INTERNATIONAL STANDARD ISO/IEC 14496-12:2012
TECHNICAL CORRIGENDUM 1

Published 2013-09-01

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Coding of audio-visual objects —

Part 12:
ISO base media file format

TECHNICAL CORRIGENDUM 1

Technologies de l'information — Codage des objets audiovisuels —

Partie 12: Format ISO de base pour les fichiers médias

RECTIFICATIF TECHNIQUE 1

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

Add the following to 8.3.2.1, at the end:

The duration field here does not include the duration of following movie fragments, if any, but only of the media in the enclosing Movie Box. The Movie Extends Header box may be used to document the duration including movie fragments, when desired and possible.

ICS 35.040 **Ref. No. ISO/IEC 14496-12:2012/Cor.1:2013(E)**

Published in Switzerland

Adjust 8.11.3 to read as follows (maintaining the correct section numbers):

8.11.3 The Item Location Box

8.11.3.1 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 container, their offset within that container, 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 place, re-adjusting offsets and references accordingly.

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

The `construction_method` field indicates the 'construction method' for the item:

i) file_offset: by the usual absolute file offsets into the file at data_reference_index; (construction_method == 0)

ii) idat_offset: by box offsets into the idat box in the same meta box; neither the data_reference_index nor extent_index fields are used; (construction_method == 1)

iii) item_offset: by item offset into the items indicated by the extent_index field, which is only used (currently) by this construction method. (construction_method == 2).

The extent_index is only used for the method item_offset; it indicates the 1-based index of the item reference with referenceType 'iloc' linked from this item. If index_size is 0, then the value 1 is implied; the value 0 is reserved.

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 source (offset 0) is implied.
- If the length is not specified, or specified as zero, then the entire length of the source 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 offsets are relative to a data origin. That origin is determined as follows:

1) when the construction_method specifies a file offset, the data origin is the beginning of the file identified by the data reference;

2) when the construction_method specifies offsets into the Item Data box, the data origin is the beginning of data[] in the Item Data box;

3) when the data reference specifies another item, the data origin is the first byte of the concatenated data (of all the extents) of that item;

Note – There are offset calculations in other parts of this file format based on the beginning of a box header; in contrast, item data offsets are calculated relative to the box contents.

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 in the item itself 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; the base offset adds to those offsets.

If an item is constructed from other items, and those source items are protected, the offset and length information apply to the source items after they have been de-protected. That is, the target item data is formed from unprotected source data.

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

8.11.3.2 Syntax

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

8.11.3.3 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.
`index_size` is taken from the set {0, 4, 8} and indicates the length in bytes of the `extent_index` 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).
`construction_method` is taken from the set 0 (file), 1 (idat) or 2 (item)

`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_index` provides an index as defined for the construction method
`extent_offset` provides the absolute offset, in bytes from the data origin of the container, of this extent data. If `offset_size` is 0, extent_`offset` takes the value 0
`extent_length` provides the absolute length in bytes of this metadata item extent. If `length_size` is 0, extent_`length` takes the value 0. If the value is 0, then length of the extent is the length of the entire referenced container.

In 8.16.4.1, change the third paragraph from

Each byte in the subsegment shall be assigned to a level. If the range is not associated with any information in the level assignment, then any level that is not included in the level assignment may be used.

There shall be 0 or 1 Subsegment Index boxes per each Segment Index box that indexes only leaf subsegments, i.e. that only indexes subsegments but no segment indexes. A Subsegment Index box, if any, shall be the next box after the associated Segment Index box. A Subsegment Index box documents the subsegment that is indicated in the immediately preceding Segment Index box.

to

Each byte in the subsegment shall be explicitly assigned to a level, and hence the range count must be 2 or greater. If the range is not associated with any information in the level assignment, then any level that is not included in the level assignment may be used.

There shall be 0 or 1 Subsegment Index boxes per each Segment Index box that indexes only leaf subsegments, i.e. that only indexes subsegments but no segment indexes. A Subsegment Index box, if any, shall be the next box after the associated Segment Index box. A Subsegment Index box documents the subsegments that are indicated in the immediately preceding Segment Index box.

In 8.16.4.2 change

```
aligned(8) class SubsegmentIndexBox extends FullBox('ssix', 0, 0) {
   unsigned int(32)  subsegment_count;
   for( i=1; i <= subsegment_count; i++)
   {
      unsigned int(32)  ranges_count;
      for ( j=1; j <= range_count; j++) {
         unsigned int(8) level;
         unsigned int(24) range_size;
      }
   }
}
```

to