendation | International Standard, the following definitions apply. The definitions defined in ITU-T Rec. T.800 (08/2002) | ISO/IEC 15444 -1:2004 Clause 3 and ISO/IEC 15444-12:2005 Clause 3 apply to this Recommendation | International Standard.

Add the following terms and definitions:

Normal decoder
Standard decoder is a process to decode a codestream that is fully compliant with the normative part of coding standard. Its behavior is not defined if it tries to decode a non-compliant codestream.

Adaptive-format decoder
Adaptive-format decoder is a process to decode a codestream which is not fully compliant with the normative part of the coding standard. It shall reconstruct the media (possibly with low quality or resolution) even if the codestream has missing packets or inconsistent packet headers. For example, a Adaptive-format decoder is able to understand a simply-transcoded codestream, such as one that has its highest resolution packets removed.

Elementary Stream (ES)
Elementary Streaming contains a sequence of samples, where each sample could be a video frame or a contiguous section of audio data. A sample in ES contains media data, ByteData structure, Pointer structure, Container Structure, or any mixture of the above.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Self-Contained ES

Self-Contained ES contains only media data, whose format is not defined in this amendment. The Self-Contained ES could be stored in MDAT box co-located with the File Format specified in this amendment, or be stored in a separate file whose format is not specified by this amendent.

Composed ES

Composed ES may contain a mixture of ByteData, Pointer and Container structures, that is, its samples are composed with data from other Elementary Streams. A Composed ES can either copy (using ByteData structure) or reference (using Pointer) data from other ESes.

Scalable Composed ES

Scalable Composed ES is made up of samples that may not be decodable by themselves. It may need to be combined with other Scalable Composed ESes to form a fully decodable codestream. Scalable Composed ES is designed to support scalability, i.e., to make media data “thinable”. For example, for a motion JPEG 2000 codestream where each picture has three layers, it can be divided into 3 Scalable Composed ESes: the first one consists of all layer 0 data, the second one consists of all layer 1 data and the third one consists of all layer 2 data.

Decodable Composed ES

Decodable Composed ES is made up of samples that are decodable by themselves. It is designed for simple adaptation where the adaptor just needs to retrieve data pointed by Pointer structure and remove the wrapper to form a fully scalable codestream. For example, for a motion JPEG 2000 codestream where each picture has three layers, it can form 3 Decodable Composed ESes: the first one consists of layer 0 data, the second one consists of layer 0 and layer 1 data and the third one consists of layer 0, 1 and 2 data.

Adaptor / transcoder

Adaptor / transcoder is a process to transform media data to lower scalability level, like lower resolution or lower quality or bit-rate, by removing portions of the file. The adaptor / transcoder can transform media data based on the information specified in this amendment. A adaptor / transcoder shall update byte offset values in file format parameters that are impacted by the process.

Secure adaptor / transcoder

Secure adaptor / transcoder is a process to transform encrypted or authenticated media data without necessity to decrypt or re-generate the MAC or signature. Thus, end-to-end security remains for the transcoded media data.

JPEG 2000-aware adaptor / transcoder

JPEG 2000-aware adaptor / transcoder combines one or more Scalable Composed ESes to form a fully decodable media codstream. It should have the capability to generate the headers and markers of media codestream and modify the packet index, such that the adapted codestream can be decoded by a normal decoder. It may also add empty packets to replace the removed ones, or it may insert POC marker.

Simple adaptor / transcoder

Simple adaptor / transcoder is able to transform data based on information specified by this amendment. It may not be capable of generating media headers or modifying packet indices. It simply retrieves data pointed by Pointer structure and removes the wrappers, and the resulting codestream can be decoded by adaptive-format decoder, which can cope with missing packets and inconsistent headers.

Authentication adaptor / transcoder

An authentication adaptor / transcoder removes data that is not verifiable with the available media data and authentication data. For example, in a streaming system, some media packets may be lost during transmission. A file format receiver may reconstruct the received data to the best of its ability based on the available data. Then, an authentication adaptor / transcoder can determine which data can be verified, and then remove the packets that are not verified. The resulting file only contains the decodable, verified data.

Container

Container structure is used to wrap a sample in a Composed ES. It might contain any number of ByteData or Pointer structures, but is not allowed to contain another Container structure.

Pointer

Pointer structure is used to reference a data segment in another ES. It must be contained inside a Container structure.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ByteData

ByteData structure is used to wrap a data segment which is physically located in a composed ES. It must be contained inside a Container structure.

4CC Code

4CC code is a 32-bit identifier, normally 4 printable characters. A 4CC code can be used to indicate the file type, the type of file format box, type of a file format track, type of a file format sample description and type of file format track reference. A 4CC code must be registered with a registration authority.

3) Annex E: File Format Security

Create a new Annex and add the following text:

Annex E

File Format Security
(This annex forms an integral part of this Recommendation | International Standard)

E.1 Scope

This annex specifies JPSEC file format derived from ISO base file format and modifications to JPEG family file format (including JP2, JPX and JPM) for protection and secure adaptation of scalable pictures, which is possibly encrypted and/or authenticated by the owner. The pictures could be either static pictures or time-sequenced pictures. In particular, the annex provides functionality to do the following:

- To store coded media data corresponding to different scalability levels. Elementary Stream (ES) is used for this purpose. There are three types of ESes, Self-contained ES, Scalable Composed ES and Decodable Composed ES.
- To define tracks describing the characteristics of the coded media data stored in ES. For example, the track should be able to indicate scalability level (resolution, layer, region, etc) and the Rate-Distortion hints of the coded media data in order to facilitate easy and secure adaptation.
- To define new file format boxes to signal protections tools and parameters applied to coded media data or metadata. The protection tools can be applied to either static JPEG 2000 pictures or a time-sequenced JPEG 2000 pictures.
- The protection tools defined in this amendment can be applied to JPEG family file formats including JP2, JPX and JPM and ISO-derived file formats such MJ2 for motion JPEG.

E.2 Introduction

E.2.1 Security Protection at File Format Level

This annex describes a JPSEC file format derived from ISO base file format and modifications to JPEG family file format, to add security protection to JPEG 2000 pictures at file format level. The protection applied at the file format level can be classified into two types: item-based protection and sample-based protection, both structures are defined by the ISO base file format. The item-based protection is designed to protect any byte ranges (including coded media data and metadata) while the sample-based protection is designed to protect time-sequenced media including JPEG 2000 pictures.

In case that the security tools applied change data length, it shall update all pointers and length fields in all boxes, to ensure correct parsing by the reader.

E.2.2 Item-Based Protection

This annex describes two item-based protection schemes in the ISO base file format, by leveraging the syntax and structures specified by the JPSEC standard. Specifically, it describes schemes for decryption and authentication. Each item in the ItemLocationBox is protected by one or more protection schemes in the ItemProtectionBox. When multiple schemes are used (or chained together), the order in which they are applied may be significant and thus must be specified. This annex also specifies how such operations should be chained together. In addition, we also add ItemDescriptionBox and ItemCorrespondingBox into the ISO base file format to allow the flexible processing properties that are provided by JPSEC. Specifically, the ItemDescriptionBox allows media-dependent metadata (such as resolution, quality layer, spatial region, and color space component) to be associated with different portions of the file. These

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

descriptions can be provided regardless of whether protection is applied. When used with scalable coded pictures, this allows the file to be scaled down or transcoded without parsing or decoding the media data. In cases where protection is applied, this provides the benefit of enabling transcoding without requiring decryption.

E.2.3 Sample-based Protection of Scalable Media

For time-sequenced pictures, the annex adds syntaxes to facilitate scalability at file format level, including Scalable Composed Elementary Stream (ES), Decodable Composed ES, Pointer structure, Container structure and ByteData Structure. The scalable coded pictures can be divided (either physically or virtually) into elementary streams at different scalability level, such that the adaptor / transcoder can "thin" media data with low complexity.

Figure E.1 gives an overview of the File Format specified by this annex and also shows how the specified FF is used to adapt the media data.

Given a sequence of JPEG 2000 pictures (also referred to as *Self-contained ES*), there are two approaches to construct the File Format. In the first approach, the MDAT box contains one or more *Scalable Composed ESes*, each of which corresponds to one scalability level of the media data, e.g., a resolution or a layer. The Scalable Compose ES must be stored in MDAT box that is co-located with the File Format. The Self-contained ES can be located in either MDAT box in the same file, or a different file whose format is not specified in this amendment. The Scalable Composed ES may not be decodable by itself, it may need to be combined with other scalable Composed ES to generate fully decodable JPEG 2000 pictures. In the second approach, the MDAT box contains one or more *Decodable Composed ESes*, and each ES constitutes fully decodable JPEG 2000 pictures by itself. Similarly, Decodable Composed ES must be stored in MDAT box co-located with the File Format, and the self-contained ES can be stored in either MDAT box in the same file, or a different file whose format is not specified by this annex.

Each Scalable Composed ES or Decodable Composed ES must be described by at least one track. The characteristics of the ES (like resolution, layer, and region) are indicated in SampleEntryBox inside each track.

To generate a fully decodable JPEG 2000 codestream from Scalable Compose ESes, a JPEG 2000-Aware Adaptor should have capability to dynamically generate the image headers (based on the number of resolutions, layers and region in the adapted codestream), insert empty packet or insert POC marker to make the resulting codestream decodable to any normal decodable. However, if a Simple Adaptor is used, the resulting codestream may have inconsistent image header and there may be missing packet, which require a JPEG 2000 adaptive-format decoder.

As Decodable Composed ES is decodable by itself, a Simple Adaptor is sufficient to generate fully compliant JPEG 2000 pictures.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

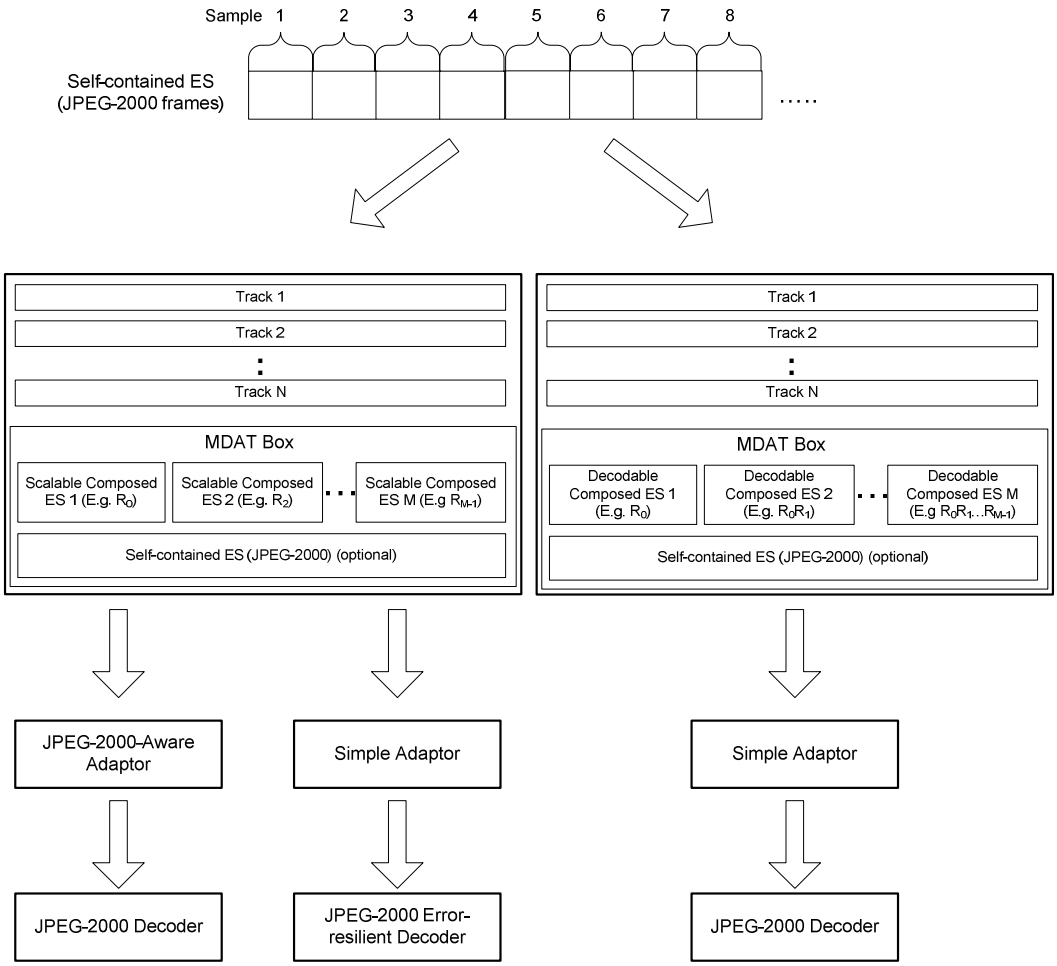


Figure E.1 – System diagram for time-sequenced scalable media

Each Elementary Stream is described by at least one media track, and its characteristics are described in SampleDescriptionEntry or SampleGroupEntry Box within the track. It is possible that a single Elementary Stream is described by multiple tracks, each of which may describe different aspects of the elementary stream.

The sample-based protection can be applied to all samples or a group of samples in a scalable composed ES or decodable composed ES. If protection is applied to all samples, a ProtectionSchemeInfoBox signaling the parameters of the protection tool is added to the SampleDescriptionBox, which is then encapsulated as described in Section E.5.1. In addition, if protection is applied to a group of samples, a ProtectionSchemeInfoBox is added to their SampleGroupEntryBox, which is then encapsulated as described in Section E.5.3.

E.3 Extension to ISO Base Media File Format

E.3.1 Overview

This clause documents technical extensions (additional box types) to the ISO Based Media File Format, which could be used for protection, adaptation, or secure adaptation of scalable coded pictures. However, the added box types could be used for other purpose as well. In particular, this section defines ProtectionSchemeInfoBox for the decryption tool and authentication tool, ItemDescriptionBox, ScalableSampleDescriptionEntry, ScalableSampleGroupEntry, and Generic Protected Box. All other boxes defined in ISO/IEC 15444-12 are still used as it is.

E.3.2 Incorporate JPSEC Codestream into ISO-driven File Format

A JPSEC codestream can be placed as a payload in the 'mdat' box of ISO base file format. In the Sample Description Box ('stsd'), the 'codingname' of the corresponding Sample Entry is defined to be 'jpsc', which is a registered identifier for JPSEC decoder. In this case, the security service is provided by JPSEC at codestream.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

E.3.3 Protected File Format Brand

Files conforming to this specification may use ‘ffsc’ as the major brand in the File Type Compatibility Box.

Files conforming to this specification i.e. containing protection or authentication information may use ‘ffsc’ as a compatible_brand in the File Type Compatibiliity Box.

There are uses of this specification which are compatible with JP2, JPX, MJ2, and JPM files. A typical use of this specification will leave the major brand of a file unchanged, but add boxes and thus ad ‘ffsc’ as a compatible brand. Thus brands including *'isom'* , *'iso2', ‘jp2\040, ‘jpx\040’* and *‘jpm\040’* should be compatible

The *'ffsc'* compatible brand indicates use of new boxes and new tools corresponding to the protection methods in JPSEC.

A file that has been protected, to the extent that an application intending to process the JP2, JPX, JPM, or other filetypecontent will be unable to do so without using protection tools, may use the 'ffsc' major brand as the file type, such a protected file must not use a major_brand for which it is no longer conformant.

E.3.4 Summary of Boxes used

The ISO Base Media File Format defines two structures to describe a presentation: the logical structure and media sequence structure. The logical structure uses the ItemLocationBox ("iloc") to describe an item which is the byte range or a series of byte ranges for a particular file, either a local file or a remote file. The media sequence structure uses the SampleGroupDescriptionBox (“spgd”) or SampleDescriptionBox ("stsd") to describe the samples, which could be a frame of video, a time-contiguous series of video frames, or a time-contiguous compressed section audio.

Accordingly, the protection in ISO Base Media File Format level is classified into item-based protection and sample-based protection, as described in Section E.5.1 and Section E.5.3, respectively.

Several boxes are used from ISO/IEC 15444-12, these are marked as “Existing” in Table E.1. Boxes defined in this specification are listed as “New” in Table E.1. The definitions for these boxes depend on the definition of Box and FullBox from ISO/IEC 15444-12, which is repeated for convenience in an informative annex of this specification.

Table E.1 – List of existing and new boxes

Box names						Status	Remarks
meta						Existing	Metadata
	iloc					Existing	Item location
	iproc					Existing	Item protection
		sinf				Existing	Protection scheme information box
			frma			Existing	Original format box
			schm			Existing	Scheme type box
			schi			Existing	Scheme information box
				gran		New	Granularity box
				vall		New	Value List box
				bcip		New	Block cipher box
					keyt	New	Key template box
				scip		New	Stream cipher box
					keyt	New	Key template box
				auth		New	Authentication box
					Keyt	New	Key template box
	iinf					Existing	Item information box
	ides					New	Item description box
		dest				New	Description type box
		desd				New	Description data box
		vide				New	Visual item description entry
		j2ke				New	JPEG 2000 item description entry
	icor					New	Item correspondence box
…	…	…	…	…	…	…	…
stbl						Existing	Sample table box
	stsd					Existing	Sample description box
		ScalableSampleDescriptionEntry				New	Scalable sample description

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

							entry
	sbgp					Existing	Sample to group box
	sgpd					Existing	Sample group box
		ScalableSampleGroupEntry				New	Scalable sample group entry
Gprt						New	Generic Protected box

E.3.5 Decryption Scheme

The Decryption protection scheme is identified in SchemeTypeBox as follows:

```
scheme_type="decr"
scheme_version=0
scheme_uri=null
```

For the Decryption protection scheme, the structure of SchemeInfoBox is as follows:

```
aligned(8) class GranularityBox extends Box('gran') {
      unsigned int(8) granularity;
}
```

Semantics:
`Granularity` is used for item-based protection. For item-based protection, 0 indicates that the processing unit is the entire item and 1 indicates that the processing unit is one extent within an item. For sample-based protection, 0 indicates that the processing unit is all samples in the track or sample group and 1 indicates that the processing unit is one sample.

```
aligned(8) class ValueListBox extends Box('vall') {
      unsigned int(8) value_size;
      unsigned int(16) value_count;
      unsigned int(16) count [value_count];
      unsigned char (value_size) value[value_count];
}
```

Semantics:
`value_size` is the size in bytes of each value in the array.
`value_count` is the number of (`count, value`) pairs in the array. For item-based protection, the (`count, value`) pairs are used to map each value to `count` processing units. For sample-based protection, the (`count, value`) pairs are used to map each value to `count` samples. For instance, the `value[0]` corresponds to the first `count[0]` sample or units, and the `value[1]` corresponds to the next `count[1]` samples or units, and so on.

```
aligned(8) class KeyTemplateBox extends Box('keyt') {
      unsigned int(16) key_size;
      unsigned int(8) key_info;
      GranularityBox GL;            //optinal
      ValueListBox    VL;
}
```

Semantics:
`key_size` is the size of key in bits.
`key_info` indicates the meaning of the values in the ValueListBox. 1 means the values are X.509 certificate; 2 means the values are URIs for certificate or secret keys.
`GL` is a GranularityBox.
`VL` is a ValueListBox, containing a list of values, whose meaning are defined by the `key_info` field.

```
aligned(8) class BlockCipherBox extends Box('bcip') {
```

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        unsigned int(16) cipher_id;
        bit (6) cipher_mode;
        bit (2) padding_mode;
        unsigned int(8) block_size;
        KeyTemplateBox KT;
}
```

Semantics:
`cipher_id` identifies which block cipher algorithm is used for protection. Values are defined in Table 25.
`cipher_mode` could be ECB, CBC, CFB, OFB or CTR. Values are defined in Table 29.
`padding_mode` is ciphertext stealing or PKCS#7-padding. Values are defined in Table 30.
`block_size` is size of block for block cipher.
`KT` is a KeyTemplateBox, holding all the key information used by the block cipher.

```
aligned(8) class StreamCipherBox extends Box('scip') {
        unsigned int(8)  cipher_type;
        unsigned int(16) cipher_id;
        KeyTemplateBox KT;
}
```

Semantics:
`cipher_type` indicates the type of cipher used. It has values `cipher_type` = STRE for stream cipher or `cipher_type` = ASYM for asymmetric cipher.
`cipher_id` identifies the stream cipher algorithm used for the protection. If `cipher_type` = STRE, see Table 26; If `cipher_type` = ASYM, see table 27.
`KT` is a KeyTemplateBox, holding all the key information used by the stream cipher.

```
aligned(8) class SchemeInfomationBox extends Box('schi', cipher_id) {
        unsigned int(8) MetaOrMedia;
        unsigned int(8) HeaderProtected;
        BlockCipherBox(); or    StreamCipherBox();
        GranularityBox GL;
        ValueListBox   VL;
}
```

Semantics:
`MetaOrMedia` is to indicate whether the protected data segment corresponds to media data segment or meta data boxes. (0 for media data and 1 for meta data boxes).
`HeaderProtected` is to indicate whether the protection is applied to the box content only (value 0) or applied to the whole box including its header (value 1).
The SchemeInformationBox can contain a BlockCipherBox, or a StreamCipherBox , which are containers for the parameters of the cipher algorithms. These boxes can only contain particular cipher_id values.
`GL` is a GranularityBox, holding information about the processing unit. This field is optional for sample-based protection and required for item-based protection.
`VL` is a ValueListBox. For block cipher, this box may be empty. For stream cipher, this box contains all the initial vectors.

E.3.6 Authentication Scheme

The Authentication protection scheme is identified by SchemeTypeBox as follows:

```
scheme_type="auth"
scheme_version=0
scheme_uri=null
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

For Authentication protection scheme, the structure of SchemeInfoBox is defined as follows:

```
aligned(8) class AuthBox extends Box('Auth') {
      unsigned int(8) auth_type; //hash, cipher, or signature
      unsigned int(8) method_id;
      unsigned int(8) hash_id;
      unsigned int(16) MAC_size;
      KeyTemplateBox KT;
}
```

Semantics:
auth_type indicates authentication type, including hash-based (HASH), cipher-based (CIPH) and signature-based (SIGN) authentication.
method_id identify the authentication method. 1 indicates HMAC. If auth_type = HASH, See Table 36; if auth_type = CIPH, see Table 39; if auth_type = SIGN, see Table 41.
hash_id identifies the hash function used. If auth_type = HASH or SIGN, see table 37; If auth_type = CIPH, see table 25.
MAC_size is size of MAC (if auth_type = HASH or CIPH) or digital signature (if auth_type = SIGN) in bits.
KT is a KeyTemplateBox, holding all the key information for the authentication.

```
aligned(8) class SchemeInfoBox extends Box('schi', auth_method) {
      unsigned char (8) auth_method;
      AuthBox authBox;
      GranularityBox GL;                  //optional
      ValueListBox VL;
}
```

Semantics:
auth_method identifies the authentication method used. 0 is for hash-based MAC; 1 is for cipher-based MAC; 2 is for digital signature. Depending on the Auth_method, this box could contain either HashAuthBox, CipherAuthBox, or SignatureAuthBox.
GL is a GranularityBox. The field is optional for sample-based protection and required for item-based protection.
VL is a ValueListBox, holding all the MACs or signatures.

E.3.7 ItemDescriptionBox

In the Item Location Box, all the items are specified as byte ranges (using the offset and length). The 'iloc' box does not contain the content-related information about the specified byte ranges, which is required by some protection methods. For instance, if secure transcoding method wants to discard the least important layer or resolution, it has to know which byte ranges correspond to the to-be-discarded layer or resolution.

As such, this section defines two new boxes to enable the content-related processing at file format level: Item Description Box ('ides') and Item Correspondence Box ('icor'). The 'ides' box specifies the content-related information, like layer, resolution (for visual content), period (audio content) and so on. The 'icor' box links the content-related information to the items in 'iloc' box, in the same way as the 'iinf' box links 'iloc' box to 'ipro' box.

The Item Description box ('ides') is defined as follows:

```
aligned(8) class ItemDescriptionBox extends Box('ides') {
      unsigned int(32) entry_count;
      for(i=0; i < entry_count; i++) {
                  DescriptionTypeBox desType;
                  unsigned int(32) item_ID;
                  DescriptionDataBox desData;
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

```
        }
}
```

Semantics:
Entry_count is the number of entries in the ItemDescriptionBox.
item_ID references an Item in ItemLocationBox. If this field is 0, then Item_ID will be specified by ItemCorrespondenceBox.

```
Aligned(8) class DescriptionTypeBox extends Box('dest') {
        Unsigned int(32) description_type;
        Unsigned int(32) description_version;
        Unsigned int(8) description_uri[];
}
```

Semantics:

description_type is the 4CC code defining the description scheme.
description_version is the version of the description.
description_uri allows for the options of directing the user to a webpage if they do not have the description definition installed on their system. It is an absolute URI formed as a null-terminated string in UTF-8 characters.

```
Aligned(8) class DescriptionDataBox extends Box('desd') {
        Box description_specific_data [];
}
```

Semantics:
If description_type = 'vide', this is an VisualItemDescriptionEntry; if description_type = 'j2ke', this is a J2KItemDescriptionEntry; For any other value of description_type, the syntax of the description can be found at description_uri.

The VisualItemDescrptionEntry is defined as follows:

```
aligned(8) Class VisualItemDescriptionEntry extends Box('vide') {
        unsigned int(16) layer_start;
        unsigned int(16) layer_count;
        unsigned int(16) res_start;
        unsigned int(16) res_count;
        unsigned int(16) horizontal_offset;
        unsigned int(16) horizontal_length;
        unsigned int(16) vertical_offset;
        unsigned int(16) vertical_length;
        unsigned int(16) color_space;
        unsigned int(16) time_start;
        unsigned int(16) time_length;
}
```

Semantics:
The layer_start and layer_count together specify the range of layers. When layer_start equals to 2^{16}-1, the layer range will start from layer 0; when layer_count equals to 2^{16}-1, the layer range will end at the last layer. When both layer_start and layer_count equal to 2^{16}-1, the layer range will include all layers.
The res_start and res_count together specify the range of resolutions. The semantics is the same to layer_start and layer_count when their value equals to 2^{16}-1.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The horizontal_offset, horizontal_length, vertical_offset and vertical_length together specify the spatial area. The semantics is the same to layer_start and layer_count when their value equals to 2^{16}-1.
The color_space specifies the color space. 0: red color space; 1: green color space; 2: blue color space.

The intersection is applied to the layer ranges, resolution ranges, areas and color space to get the portion of the image/video specified by the VisualItemDescriptionEntry.

The J2KItemDescriptionEntry is defined as follows:

```
aligned(8) class J2KItemDescriptionEntry extends Box('j2ke') {
      VisualItemDescriptionEntry visualDesEntry; //optional
      unsigned int(16) tile_start;
      unsigned int(16) tile_count;
      unsigned int(16) precinct_start;
      unsigned int(16) precinct_count;
      unsigned int(16) j2k_packet_start;
      unsigned int(16) j2k_packet_count;
}
```

Semantics:
visualDesEntry specifies image/video-specific attributes. It is optional.
Tile_start and tile_count specify the tiles.
Precinct_start and precinct_count specify the precincts.
J2k_packet_start and j2k_packet_count specify the JPEG 2000 defined packets.

Similar to VisualItemDescriptionEntry and AudiItemDescriptionEntry, the intersection is applied to tiles, precincts and JPEG 2000 packets to get the portion of the JPEG 2000 codestream specified by J2KItemDescriptionEntry.

The ItemCorrespondenceBox ('icor') is defined as follows:

```
aligned(8) class ItemCorrespondenceEntry extends Box('icoe') {
      unsigned int(16) item_ID;
      unsigned int(16) desc_ID;
}
```

Semantics:
Item_ID is pointing to one item in the ItemLocationBox.
Desc_ID is pointing to one description entry in the ItemDescriptionBox.

E.3.8 ScalableSampleDescriptionEntry Box

The ScalableSampleDescriptionEntry is used to describe characteristics associated with Scalable Composed ES or Decodable Composed ES, like resolution levels, quality layers, cropped region. For Scalable Composed ES, res and layer indicate the media data of that particular resolution and layer, while for decodable Composed ES, they indicate the highest resolution and highest quality layer.
When the media samples are protected by a protection tool, the ScalableSampleDescriptionEntry is encapsulated as follows:

The four-character-code of the ScalableSampleDescriptionEntry is replaced with another four-character-code indicating protection encapsulation, which varies only by media type, as defined below:

- Encv: to indicate that video samples are encrypted and thereby un-protection must be applied to get meaningful media data
- Autv: to indicate that video samples are authenticated and the media data can still be meaningful before un-protection.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- Enct: to indicate that text samples are encrypted.
- Autt: to indicate that text samples are authenticated
- Encs: to indicate that system samples are encrypted
- Auts: to indicate that system samples are authenticated.

A protectionSchemeInfoBox is added to the ScalableSampleDescriptionEntry, leaving all other boxes unmodified. The original sample entry type is stored in the OriginalFormatBox within the protectionSchemeInfoBox.

```
Class ScalableSampleDescriptionEntry(codingname)extends
      VisualSampleEntry(codingname) {
      Unsigned int(8)         res;
      Unsigned int(8)         layer;
      Unsigned int(32)        cropped_width, cropped_height;
      If(cropped_width > 0 && cropped_height > 0) {
            Unsigned int(32)        startx;
            Unsigned int(32)        starty;
      }
      ProtectionSchemeInfoBox protectionSchemes[]; //optional
}
```

Semantics:
`Codingname` is "sces" if the track is referring to a Scalable Composed ES and "dces" if the track is referring to a Decodable Composed ES.
`Res` is the resolution of the described samples. A value of -1 indicates all resolution levels.
`Layer` is the quality layer of the described samples. A value of -1 indicates all quality layers.
`cropped_width` is the width of the cropped region.
`cropped_height` is the height of the cropped region.
`startx & starty` indicate the position of the top-left corner of the cropped region. If either cropped_width or cropped_height is zero, the startx and starty will not present.
`protectionSchemes` is a list of ProtectionSchemeInfoBoxes to indicates the protection tools applied to the described samples. The un-protection process has to following the order they appear in the list.

E.3.9 ScalableSampleGroupEntry Box

A track may be made up of samples with different characteristics and protected by different protection tools or with different parameters. The ScalableSampleGroupEntry box is used to signal the grouping of samples based. For example, if a track has 1000 samples where the first 500 samples are encrypted with Key 1 and the second 500 samples are encrypted with Key 2, the SampleGroupDescription Box contains two ScalableSampleGroupEntry boxes: the first box describes the first 500 samples while the second box describes the second 500 samples.

When the media samples are protected by a protection tool, the ScalableSampleGroupEntry is encapsulated in the same way as ScalableSampleDescritpionEntry in Section E.3.7.

```
Class ScalableSampleGroupEntry(type)      extends VisualSampleGroupEntry(type) {
      unsigned int(8)         res;
      unsigned int(8)         layer;
      unsigned int(32)        cropped_width, cropped_height;
      If(cropped_width > 0 && cropped_height > 0) {
            unsigned int(32)        startX;
            unsigned int(32)        startY;
      }
      ProtectionSchemeInfoBox protectionSchemes[]; //optional
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
}
```

Semantics:
`type` indicates the grouping type. If the grouping is based on different protection tools applied to the samples, the type is "prot"; if the grouping is based on different media characteristics (like resolution or layers), the type is "attr".
`res` is the resolution of the described samples. A value of -1 indicates all resolution levels.
`layer` is the quality layer of the described samples. A value of -1 indicates all quality layers.
`cropped_width` is the width of the cropped region.
`cropped_height` is the height of the cropped region.
`startx & starty` indicate the position of the top-left corner of the cropped region. If either cropped_width or cropped_height is zero, the startx and starty will not present.
`protectionSchemes` is a list of ProtectionSchemeInfoBoxes to indicates the protection tools applied to the described samples. The un-protection process has to following the order they appear in the list.

E.3.10 Generic Protected Box

E.3.10.1 Definition

Box Types: 'gprt'
Container: File or any container box
Mandatory: Yes, when failure to use would prevent parsing file
Quantity: Any number

The JProtected Box is used when a protection scheme is applied to a box, and use of the protection scheme prevents parsing of the box. For example if the contents of a superbox are encrypted including the box lengths and types that box is no longer parsable, and thus fails to meet the original defintion for the box. In this case a JProtected Box may be placed in the file in the place of the box that no longer parses correctly, and the encrypted data placed in the JProtected Box.

Interpreting the content of the data[] portion of this box, shall be done using the Item Location Box.

Once all of the protection methods have been operated on from the ItemInformation Box, the contents may be reorganized into original boxes. The first size[0] bytes of the unprotected data[] array are placed in a box of type type[0], and the next size[i] bytes are placed in a box of type[i] and so on. Note that when size_flag is 0, the size of the original box is not disclosed and when type_flag is 0, the type of the original box is not disclosed. This is to prevent known-plaintext attacks. When both the size_flag and type_flag are 0, total_size provides the total size of the protected content.

If the entry count is 0, or if there is data left over, then that data shall be in the format of legal boxes with type and size codes, i.e. the unprotected data contains the types and sizes.

If any part-1 mandatory box is encrypted, the "jp2" brand should be removed from the compatible list in file type box.

E.3.10.2 Syntax

```
aligned(8) class JProtectedBox extends Box('gprt') {
      bit(1) type_flag;
      bit(1) size_flag;
      bit(1) location_flag;
      unsigned int(5) reserved; // for ISO use
      if(size_flag == 1 || type_flag == 1 || location_flag == 1) {
            unsigned int(32) entry_count;
            if(location_flag == 1)
                  unsigned int(8) offset_size;
            for(i=0; i<entry_count; i++) {
                  if(size_flag == 1)
                        unsigned int(32) size;
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                if(type_flag == 1)
                        unsigned int(32) type = boxtype;
                if(size_flag == 1 && size == 1)
                        unsigned int(64) large_size;
                if(location_flag == 1)
                        unsigned int(offset_size*8) offset;
            }
        }
        else {
            Unsigned int(32) total_size;
            if(total_size == 1)
                unsigned int(64) large_total_size;
        }
        unsigned int(8) data[];
}
```

E.3.10.3 Semantics

`size_flag` indicates whether or not size value presents. A value of 1 means the size of each entry in Generic Protected Box presents; 0 means the size does not present.
`type_flag` indicates whether or not the box type presents. A value of 1 means the original type of each entry in Generic Protected Box presents; 0 means the type does not present.
`location_flag` indicates whether or not the location presents. A value of 1 means the original location of each entry in Generic Protected Box presents; 0 means the original location does not present.
`entry_count` is the number of entries in the Generic Protected Box.
`offset_size` is the length of the `offset` in bytes.
Each `size` entry is the size of a box that has been replaced by the Generic Protected Box.
Each `type` entry is the original type of a box that has been replaced by the Generic Protected Box.
Each `offset` entry is the offset of a box that has been replaced by the Generic Protected Box.
`total_size` is the total size of the entries in the Generic Protected Box.
`data[]` is an array of bytes to the end of the box the may be referenced by the Item Location box and thus protected with a method defined in the Item Protection box.

E.4 Elementary Stream and Sample Definitions

E.4.1 Overview

This section defines the structure of an Elementary Stream (ES).

An elementary stream contains media data, ByteData structure, Container structure, Pointer structure, or any mixture of the above. There are two types of ES: One type is called *self-contained ES*, which does not contain any Container, Pointer or ByteData structure. The format of self-contained ES is not defined in this specification, and it can be any media format like motion JPEG2000 sequence. The other type is referred to as *composed ES*, where the media data is composed from other Elementary Streams. A composed ES can either copy data from other tracks or reference data in other tracks. When a media data segment is copied from other track, the segment must be wrapped in ByteData structure; the Pointer structure is used to reference media data segments in other ES. When a sample in composed ES consists of more than one structure, a Container structure must be used to wrap all structures belonging to that sample.

The composed ES can be classified into *Scalable Composed ES* and *Decodable Composed ES*. The former is designed to support scalability, i.e., to make the codestream "thinable", the adaptor can make a codestream by combining multiple such ESs. However, it is assumed that the adaptor should be able to generate new headers and modify the index number within each packet. The Decodable composed ES is designed for simple adaptor that has no capability to generate headers or modify packet index numbers. The adaptor simply follows the instructions in a Decodable Composed ES to generate a complete decodable codestream.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Figure E.2 illustrates the relationship between Self-contained ES and Scalable Composed ES. In Scalable Composed ES, each sample is wrapped by a Container structure, where a Pointer is referring to the desired data segment (s) in the Self-contained ES. Note that a Scalable Composed ES does not contain any headers, the adapter has to dynamically generate the header to make a codestream that is decodable by normal decoder. In addition, each Scalable Composed ES may not constitute a complete decodable codestream by itself, and multiple Scalable Composed ESes may be required to make a complete codestream. For example, in Figure E.2, the adaptor has to combine the Scalable Composed ES 0 and 1 to make a resolution-1 JPEG 2000 frame.

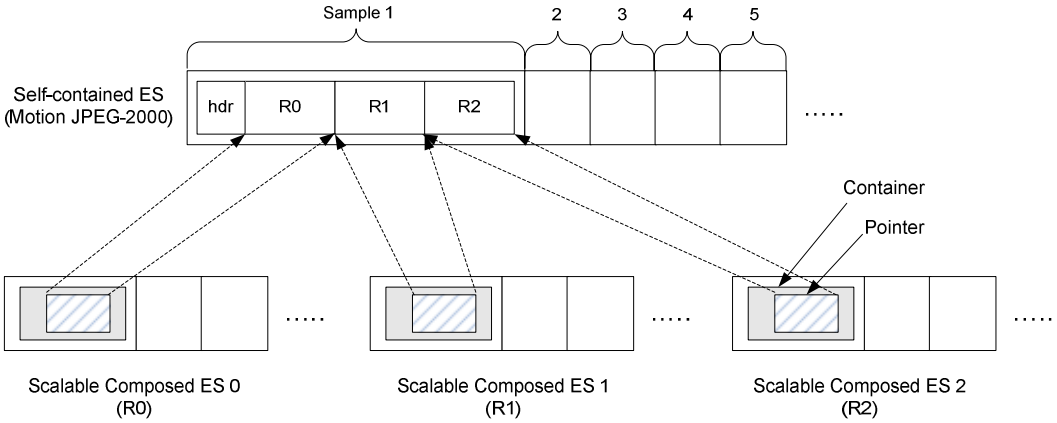


Figure E.2 – Self-contained ES and Scalable Composed ES

Figure E.3 illustrates the relationship between Self-contained ES and Decodable Composed ESs. In Decodable Composed ES, a sample is wrapped by a Container, which includes the *Data* structure (with header information), and one or more Pointer structures referring to data segments in the self-contained ES.

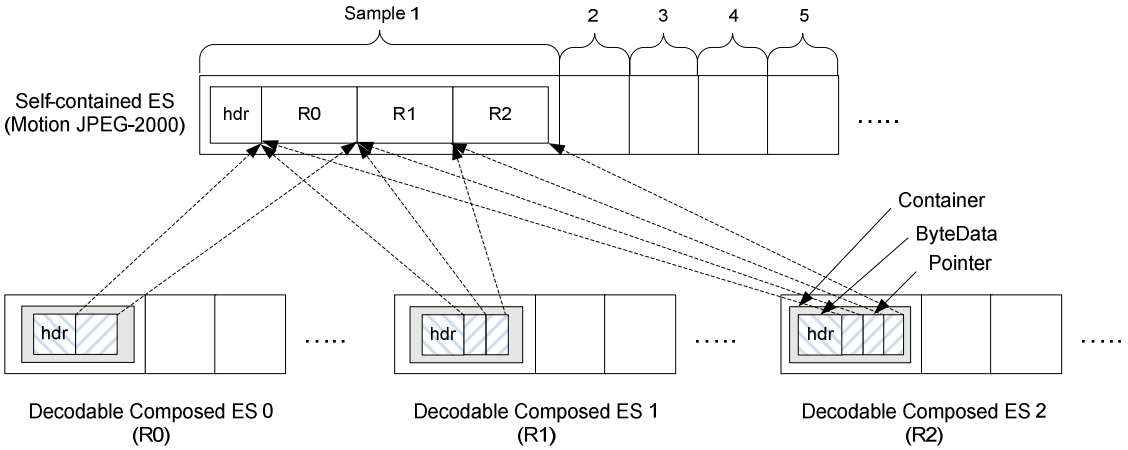


Figure E.3 – Self-contained ES and Decodable Composed ES

E.4.2 In-stream Structures

This section defines the in-stream structures used in Scalable Composed ES and Decodable Composed ES. A Composed ES can include media from other Elementary Streams either by copy or by reference. The ByteData structure is used to include media data by copy while the Pointer structure is used to include media data by reference. In addition, each sample is wrapped by a Container structure, which may contains one or more ByteData or Pointer structures.

```
class aligned(8) DataUnit(type) {
      unsigned int(32) type;
      unsigned int(32) size;
      unsigned int(32) RDHints;
}
class ByteData (type='bdat') extends DataUnit {
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        unsigned int(8) data[];
}
class Container (type='cont') extends DataUnit {
        DataUnit(type) units[];
}
class Pointer(type='poin') extends DataUnit {
        unsigned int(8) track_ref_index;
        unsigned int(8) segment_count;
        for(int i=0; i<segment_count; i++) {
                unsigned int(32) offset;
                unsigned int(32) length;
        }
}
```

Semantics:

`Type` indicates the type of the data structure, like "ByteData", "Container" and "Pointer".
`Size` is the size of this structure, including itself and subsequent data in this structure.
`RDHints` indicates the relative or absolute importance of media data contained or referenced by this object. For instance, `RDHints` could be associated distortion increment value, the amount by which the media distortion will increase if the media data is lost, or it could be importance ranking.
`Data[]` is a byte array containing media data in its original format. This amendment does not specify the format of the media data.
`Units[]` is a list of objects contained by Container object. The object can be either "ByteData" or "Pointer", but cannot be another "Container" object.
`Track_ref_index` specifies the index of the track to which this "pointer' is pointing.
`Segment_count` is the number of "offset" and "length" pair.
`Offset` is the offset of the first byte within the referred sample to copy;
`Length` is the number of bytes of the data segment

E.5 Protection at file format level

E.5.1 Overview

This section describes how media data is protected using the protection schemes (i.e. authentication or description) and how the protection is signaled using the boxed defines in Section E.3.4 and Section E.3.5.

In case that the security tools applied change data length, it shall update all pointers and length fields in all boxes, to ensure correct parsing by the reader.

E.5.2 Item-based protection for ISO base file format and JPEG family file formats

This annex defines two types of ProtectionSchemeInfoBox: authentication and decryption. The two protection schemes protect the scalable media without loss of scalability.

Each instance of a protection scheme and used parameters (like MAC, IV, keys, etc) are described by a ProtectionSchemeInfoBox located inside ItemProtectionBox. On the other hand, the ItemLocationBox contains a list of items and each item points to one or more contiguous segments of media data. The IteminformationBox maintains a mapping table between the items in ItemLocationBox and the protection schemes in ItemProtectionBox. Figure E.4 gives an example with two items and three protection schemes: item 0 is protected with Scheme 0 and Scheme 2, while Item 1 is protected with Scheme 1.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

As illustrated in Figure E.4, the same item can be applied with multiple protection schemes, e.g. item 0. In addition, the items can overlap each other and the overlapped region may be applied with multiple protection schemes. Therefore, it is important to specify the processing order among the protection schemes. The 'ffsc' file format mandates that the file MUST be un-protected in the same order as the schemes appear in 'iinf' box. For instance, for un-protection, the first scheme appearing in ‘iinf” box is applied first and the last scheme appearing in ‘iinf” box is applied last. In this example, the file is first un-protected with Scheme 2 on Item 0, followed by Scheme 1 on Item 1 and scheme 0 on Item 0.

Note: The protection order (with multiple protection tools) must be exactly the reverse of the un-protection order, that is, the first scheme appearing in “iinf” box is the last tool to be applied, and the last scheme appearing in “iinf” box is the first tool to be applied.

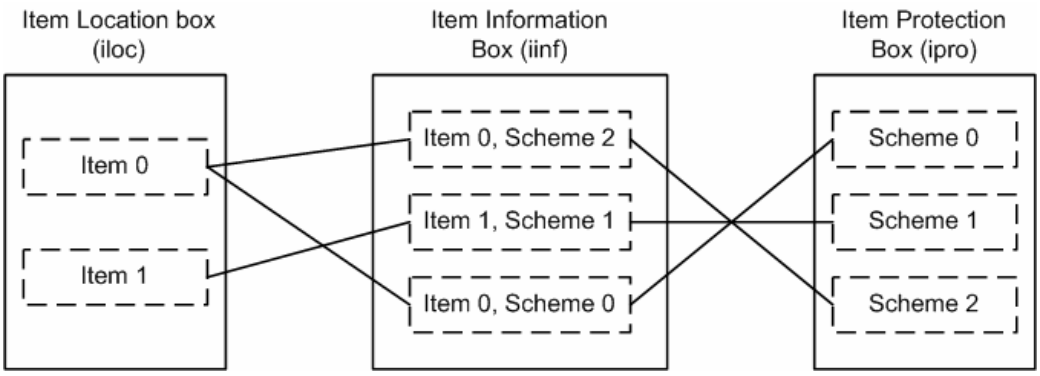


Figure E.4 – Relationship between iloc, iinf and ipro

E.5.3 Addition requirements for item-based protection for JPEG family file formats

JP2, JPX and JPM files are not based on the ISO base file format but have some common boxes, e.g. the File Type Box (‘ftyp’). In order to use the Item-based protection in this standard these file formats shall incorporate protection by adding the ‘meta’, ‘hdlr’, ‘ipro’, ‘iloc’, and ‘iinf” from 15444-12: 2005, in addition to the new boxes defined by this amendment.

Operation of protection is the same as for the ISO base file format. However, codestreams in JPEG family file formats may appear in contiguous codestream boxes in addition to Media Data (‘mdat’) boxes.

E.5.4 Sample-based protection

When a protection scheme is applied on sample basis, it is signaled by one protectionSchemeInfoBox located in either ScalableSampleDescriptionEntry or ScalableSampleGroupEntry. When the protectionSchemeInfoBox is located in ScalableSampleDescriptionEntry, the protection is applied to all samples in the track; when the protectionSchemeInfoBox is located ScalableSampleGroupEntry, the protection is applied only to the samples in the sample group.

When samples are applied with multiple protection schemes, the ScalableSampleDescriptionentry or ScalableSampleGroupEntry contains multiple ProtectionSchemeInfoBoxes. Under the major brand of 'ffsc', the samples must un-protected in the order that the corresponding ProtectionSchemeInfoBoxes.

Figure E.5 gives an example of "mp4v" sample description entry that is protected with an authentication followed by an encryption. Note that there are two ProtectionSchemeInfoBoxes in this entry: the first one is for the decryption scheme while the second one is for the authentication scheme. To un-protected the sample, the file reader has to apply the decryption scheme followed by the authentication scheme, which is the order that they appear in the sample entry.

When sample-based protection is applied to composed ES, the protection is actually applied to the media data which is contained or pointed by the Container, ByteData or Pointers object. For instance, to encrypt a sample in a composed ES, which is wrapped by a Container object, the protection process shall encrypt its media data only, retaining the structure of Container.

The sample-based protection can be applied to all samples in a track or sample group as a whole (when GL=0 in SchemeInformationBox) or separately to each sample (when GL=1 in SchemeInformationBox). In the former case, the ValueList has only one MAC or IV and in the latter case, the ValueList has one MAC or IV for each sample.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

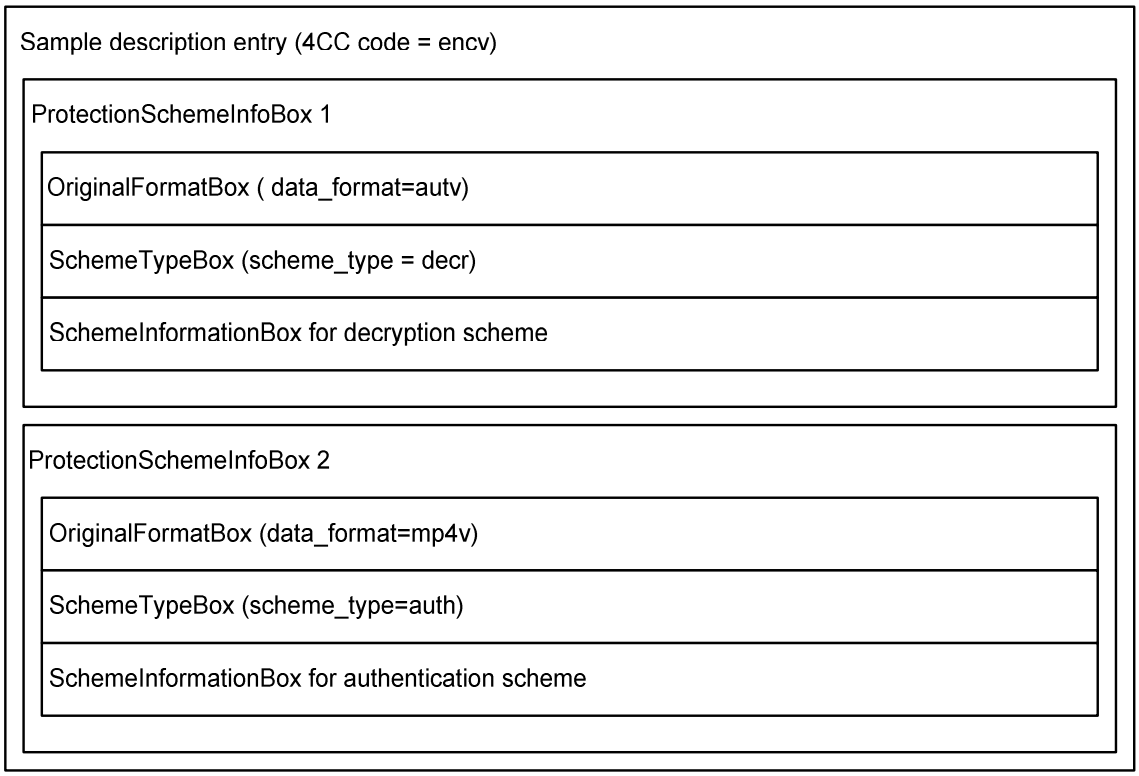


Figure E.5 – An example sample description entry protected by authentication scheme followed by description scheme

E.6 Examples (Informative)

E.6.1 Example 1

This example shows a very simple JPEG file (using JP2 file format) using only authentication. As shown in Figure E.6, authentication is applied to the coded media data stored in the "Contiguous codestream box", authentication algorithm is HMAC with SHA-1 hashing.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

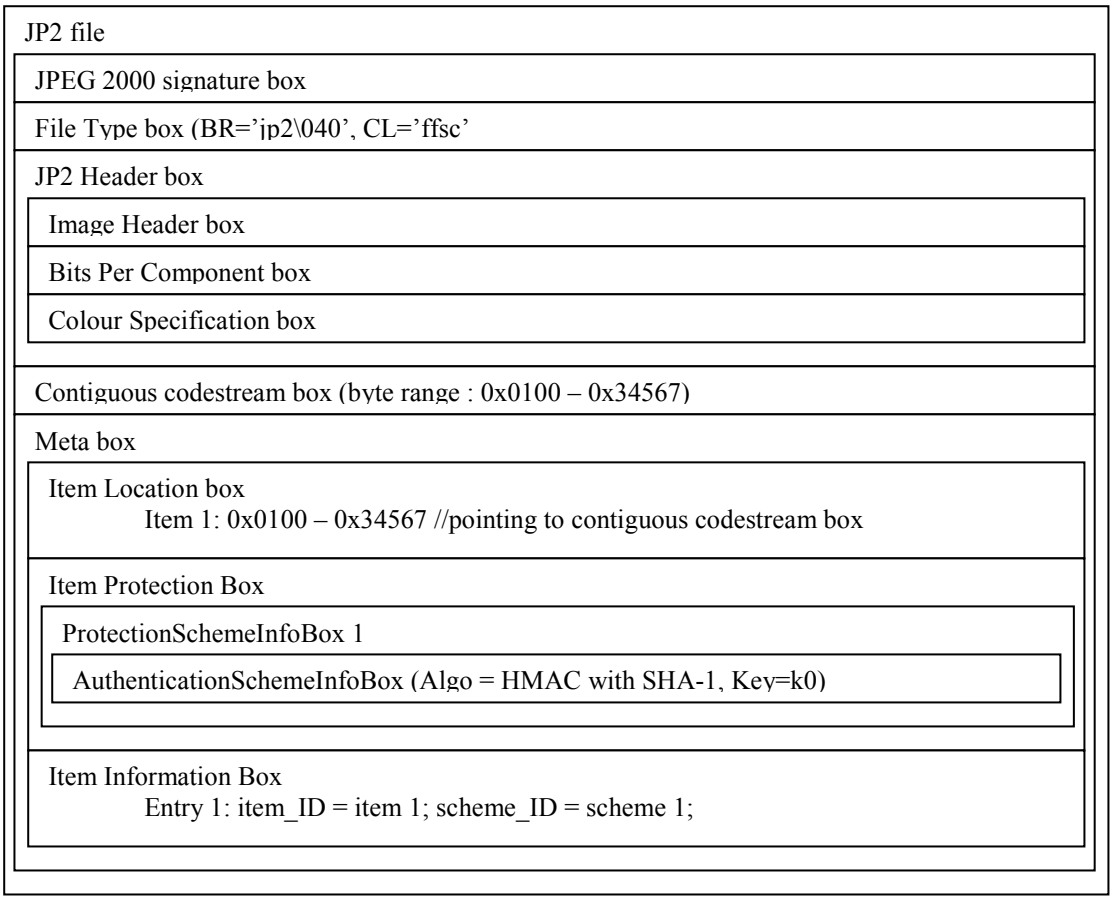


Figure E.6 – Example 1: Item-based protection of JP2 file (authentication)

E.6.2 Example 2

This example corresponds to the example given in Section 6.3.1. An image is coded with JPEG 2000 and has three resolutions. The first resolution is not encrypted in order to provide preview capability, and the second and third resolutions are encrypted with keys k1 and k2, respectively. The input image is coded in RLCP progression order and has 1 tile and 3 resolutions. (the number of layers, components and precincts are not important in this specific example). Encryption is performed using AES in CBC mode without padding (using cipher-text stealing), using k1 to encrypt resolution 1 and using k2 to encrypt resolution 2, and resolution is left unencrypted.

First of all, the ItemlocationBox contains two Items: one item points to the byte range from 0x31CC to 0xA3E8 and the other one points to the byte range from 0xA3E9 to 0x31101. The ItemProtectionBox contains two DecryptionSchemeInformationBoxes: the first one uses AES in CBC mode and key k1, the other uses AES in CBC mode and k2. The ItemInformationBox links the DecryptionSchemeInformationBoxes in ItemProtectionBox to the Items in ItemLocationBox. In ItemDescriptionBox, ItemDescription 1 describes Item1 as resolution 1 of the image, while ItemDescription 2 describes Item 2 as resolution 2 of the image.

Note that the JPEG 2000 image can be located in either MDAT box in the sample file as the META box, or a different file whose format is described by ISO Base File Format.

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

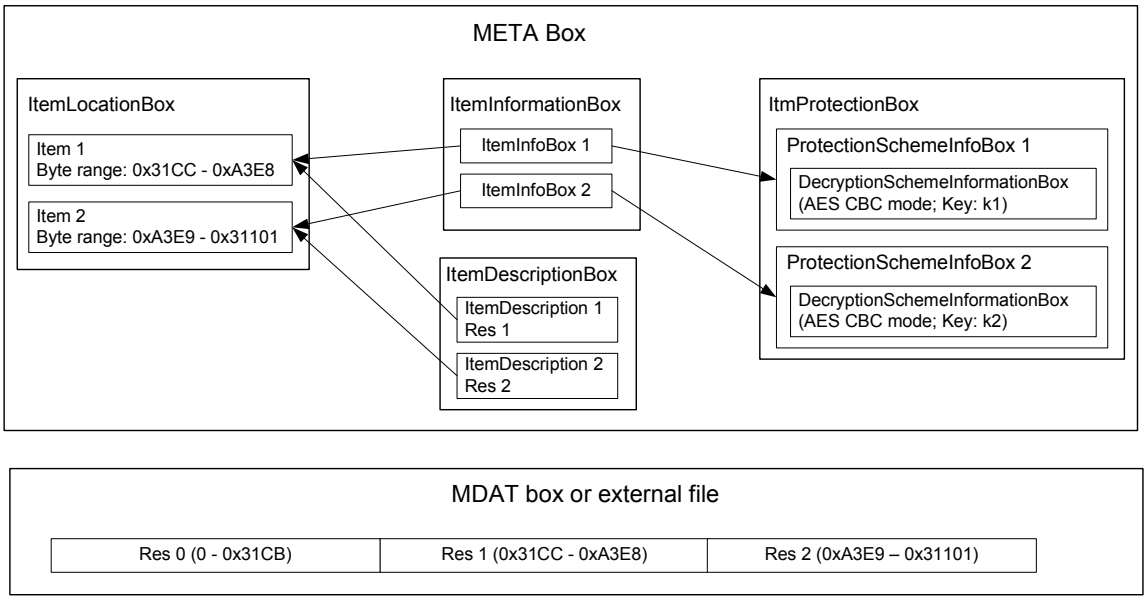


Figure E.7 – example 2: Item-based protection of a JPEG 2000 images (encryption)

E.6.2.1 Transcoding to resolution 1

To securely transcode the above example to lower resolution by discarding resolution 2, the transcoder needs to do the following:

- Discard the trunk of media data corresponding to resolution 2, i.e., byte range 0xA3E9 – 0x31101.
- Remove Item 2 from ItemLocationBox
- Remove the ItemInfoBox 2 from ItemInformationBox
- Remove ItemDescription 2 from ItemDescriptionBox
- Remove ProtectionSchemeInfoBox 2 from ItemProtection Box

After transcoding, the resulting file format is shown in Figure E.8. Note that the transcoder does not have to decrypt the media data, and thereby achieves end-to-end security.

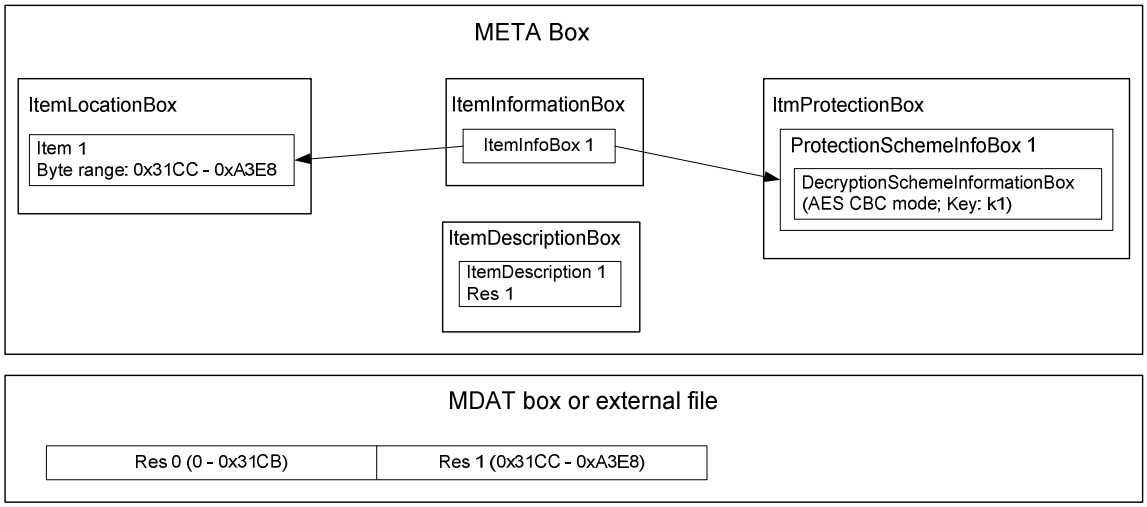


Figure E.8 – Example 2 : secure transcoding to lower resolution (discarding resolution 2)

E.6.3 Example 3

This example corresponds to the example given in Section 6.3.2. In this case authentication is applied to the same JPEG 2000 coded image as in Example 1. In this example all three resolutions are authenticated, where the authentication for each resolution uses a different key. Each resolution is authenticated using different keys, and each layer within a resolution has its own MAC value. In summary, there will be a total of three keys and nine MAC values for the entire image. Specifically

- Resolution 0 has three MAC values M0, M1 and M2 (one for each layer) using K0

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

- Resolution 1 has three MAC values M3, M4 and M5 (one for each layer) using K1
- Resolution 2 has three MAC values M6, M7 and M8 (one for each layer) using K2

The authentication is performed using HMAC with SHA-1 hash. As shown in Figure E.9, the ItemLocationBox has three items corresponding to three resolutions, and each item has three extents corresponding to three layers within a resolution. The ItemProtectionBox contains three PretectionSchemeInformationBoxes, the first one signals the applied authentication tool using K0 and three MAC values M0, M1 and M2, and so forth. The ItemInformationBox and ItemDescriptionBox are used in the same way as previous example.

Note that when "granularity" field in AuthenticationSchemeInformationBox is used to indicate the smaller protection unit of the authentication tool. When "granularity" is "extent", the authentication tool will generate one MAC for each extent (i.e., three MAC values for the entire Item in this example); when it is set to "item", it will generate only one MAC value for the entire item.

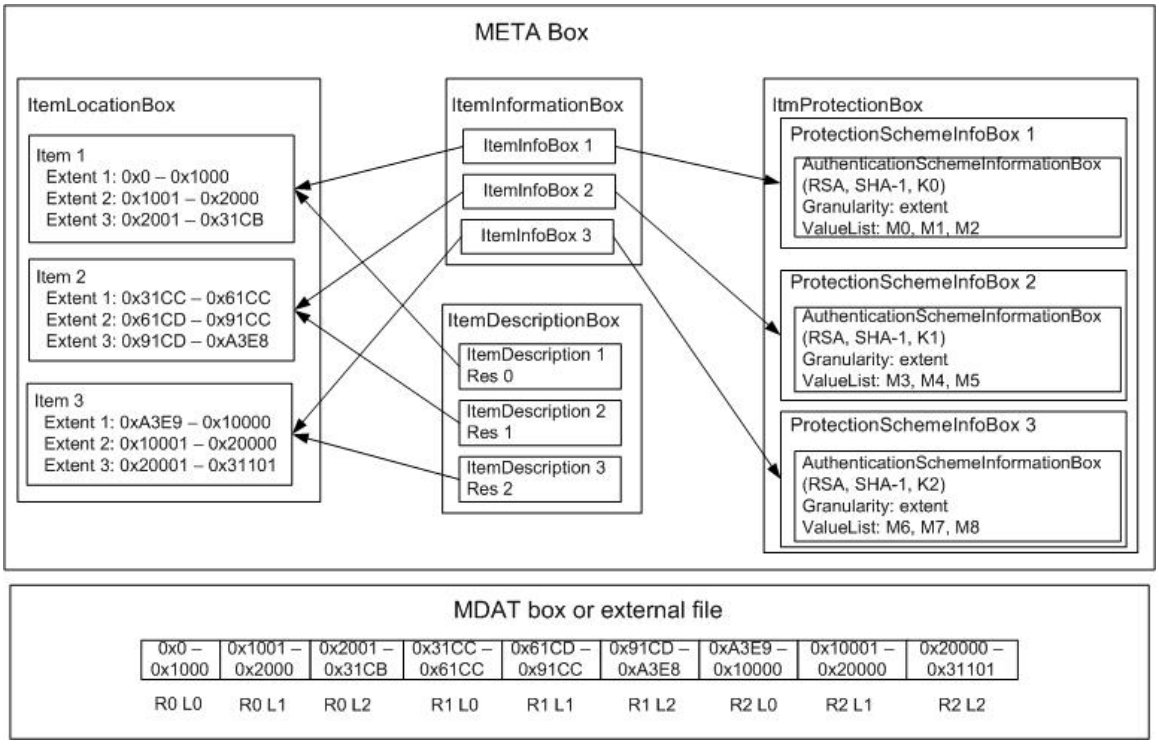


Figure E.9 – Example 3: Item-based Protection of a JPEG 2000 image (Authentication)

E.6.3.1 Transcoding to resolution 1

To securely transcode the above example to resolution 1, the transcoding has to do the following:
Discard trunk of media data corresponding to resolution 1, i.e., byte range 0xA3E9 – 0x31101.

- Remove the Item 3 from ItemLocationBox
- Remove ItemInfoBox 3 from ItemInformationBox
- Remove ItemDescription 3 from ItemDescriptionBox
- Remove ProtectionScheme 3 from ItemProtectionBox

The resulting file format of the transcoded codestream is shown in Figure E.10.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

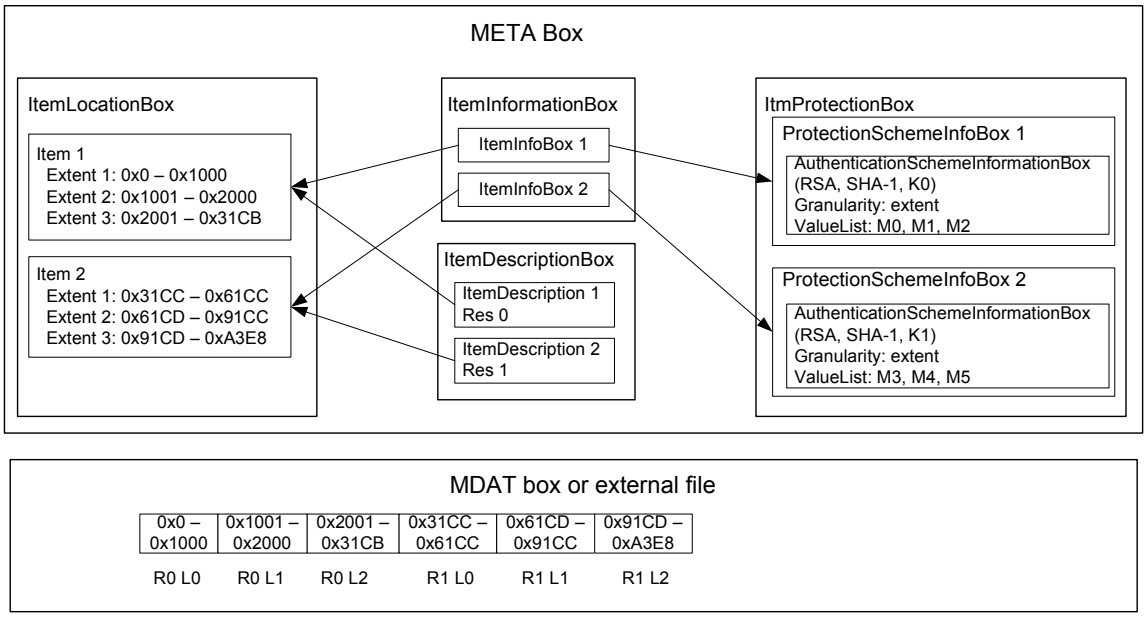


Figure E.10 – Example 3: transcoding to resolution 1

E.6.4 Example 4

This example illustrates sample-based protection for a time-sequenced JPEG 2000 coded pictures. Each pictures has three layers (L0, L1 and L2), and all three layers are authenticated using a unique key (AK0, AK1 and AK2). After that, the L1 and L2 are encrypted using a unique key (CK0 and CK1). That is, L0 is authenticated with AK0 producing one MAC value M0. L1 and L2 are first authenticated with AK1 and AK2 (producing M1 and M2) and then encrypted using CK0 and CK1 respectively. Authenticated is applied using HAMC with SHA-1 and encryption is achieved using AES in CBC mode.

To preserved scalability at file format level, three Scalable Composed ESes are generated, ES0, ES1, and ES2, one for each layer. Each ES is described by one media track, for instance, ES0 is described by Trak0, ES 1 is described by Trak1, and so on.

The format of the three tracks (Trak 0, Trak 1 and Trak 2) is illustrated in Figure E.11. Trak 0 has one ProtectionSchemeInfoBox as L0 is protected by authentication tool only. Trak 1 and Trak 2 have two ProtectionSchemeInfoBox as L1 and L2 are protected by both authentication and encryption tool. Note that the decryption tool appears in the first ProtectionSchemeInfoBox and the authentication tool appears in the second ProtectionSchemeInfoBox. The un-protection process has to follow the same order to get decodable JPEG 2000 codestream.

Note for Decodable Composed ES and Self-contained, sample-based protection are applied in the same way as for Scalable Composed ES.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

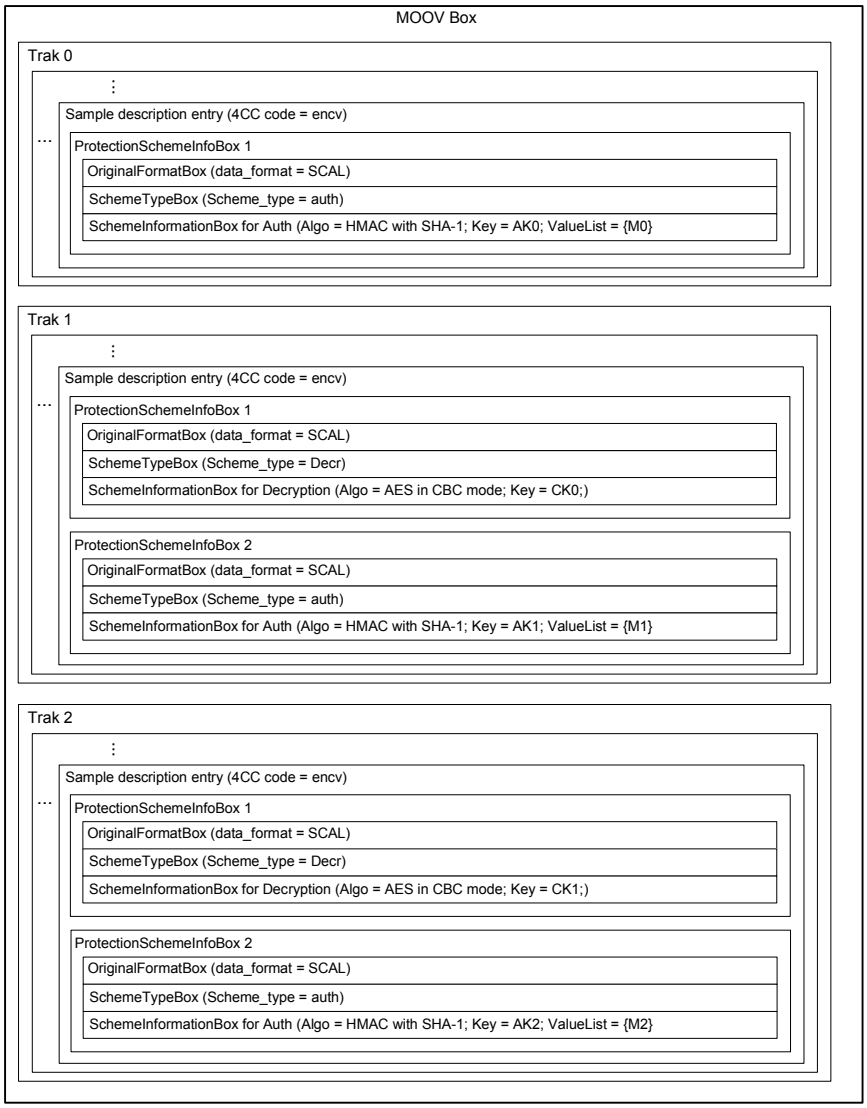


Figure E.11 – Example 4: sample-based protection of a time-sequenced JPEG 2000 pictures

E.6.4.1 Transcoding to layer 1

To securely transcode the sequence of JPEG 2000 picture to layer 1, the transcoder needs to do the following:

- Discard the Scalable Composed ES 2 corresponding to layer 2, and discard the trunk of media data in self-contained ES.
- If necessary, the transcoder also needs to update the byte range values of Pointer objects in the remaining two Scalable Composed ES.
- Discard Trak 2 that describes Scalable Composed ES 2.

The resulting file format is shown in Figure E.12.

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

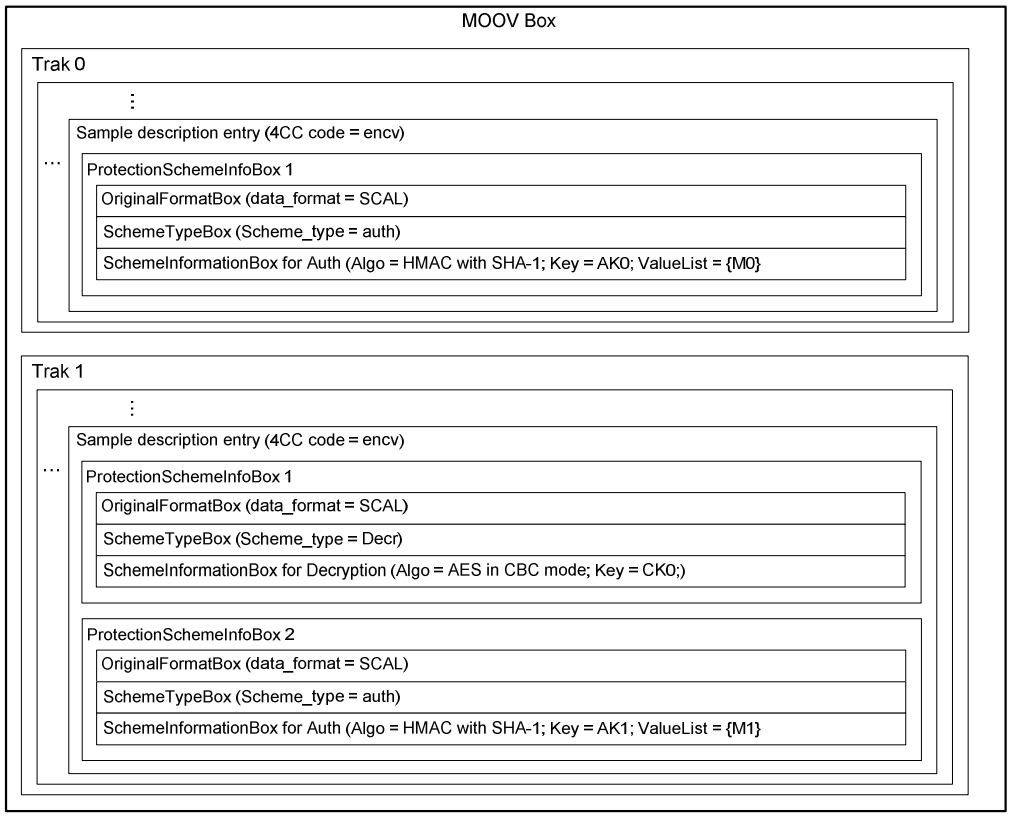


Figure E.12 – Example 4: secure transcoding to lower SNR quality (layer 1)

E.6.5 Example 5

This example illustrates how sample-based protection is applied to for video browsing and video summarization. In this example, a sequence of 10000 frames is summarized into 10 scenes. For instance, the first 1000 frames constitute the first scene; the second 1000 frames constitute the second scene and so on. For each scene, the first 50 frames (5-second video at 10 fps) are left unencrypted for preview purpose, the rest of the frames are encrypted using AES in CBC mode with key K0. In this example, the 10000 pictures are stored in a self-contained ES, which can be located in MDAT box or external file whose format is not specified by ISO Base File Format.

Figure E.13 illustrates the file format when the above protection is applied to a self-contained ES, which is described by Track 0. The SampleToGroupBox has 20 entries, the first 50 pictures of every scene are mapped to no descriptor and the next 950 pictures of each scene are mapped to the first ScalableSampleGroupEntry in SampleGroupDescriptionBox. The group_type is “prot”, for protection purpose.

The ScalableSampleGroupEntry has handler_type of “encv” and the protection is applied all resolutions (res=-1), all layers (layer=-1) and all regions of each sample (cropped_with=cropped_hight=0). As only one protection is applied to this sample group, the ScalableSampleGroupEntry contains only one ProtectionSchemeInfoBox, which in turn contains the OriginalFormatBox, SchemeTypeBox and SchemeInformationBox.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

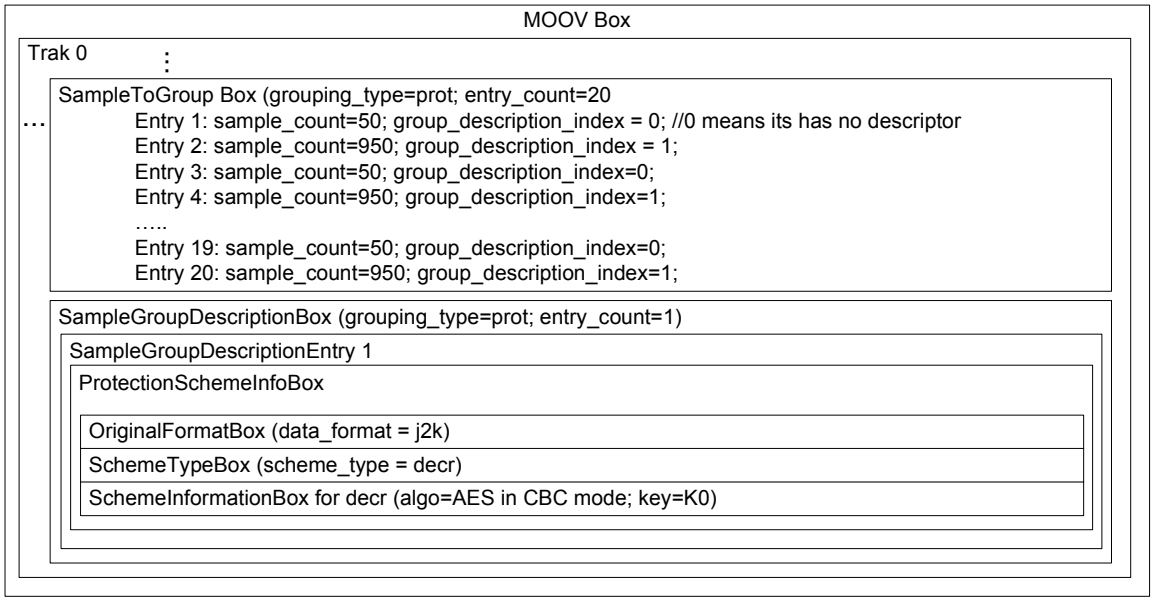


Figure E.13 – Example 5: sample-based protection for video browsing or video summarization

E.6.5.1 Example 5: transcoding to shorter time length

This example illustrates how to securely transcode the above codestream to shorter time length, i.e., discarding the last 5000 pictures. The transcoder has to do the following things:

- Discard the trunk of media data corresponding to the last 5000 pictures.
- Remove the entries corresponding to the last 5000 pictures in SampleToGroup Box.

The resulting codestream is shown in Figure E.14.

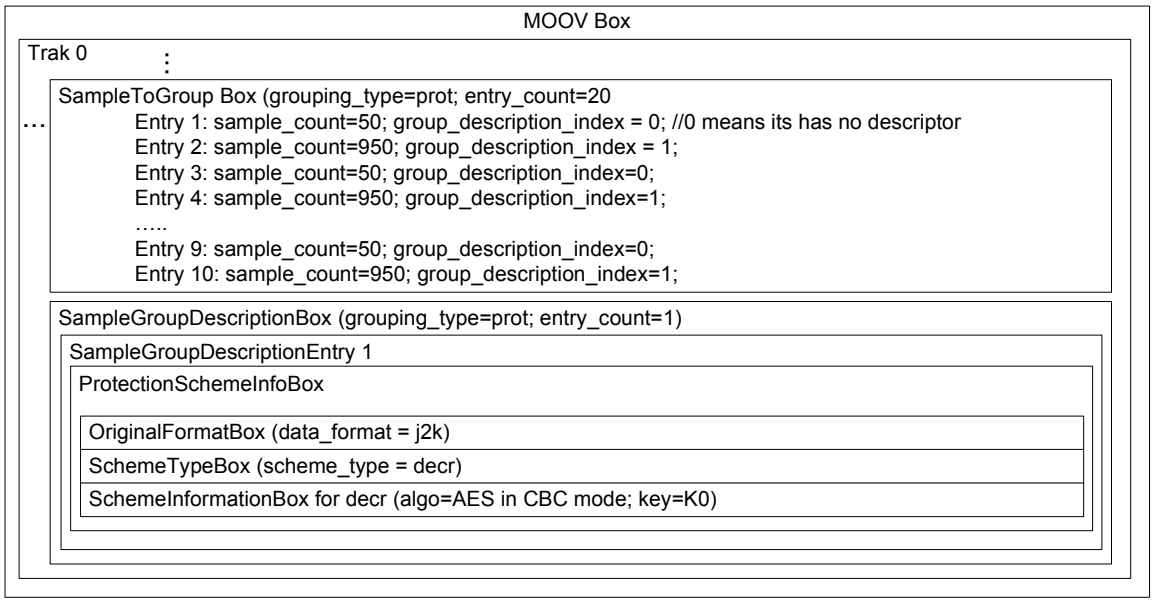


Figure E.14 – Example 5: trancoding to shorted time length (discarding last 500 pictures)

E.6.6 Example 6

This example demonstrates how the structures defined in this annex can be used to perform authentication and decoding of verified data.
An authentication adaptor / transcoder removes data that is not verifiable with the available media data and authentication data. For example, in a streaming system, some media packets may be lost during transmission. A file format receiver may reconstruct the received data to the best of its ability based on the available data. Then, an authentication adaptor / transcoder can determine which data can be verified, and then remove the packets that are not verifiable. The resulting file only contains the decodable, verified data.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

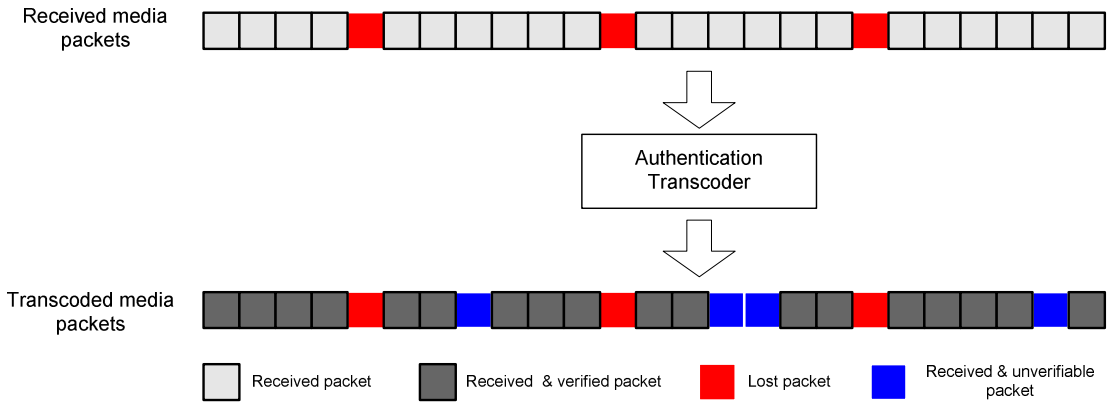


Figure E.15 – Example 6: authentication transcoding, discarding received but unverifiable packets

E.6.7 Example 7

This example shows the effect of length changes on the contents of a file due to protection coding.

The 'moov' box expanded in the figure below contains boxes that reference samples in the 'mdat' box. In this case the 'mdat' box contains six samples, labeled D1, …, D6.

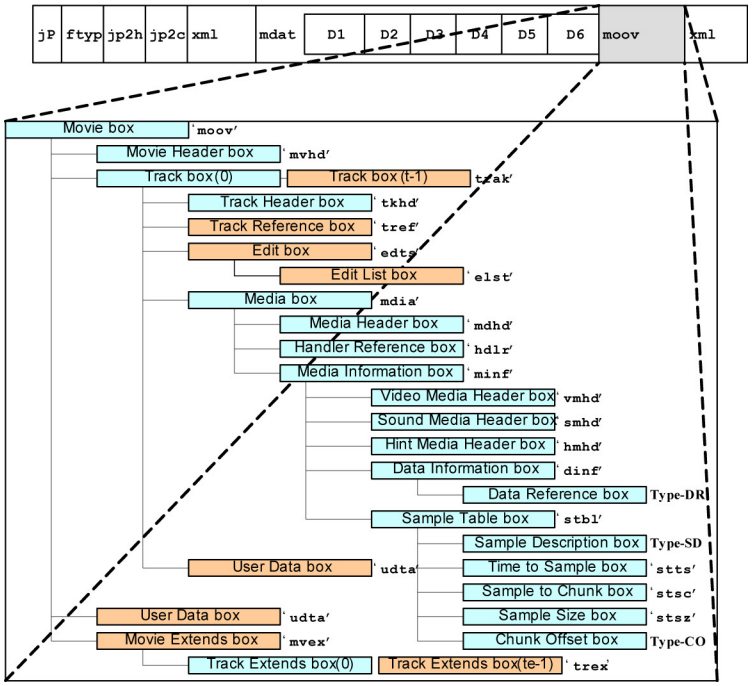


Figure 16 – Motion JPEG 2000 file with detailed box structure

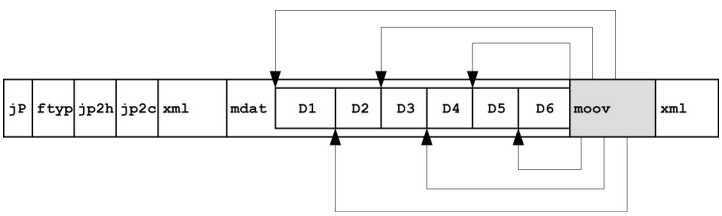


Figure E.17 – Simplified Motion JPEG 2000 box structure showing references

Without any protection some of the boxes and the references to those boxes appear as shown in the figure above.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

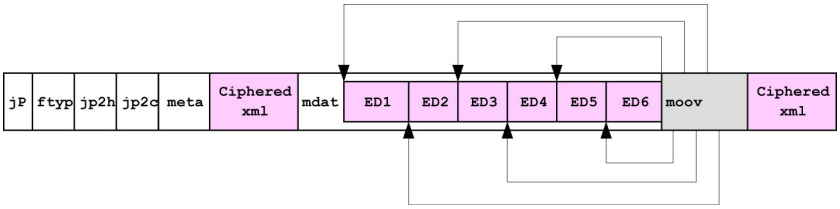


Figure E.18 – Simplified Motion JPEG 2000 box structure showing references after length changing protection operations

The figure above shows boxes or samples in pink that have been protected and may have had their length change. The pointers in the 'moov' box have been adjusted to point to the correct location of the protected samples.

Note: In order to function as shown in these figures, the protection tool must understand all pointers present in the file. Thus a protected Motion JPEG 2000 file requires a protection tool capable of adjusting Motion JPEG 2000 references. The alternative is for the file with portions that have been changed to be marked as unreadable by a Motion JPEG 2000 reader until all length changing operations have been undone.

E.6.8 Example 8

This example shows the effect of length changes on the conents of a JPM file due to protectioni coding.

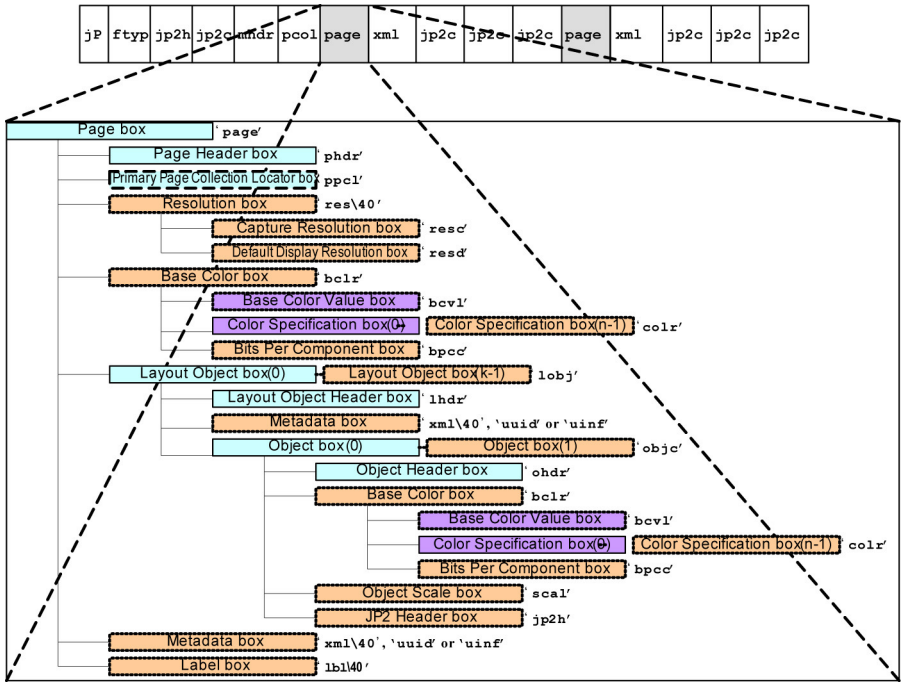


Figure E.19 – JPM file with detailed box structure

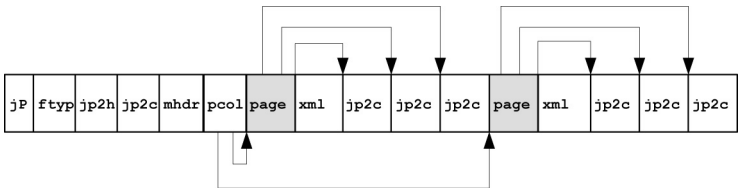


Figure E.20 – Simplified JPM box structure showing references

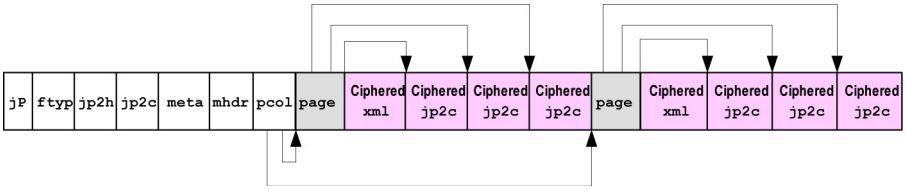


Figure E.21 – JPM box structure showing references after length changing protection operations

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

E.7 Boxes defined in ISO/IEC 15444-12 (informative)

This annex is not normative part of this amendment. The definitions listed here are repeated from ISO/IEC 15444-12 for convenience.

Box

Files are formed as a series of objects, called boxes in this specification. All data is contained in boxes; there is no other data within the file.

Boxes start with a header which gives both size and type. The header permits compact or extended size (32 or 64 bits) and compact or extended types (32 bits or full UUIDs). The standard boxes all use compact types (32-bit) and most boxes will use the compact (32-bit) size. Typically only the Media Data Box(es) need the 64-bit size.

The size is the entire size of the box, including the size and type header, fields, and all contained boxes. This facilitates general parsing of the file. The definitions of boxes are given in the syntax description language (SDL) defined in MPEG-4 (see reference in clause 2). Comments in the code fragments in this specification indicate informative material. The fields in the objects are stored with the most significant byte first, commonly known as network byte order or big-endian format.

```
aligned(8) class Box (unsigned int(32) boxtype,
                  optional unsigned int(8)[16] extended_type) {
      unsigned int(32) size;
      unsigned int(32) type = boxtype;
      if (size==1) {
            unsigned int(64) largesize;
      } else if (size==0) {
            // box extends to end of file
      }
      if (boxtype=='uuid') {
            unsigned int(8)[16] usertype = extended_type;
      }
}
```

The semantics of these two fields are:

`size` is an integer that specifies the number of bytes in this box, including all its fields and contained boxes; if size is 1 then the actual size is in the field largesize; if size is 0, then this box is the last one in the file, and its contents extend to the end of the file (normally only used for a Media Data Box)
`type` identifies the box type; standard boxes use a compact type, which is normally four printable characters, to permit ease of identification, and is shown so in the boxes below. User extensions use an extended type; in this case, the type field is set to 'uuid'.

Boxes with an unrecognized type shall be ignored and skipped.
Many objects also contain a version number and flags field:

```
aligned(8) class FullBox(unsigned int(32) boxtype, unsigned int(8) v, bit(24) f)
                  extends Box(boxtype) {
      unsigned int(8) version = v;
      bit(24) flags = f;
}
```

The semantics of these two fields are:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

`version` is an integer that specifies the version of this format of the box.
`flags` is a map of flags

Boxes with an unrecognized version shall be ignored and skipped.

Item Location Box

Definition

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

The item location box provides a directory of resources in this or other files, by locating their containing file, their offset within that file, and their length. Placing this in binary format enables common handling of this data, even by systems which do not understand the particular metadata system (handler) used. For example, a system might integrate all the externally referenced metadata resources into one file, re-adjusting file offsets and file references accordingly.

The box starts with three values, specifying the size in bytes of the offset field, length field, and base_offset field, respectively. These values must be from the set {0, 4, 8}.

Items may be stored fragmented into extents, e.g. to enable interleaving. An extent is a contiguous subset of the bytes of the resource; the resource is formed by concatenating the extents. If only one extent is used (extent_count = 1) then either or both of the offset and length may be implied:

- If the offset is not identified (the field has a length of zero), then the beginning of the file (offset 0) is implied.
- If the length is not specified, or specified as zero, then the entire file length is implied. References into the same file as this metadata, or items divided into more than one extent, should have an explicit offset and length, or use a MIME type requiring a different interpretation of the file, to avoid infinite recursion.

The size of the `item` is the sum of the extent_lengths.

Note: extents may be interleaved with the chunks defined by the sample tables of tracks.

The data-reference index may take the value 0, indicating a reference into the same file as this metadata, or an index into the data-reference table.

Some referenced data may itself use offset/length techniques to address resources within it (e.g. an MP4 file might be 'included' in this way). Normally such offsets are relative to the beginning of the containing file. The field 'base offset' provides an additional offset for offset calculations within that contained data. For example, if an MP4 file is included within a file formatted to this specification, then normally data-offsets within that MP4 section are relative to the beginning of file; base_offset adds to those offsets.

Syntax

```
aligned(8) class ItemLocationBox extends FullBox('iloc', version = 0, 0) {
      unsigned int(4) offset_size;
      unsigned int(4) length_size;
      unsigned int(4) base_offset_size;
      unsigned int(4) reserved;
      unsigned int(16) item_count;
      for (i=0; i<item_count; i++) {
            unsigned int(16) item_ID;
            unsigned int(16) data_reference_index;
            unsigned int(base_offset_size*8) base_offset;
            unsigned int(16) extent_count;
            for (j=0; j<extent_count; j++) {
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                unsigned int(offset_size*8) extent_offset;
                unsigned int(length_size*8) extent_length;
            }
        }
}
```

Semantics

`offset_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the offset field.
`length_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the length field.
`base_offset_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the base_offset field.
`item_count` counts the number of resources in the following array.
`item_ID` is an arbitrary integer 'name' for this resource which can be used to refer to it (e.g. in a URL).
`data-reference-index` is either zero ('this file') or a 1-based index into the data references in the data information box.
`base_offset` provides a base value for offset calculations within the referenced data. If base_offset_size is 0, base_offset takes the value 0, i.e. it is unused.
`extent_count` provides the count of the number of extents into which the resource is fragmented; it must have the value 1 or greater
`extent_offset` provides the absolute offset in bytes from the beginning of the containing file, of this item. If offset_size is 0, offset takes the value 0
`extent_length` provides the absolute length in bytes of this metadata item. If length_size is 0, length takes the value 0. If the value is 0, then length of the item is the length of the entire referenced file.

Item Information Box

Definition

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

The Item information box provides extra information about selected items, including symbolic ('file') names. It may optionally occur, but if it does, it must be interpreted, as item protection or content encoding may have changed the format of the data in the item. If both content encoding and protection are indicated for an item, a reader should first un-protect the item, and then decode the item's content encoding. If more control is needed, an IPMP sequence code may be used.

This box contains an array of entries, and each entry is formatted as a box. This array is sorted by increasing item_ID in the entry records.

Syntax

```
aligned(8) class ItemInfoEntry
            extends FullBox('infe', version = 0, 0) {
      unsigned int(16) item_ID;
      unsigned int(16) item_protection_index
      string item_name;
      string content_type;
      string content_encoding; //optional
}
aligned(8) class ItemInfoBox
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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

Semantics

`item_id` contains either 0 for the primary resource (e.g. the XML contained in an 'xml ' box) or the ID of the item for which the following information is defined.
`item_protection_index` contains either 0 for an unprotected item, or the one-based index into the item protection box defining the protection applied to this item (the first box in the item protection box has the index 1).
`item_name` is a null-terminated string in UTF-8 characters containing a symbolic name of the item;
`content_type` is the MIME type for the item.
`content_encoding` is an optional null-terminated string in UTF-8 characters used to indicate that the binary file is encoded and needs to be decoded before interpreted. The values are as defined for Content-Encoding for HTTP /1.1. Some possible values are "gzip", "compress" and "deflate". An empty string indicates no content encoding.
`entry_count` provides a count of the number of entries in the following array.

Item Protection Box

Definition

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

The item protection box provides an array of item protection information, for use by the Item Information Box.

Syntax

```
aligned(8) class ItemProtectionBox
            extends FullBox('ipro', version = 0, 0) {
      unsigned int(16) protection_count;
      for (i=1; i<=protection_count; i++) {
            ProtectionSchemeInfoBox protection_information;
      }
}
```

Semantics

`protection_count` is the number of protection tools applied
`protection_information` contains the information for each protection tool applied

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.040

Price based on 31 pages

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale