DRAFT AMENDMENT

ISO/IEC 23001-14:2019 DAM 1

ISO/IEC JTC **1**/SC **29** Secretariat: **JISC**

Voting begins on:
2021-01-21

Voting terminates on:
2021-04-15

Information technology — MPEG systems technologies —

Part 14: **Partial file format**

AMENDMENT 1: Support for HTTP entities, enhanced file type and byte-range priorities

Partie 14: Format de fichier partiel

AMENDEMENT 1: Titre manque

ICS: 35.040.40

THIS DOCUMENT IS A DRAFT CIRCULATED FOR COMMENT AND APPROVAL. IT IS THEREFORE SUBJECT TO CHANGE AND MAY NOT BE REFERRED TO AS AN INTERNATIONAL STANDARD UNTIL PUBLISHED AS SUCH.

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.

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.

This document is circulated as received from the committee secretariat.

Reference number
ISO/IEC 23001-14:2019/DAM 1:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Information Technology — MPEG Systems Technologies— Part 14: Partial File Format, AMENDMENT1: Support for HTTP entities, enhanced file type and byte-range priorities

In section 4.2.4 of, replace:

"The FileTypeBox of the source file, whether or not correctly received, shall not be modified; it shall be encapsulated in one partial segment and stored in a PartialDataBox, in order to ensure unicity of the FileTypeBox at the container level."

with

"The FileTypeBox of the source file, and, if present, the OriginalFileTypeBox and/or ExtendedTypeBox, whether or not correctly received, shall not be modified; they shall either:

- be encapsulated altogether in one partial segment and stored in a PartialDataBox, or
- be encapsulated altogether in an OriginalFileTypeBox immediately following the new FileTypeBox of the mixed partial file.

This ensures that a single FileTypeBox is present at the container level."

Add new subclause 5.1.9

5.1.9 HTTP Entity Box

5.1.9.1 Definition

Box Type: 'htte'
Container: PartialSegmentBox or PartialFileBox
Mandatory: No
Quantity: At most one per PartialSegmentBox, or one in PartialFileBox

The HTTPEntityBox is used to store a set of HTTP entities (header name and body) applying to the file identified by the source URL. It is typically inserted in PartialFileBox or PartialSegmentBox by the receiver based on information carried in the delivery protocol, and can be used by the receiving entity to populate an HTTP cache.

There may be several HTTPEntityBox in a partial file. HTTPEntityBox declared in PartialFileBox define entities valid for the entire partial file; HTTPEntityBox declared in PartialSegmentBox define entities valid for the partial segment only.

The Content-Type and Content-Length entities shall not be included in this box. Content-Type may be signalled through SourceURLBox. Content-Length shall be recomputed from the partial file structures (chunks in PartialSegmentLocationBox).

5.1.9.2 Syntax

```
aligned(8) class HTTPEntityBox extends FullBox('htte', 0, 0) {
   unsigned int(32) entry_count;
   for (i=0; i<entry_count; i++) {
      utf8string name;
      utf8string body;
   }
}
```

5.1.9.3 Semantics

entry_count indicates the number of HTTP entities in the box.
name gives the name of the HTTP entity described.
body gives the body (content) of the HTTP entity described.

Add new subclause 5.1.10

5.1.10 Byte-Range Priority Info Box

5.1.10.1 Definition

Box Type: 'brpi'
Container: PartialSegmentBox or PartialFileBox
Mandatory: No
Quantity: At most one per PartialSegmentBox, or one in PartialFileBox

The ByteRangePriorityInfoBox indicates transmission priority levels of byte ranges in the source file. This allows a file reader to further optimize its repair process. By using external data reference in a partial file, it is possible to build a companion file containing priority levels of byte ranges of the source file, allowing a server to optimize its distribution (retransmission policies, FEC, ...) of a file without modifying it.

NOTE This information is usually transported out-of-band or through well-protected packets with more FEC in the transport layer; for example, the information can be described in the FDT of the file in a FLUTE session.

If this box is present in the PartialFileBox, it shall indicate the byte ranges priorities for the complete file using absolute offsets, and no other ByteRangePriorityInfoBox shall be present in any subsequent PartialSegmentBox.

The following flags are defined for the ByteRangePriorityInfoBox:

- relative_offset: flag value is 0x000001. Presence of this flag indicates that indicated byte ranges are relative to the first byte of the first chunk of the partial segment containing this box. Absence of this flag indicates that indicated byte ranges are relative to the beginning (first byte) of the source file. This flag shall not be set if the container box is a PartialFileBox.
- dependencies_present: flag value is 0x000002. Presence of this flag indicates that the priority level depends on an explicit list of priority levels, rather than on levels with lower priority.

Byte in the source file not included in any of the byte ranges listed in ByteRangePriorityInfoBox shall be treated as having priority 0.

5.1.10.2 Syntax

```
aligned(8) class ByteRangePriorityInfoBox extends FullBox('brti', version, flags)
{
   unsigned int(32) entry_count;
   for (i=0; i < entry_count; i++) {
      if (version==1) {
         unsigned int(64) byte_range_start;
      } else {
         unsigned int(32) byte_range_start;
      }
      unsigned int(32) byte_range_length;
      unsigned int(16) priority_level;
      if (flags & dependencies_present) {
         unsigned int(16) num_dependencies;
         for (i=0; i<num_dependencies; i++) {
            unsigned int(16) depends_on_level;
         }
      }
   }
}
```

5.1.10.3 Semantics

entry_count is the number of index points listed in this box.

byte_range_start specifies the start of the byte range of the index in the source file. If version 1 is used, 64 bits data offsets are used; otherwise 32 bits data offsets are used.

byte_range_length specifies the size in bytes of the byte range.

priority_level specifies the priority level of that byte range. A value of 0 indicates the highest priority. Repair or sending operation can be prioritize based on this value.

num_dependencies indicates the number of explicit dependencies for this level. If 0 or not present, this indicates that byte ranges with priority in [0, priority_level-1] inclusive are required to process the byte range if priority_level is not 0, or that no additional byte ranges are needed if priority_level is 0. If present and not 0, this indicates the number of dependent levels required to process this byte range.

depends_on_level indicates the priority level of each byte range depended on. The dependencies should be ordered by increasing value of levels.