INTERNATIONAL STANDARD

ISO/IEC 23001-18

First edition
2022-06

Information technology — MPEG systems technologies —

Part 18: Event message track format for the ISO base media file format

Technologies de l'information — Technologies des systèmes MPEG —

Partie 18: Format de la piste du message d'événement pour le format ISO de base pour les fichiers médias

Reference number
ISO/IEC 23001-18:2022(E)

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

Published in Switzerland

Contents

Page

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

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

A list of all parts in the ISO/IEC 23001 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Information technology — MPEG systems technologies —

Part 18:
Event message track format for the ISO base media file format

1 Scope

This document specifies the format of the event message track.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

ISO/IEC 23009-1, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

3 Terms, definitions, abbreviated terms and notations

3.1 Terms and definitions

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

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
active event
event (3.1.2) with a start time up to start time plus duration overlapping the current media time

3.1.2
event
aperiodic sparse metadata information that is intended for a specific interval of media-time

3.1.3
event duration
duration when the *event* (3.1.2) is applicable

3.1.4
event instance
representation of the *event* (3.1.2) in the track represented by an event message instance box

3.1.5
event start time
start of the interval in media-time when event information is applicable

3.1.6
event message track
ISO base media file format timed metadata track that carries event messages

3.2 Abbreviated terms

DASH	dynamic adaptive streaming over HTTP, as specified in ISO/IEC 23009-1
ISO-BMFF	ISO base media file format, as specified in ISO/IEC 14496-12

3.3 Notation

For the purposes of this document, the notation used in ISO/IEC 23009-1 applies.

4 General

The `DASHEventMessageBox`, defined in ISO/IEC 23009-1, is a box structure that may be used in streaming applications. Since it is a top-level box and has its own timeline, it is tricky to work within ISO-BMFF formatted media files, such as those based on the common media application format (CMAF) (see ISO/IEC 23000-19). For instance, identifying active events at any point in the media track may require scanning of a large part of that track. In addition, it is not clear what happens to `DASHEventMessageBox`es when tracks are de-fragmented. Further, `DASHEventMessageBox`es cannot be de-multiplexed from track files based on ISO/IEC 23009-1 or ISO/IEC 23000-19.

This document specifies the event message track format for the carriage of event messages in tracks of the ISO base media file format (ISO-BMFF) ISO/IEC 14496-12. This event message track format associates the timeline of the newly defined event message instance box to the track timeline. The specified track format enables all common ISO-BMFF processing such as multiplexing and de-fragmentation. In addition, multiplexing and de-multiplexing operations using top-level `DASHEventMessageBox` based on this event message track format are defined. The carriage in the event message track format makes this information more easily accessible to devices that can seek through ISO-BMFF formatted media files.

The event message track format is defined in the following clauses.

5 Background

In CMAF track files and DASH segments `DASHEventMessageBox`es may occur as top-level boxes as defined in ISO/IEC 23000-19 and ISO/IEC 23009-1. The `DASHEventMessageBox` arises in two versions as defined in ISO/IEC 23009-1:

```
aligned(8) class DASHEventMessageBox
      extends FullBox('emsg', version, flags = 0){
if (version==0) {
   string scheme_id_uri;
   string              value;
     unsigned int(32)  timescale;
   unsigned int(32)  presentation_time_delta;
   unsigned int(32)  event_duration;
   unsigned int(32)  id;
}
else if (version==1) {
    unsigned int(32)      timescale;
    unsigned int(64)   presentation_time;
    unsigned int(32)      event_duration;
     unsigned int(32)      id;
    string                scheme_id_uri;
    string              value;
}
unsigned int(8)   message_data[];
}
```

The semantics are defined in ISO/IEC 23009-1.

6 Event message track structure

6.1 EventMessageInstanceBox

6.1.1 Definition

The EventMessageInstanceBox is defined to enable carriage of the information contained in DASHEventMessageBoxes in tracks. This box is very similar to the DASHEventMessageBox defined in ISO/IEC 23009-1 but instead occurs only in samples of a track.

6.1.2 Syntax

```
aligned(8) class EventMessageInstanceBox
      extends FullBox('emib', version, flags = 0){
   unsigned int(32)      reserved = 0;
   signed int(64)        presentation_time_delta;
   unsigned int(32)      event_duration;
   unsigned int(32)      id;
   string                scheme_id_uri;
   string                value;
   unsigned int(8)       message_data[];
}
```

6.1.3 Semantics

The semantics of all fields except the following are defined in ISO/IEC 23009-1 for the DASHEventMessageBox:

— presentation_time_delta provides the event start time on the media presentation timeline relative to the presentation time of the sample enclosing the EventMessageInstanceBox.

— The presentation_time_delta and event_duration values are in the number of ticks in the timescale defined in the track's MediaHeaderBox.

NOTE A DASHEventMessageBox can be converted to one or more EventMessageInstanceBoxes to enable carriage in ISO-BMFF samples based on this document. An example algorithm is given in 9.2.

6.2 EventMessageEmptyBox

```
aligned(8) class EventMessageEmptyBox extends Box('emeb')
{
}
```

The EventMessageEmptyBox is defined to signal duration when no event is active, with the following semantics. The box is empty, and the duration is defined by the sample enclosing the EventMessageEmptyBox.

NOTE Using zero-size samples to signal when no event metadata or event is active is avoided because this can cause problems in some devices.

7 Event message track format

7.1 Track format

EventMessageInstanceBoxes shall be carried in ISO-BMFF timed metadata tracks as defined in ISO/IEC 14496-12. As a consequence, use the 'meta' media handler type, and the associated null media header. EventMessageInstanceBoxes are carried in samples of the track.

Event message tracks shall not use any composition time offset, thus composition and decoding times are identical.

7.2 Sample entry format

Event message tracks shall use the `EventMessageSampleEntry` format.

```
class EventMessageSampleEntry() extends MetaDataSampleEntry ('evte'){
   BitRateBox ;    // optional
   SchemeIdListBox ;    // optional
}
```

7.3 Scheme identifier list box

7.3.1 Definition

The event schemes that may appear in the track may be specified using `SchemeIdListBox`. This box is optional and if present, it specifies the event schemes in the track, whether each scheme appears at least once in the track, and/or whether the provided schemes are complete, i.e. no other scheme may appear in the track.

7.3.2 Syntax

```
aligned(8) class SchemeIdListBox
   extends FullBox('silb', version, flags = 0){
   unsigned int(32)      number_of_schemes;
   for(int i=1; i<number_of_schemes; i++){
      utf8String   scheme_id_uri;
      utf8String   value;
      bit(7)     reserved=0;
      bit(1)   atleast_once_flag=0;
   }
   bit(7)   reserved;
   bit(1)   other_schemes_flag;
}
```

7.3.3 Semantics

`number_of_schemes` is the number of schemes listed in this box.

`scheme_id_uri` is a NULL-terminated C string declaring either the identifier of the scheme, if no value follows, or the identifier of the naming scheme for the following value.

`value` is a name from the declared scheme.

`atleast_once_flag` is where, if it is set to TRUE, the track contains at least one event instance of this scheme.

`other_schemes_flag` is where, if it is set to TRUE, the track may contain other schemes other than the ones listed in this box.

7.4 Sample format

Each ISO-BMFF sample in the track shall contain either:

a) one or more `EventMessageInstanceBox`es;

b) a single `EventMessageEmptyBox`; in this case, the sample contains no `EventMessageInstanceBox`es, and no events are active during the sample presentation time.

NOTE There is no required ordering of the `EventMessageInstanceBox`es in the sample.

All instances of the same event shall contain identical values for all fields except the `presentation_time_delta`. Each instance is documented with one `EventMessageInstanceBox` in the track. Instances of the same event in the track are detected by using the conditions as defined in ISO/IEC 23009-1 for detecting duplicate event messages.

Each `EventMessageInstanceBox` documents an event message which has or will have an active interval. If the media presentation time of the containing sample is `T`, the active interval is defined to run: from `(T + presentation_time_delta)` to, but not including `(T + presentation_time_delta + event_duration)`.

Each event has an event duration, represented by the `event_duration` field. Each sample in the track also has a duration D.

Each sample shall contain all events that have an active interval that overlaps the sample's time interval [T, T+D). A sample may also contain instances of other events that become active after T+D, i.e. future events.

7.5 Codecs parameter

The `'codecs'` parameter for event tracks is as defined in IETF RFC 6381 with the following specifics. The first element is the four-character code of the sample description entry of the event message sample entry `'evte'`. Additional optional elements are reserved and may be defined in future revisions.

8 Timing constraints

a) Any sample with a presentation time and duration shall contain all `EventMessageInstanceBox`es active during the timespan from the presentation time up to but not including presentation time plus duration. Consequently, `EventMessageInstanceBox`es with a duration extending multiple samples are carried in each of these samples.

b) If the sample is the first sample containing one or more specific instances of `EventMessageInstanceBox`, the `EventMessageInstanceBox.presentation_time_delta` should not be negative. However, if prior samples or parts of the track are not available, `EventMessageInstanceBox.presentation_time_delta` may be negative to reflect the accurate presentation time of the event.

c) A change in the set of active events shall trigger a sample boundary, with a matching presentation time, i.e. a new sample is introduced anytime an event is starting or ending.

d) If the `EventMessageInstanceBox.duration` is zero or unknown, the sample duration shall not be zero, and may be, for example, a small value such as a single tick on the timescale.

e) One or more samples carrying `EventMessageEmptyBox` should be used to cover timespans where no event is active. If not the case, the sample contains one or more `EventMessageInstanceBox`(es) that will become active or were active in the past.

9 Processing

9.1 Client processing

The semantics of the `EventMessageInstanceBox` are used when processing the samples. Briefly summarized they are as follows:

a) The fields `scheme_id_uri` and `value` signal the scheme and subscheme of the message, respectively.

b) The identical values of `id`, `scheme_id_uri` and `value` between two or more `EventMessageInstanceBox`es are used to detect duplicate event messages.

c) The value `presentation_time_delta` signals the relative presentation time of the event compared to the ISO-BMFF sample presentation time.

d) The value `event_duration` signals the actual duration of the event in ticks (which may be longer than sample duration), and may also signal indefinite duration.

e) The value `message_data` contains the binary payload of the message.

The timed metadata samples can be acquired, and their event messages are processed and then passed to the application using a client processing model such as the one defined by the DASH processing model for events and timed metadata defined in ISO/IEC 23009-1.

9.2 Conversion of `DASHEventMessageBox to EventMessageInstanceBox`

9.2.1 General

This subclause documents one algorithm for placing a sequence of event messages into samples.

This algorithm operates in the following steps:

a) Version 0 `DASHEventMessageBox`es are logically converted to version 1, by calculating their presentation times.

b) The presentation times and duration values are converted to be based on the common timescale of the track.

c) A set of samples is defined; a new sample occurs at each time that the set of active events changes.

d) New event instances are inserted, and possibly samples are split to carry those instances, for warning (possible future events) and recovery (probable past events).

The events are considered in time order, and a new sample occurs at each time that set of active events changes. Within each sample, there is an instance for every event whose active interval overlaps the time interval of the sample. The `presentation_time_delta` of each instance is set to the presentation time of the event minus the sample presentation time. All other fields with identical names are copied from `DASHEventMessageBox` to `EventMessageInstanceBox`.

The algorithm for step 3 is given in Example 2 (see 9.2.3), using the routine in Example 1 (see 9.2.2). The algorithm takes the input list `DASHEventMessageBoxesV1` and the target track or segment boundaries T and T+D. Each event starting or ending within the segment boundaries results in a sample boundary. In cases where the `event.event_duration` is zero, a sample boundary is introduced one tick after the `event.presentation_time`. The sort_unique() routine orders sample boundaries in ascending order and removes duplicates.

Example 2 (see 9.2.3) illustrates the routine to find the contents of each sample. For each current and next sample boundary computed using the routine from Example 1 (see 9.2.2) it is checked if the event is active within these boundaries. If the event is active, it is converted to `EventMessageInstanceBox` using the original `Event` and the presentation time of the first sample boundary. The result is appended to the list of event message instance boxes of the sample. The `EventSample` structure contains the sample presentation time, the sample duration and the list of event message instance boxes to be carried in the sample. If the list of instance boxes is empty, the `EventMessageEmptyBox` is enclosed.

NOTE This conversion algorithm can be used per track spanning T to T+D or per segment spanning T to T+D. By doing the conversion per segment, live and dynamic cases can be supported, generating a new segment of the event track each update in time.

9.2.2 Example 1: Algorithm for finding sample boundaries in a segment of [T,T+D)

```
Algorithm find_sample_boundaries
Input(DASHEventMessageBoxesV1, T, T+D)
Output(list_of_sample_boundaries)

list_of_sample_boundaries.append(T);
list_of_sample_boundaries.append(T+D);

for event in DASHEventMessageBoxesV1 {
   pt = event.presentation_time;
   pt_du = event.presentation_time + event.event_duration
   if event.event_duration == 0
       pt_du = event.presentation_time + 1;
```

```
   if pt >= T and  pt < T + D
       list_of_sample_boundaries.append(pt);
   if pt_du >= T and pt_du < T + D
       list_of_sample_boundaries.append(pt_du);

}
list_of_sample_boundaries.sort_unique()
```

9.2.3 Example 2: Algorithm for finding the samples and the sample contents

```
Algorithm find_sample_contents
Input(DASHEventMessageBoxesV1, T, T+D)
Output(samples_content)

sample_boundaries= find_sample_boundaries(DASHEventMessageBoxesV1,T,T+D)

for current_boundary,next_boundary in sample_boundaries {
  EventSample s;
  s.sample_presentation_time = boundary;
  s.sample_duration = next_boundary - boundary;
  s.is_empty = True;
    for event in DASHEventMessageBoxesV1 {
         pt = event.presentation_time;
         pt_du = pt + event.event_duration;

         if (event.event_duration == 0)
           pt_du = pt + 1;
         if (pt < next_boundary and pt_du > current_boundary){
             s.instance_boxes.append(EventMessageInstanceBox(
                     (event, s.sample_presentation_time));
             s.is_empty = False;
         }

     }
     samples_content.append(s);
}
```

9.3 Track conversion

9.3.1 General

The following processing is defined in this subclause:

— de-multiplexing a CMAF track file with `DASHEventMessageBoxes` to an event message track;

— multiplexing of an event message track with a CMAF track file resulting in a CMAF track file with `DASHEventMessageBox`es;

— fragmentation and de-fragmentation of an event message track.

9.3.2 De-multiplex a CMAF track file with `DASHEventMessageBoxes`

9.3.2.1 General

The following steps convert a CMAF track file with `DASHEventMessageBox`es (e.g. a CMAF file based on ISO/IEC 23000-19) to a separate event message track. The algorithm is shown in Example 3 (see 9.3.2.2). All events are extracted from the CMAF track, and then the samples are computed based on the `find_sample_contents` defined in Example 2 (see 9.2.3).

NOTE 1 The timescale of `DASHEventMessageBox` in a CMAF track file equals the timescale defined in the `MediaHeaderBox`.

NOTE 2 For cases where events do not overlap or have zero/indefinite duration, the processing model is simplified.

NOTE 3 There are no strict rules for the ordering of DASHEventMessageBox in a CMAF track file. This is why the entire track file is scanned first in this processing model.

9.3.2.2 Example 3: De-multiplex a CMAF track file with DASHEventMessageBoxes

```
Input(CMAF Track File or segment in_cmaf) Output(track out_event_message)

  1. Initialize out_event_message as an Event Message Track
  2. Initialize a set of Event_Boxes to hold DASHEventMessageBoxes
  3. Read in_cmaf fragment by fragment,
  4. In each case a DASHEventMessageBox is detected, do:
        a. If version 0 DASHEventMessageBox, convert it to version 1
        b. Add the DASHEventMessageBox to Event_Boxes
        c. In case the SchemeIdListBox is used, include the emsg.scheme_id_uri and emsg.
value to the box if that pair is not already included, continue to 3)
  5. samples = find_sample_contents(Event_Boxes,in_cmaf.start,in_cmaf.end)
  6. write samples to out_event_message
```

9.3.3 Multiplex a CMAF track file with DASHEventMessageBoxes

9.3.3.1 General

A CMAF track (see ISO/IEC 23000-19) can contain one or more DASHEventMessageBoxes. This clause describes the processing of inserting DASHEventMessageBox in a CMAF track using an event message track, as shown in Example 4 (see 9.3.3.2). The announce_time is defined to be the amount of time a DASHEventMessageBox is included before the earliest presentation time of the fragment it applies to.

NOTE The variable announce_time is introduced for this processing operation. It is an approximation of the time in advance a DASHEventMessageBox is inserted.

In this case, it is assumed that both the CMAF track file and the event message track have a common timeline (the timeline origin and timescale). The content of the metadata track is multiplexed in the CMAF track file using DASHEventMessageBoxes.

9.3.3.2 Example 4: De-multiplex a CMAF track file with DASHEventMessageBoxes

```
Input (CMAF Track File in_cmaf, EventTrack in_event, announce_time, target_version)

Output(CMAF Track file out_cmaf)

  a) Initialize out_cmaf

For each CMAF Fragment frag in in_cmaf

  b) Load the earliest presentation time (frag.ept) and duration (frag.duration) of the
fragment. Calculate the fragment end time (frag.end).
  c) Load samples from the in_event corresponding to time interval starting at the
earliest presentation time of the fragment and ending at the calculated end time of the
fragment. Extract all EventMessageInstanceBoxes enclosed in these samples and convert them
to one or more DASHEventMessageBoxes. This conversion results in a version 1 if target_
version=1, and version 0 if target_version=0. Insert the DASHEventMessageBoxes in front of
the MovieFragmentBox of frag if present.
  d) Copy frag including DASHEventMessageBoxes if present, to out_cmaf
```

NOTE This processing model can lead to more identical DASHEventMessageBoxes being inserted in front of different fragments. Duplicate DASHEventMessageBoxes can be detected with identical value, id and schemeIdUrivalues between two or more DASHEventMessageBoxes. Alternative processing to avoid duplicate event message boxes can be defined. However, the duplication of event messages can be useful for a redundant delivery of events.

9.3.4 Fragmentation and de-fragmentation of event message tracks

All samples in the event message track are sync samples. Event message tracks can therefore be fragmented as desirable. The fragmentation can be useful for some delivery schemes or protocols to

