Digital Video Broadcasting (DVB); Guidelines for the Use of the DVB File Format Specification for the Storage and Playback of DVB Services

DVB Document A158

March 2011


DVB
Digital Video
Broadcasting

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for ETSI members and non-members, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by Joint Technical Committee (JTC) Broadcast of the European Broadcasting Union (EBU), Comité Européen de Normalisation ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI).

NOTE: The EBU/ETSI JTC Broadcast was established in 1990 to co-ordinate the drafting of standards in the specific field of broadcasting and related fields. Since 1995 the JTC Broadcast became a tripartite body by including in the Memorandum of Understanding also CENELEC, which is responsible for the standardization of radio and television receivers. The EBU is a professional association of broadcasting organizations whose work includes the co-ordination of its members' activities in the technical, legal, programme-making and programme-exchange domains. The EBU has active members in about 60 countries in the European broadcasting area; its headquarters is in Geneva.

European Broadcasting Union
CH-1218 GRAND SACONNEX (Geneva)
Switzerland
Tel: +41 22 717 21 11
Fax: +41 22 717 24 81

The Digital Video Broadcasting Project (DVB) is an industry-led consortium of broadcasters, manufacturers, network operators, software developers, regulatory bodies, content owners and others committed to designing global standards for the delivery of digital television and data services. DVB fosters market driven solutions that meet the needs and economic circumstances of broadcast industry stakeholders and consumers. DVB standards cover all aspects of digital television from transmission through interfacing, conditional access and interactivity for digital video, audio and data. The consortium came together in 1993 to provide global standardisation, interoperability and future proof specifications.

Introduction

This present document provides guidelines for the use of the specification TR 102 833 “DVB File Format Specification for the Storage and Playback of DVB Services”, known as the DVB File Format Specification.

The guidelines in this document are intended to provide guidance both on how files should be created in a fashion compatible with the specification and how the files should be parsed and processed when their content is presented to the user. These guidelines do not provide an exhaustive coverage of the functionality in the DVB File Format Specification, but concentrate on certain key aspects to maximise interoperability and assist in supporting key functionality, and are intended to be used as recommended practices.

These Guidelines, as with the DVB File Format Specification, build on the ISO File Format Specification and as such many guidelines that are applicable to ISO File Format Specification will apply to DVB files, especially where brand compatibility is to be achieved.

These guidelines are informative, and as such do not constraint actual implementations, either where additional features are provided, or aspects of the specification not discussed in this present document are utilised.

This document should be read in conjunction with the following specifications:

- The ISO Base Media File Format [i.3], as this is the basis of the file format
- The DVB File Format [i.1], to which these guidelines related
- The CPCM specification [i.13], and its guidelines [i.16], which includes recommendations on the usage of CPCM which can be used together with DVB files, where interaction with CPCM is expected.

1 Scope

The present document provides guidelines on the use of the DVB File Format [i.1]. As such these guidelines should apply to the creation or processing of any file that is branded, or co-branded, as any of the DVB brands.

The main scope of this present document is to provide assistance in the implementation of the DVB File Format for both the creation and processing of DVB files. This assistance takes the form of recommendations on modes of operation, features to be supported and expected modes of operation.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references,only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

2.1 Informative references

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 102 833: "Digital Video Broadcasting (DVB); File Format Specification for the Storage and Playback of DVB Services".

[i.2] ETSI TS 126 244: “3GPP file format (3GP) (Release 9)” (3GPP TS 26.244)

[i.3] ISO/IEC 14496-12:2008 "Information Technology - Coding of audio-visual objects - Part 12: ISO base media file format".

[i.4] ETSI EN 300 421: "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for 11/12 GHz satellite services".

[i.5] ETSI EN 300 429: "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for cable systems".

[i.6] ETSI EN 300 744: "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial television".

[i.7] ETSI TS 102 034: "Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks".

[i.8] ETSI EN 302 304: "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)".

[i.9] IETF RFC 3550: "RTP: a transport protocol for real-time applications".

[i.10] ETSI EN 301 192: "Digital Video Broadcasting (DVB); DVB specification for data broadcasting"

[i.11] IETF RFC 5117: “RTP Topologies”

[i.12] ISO/IEC 14496-15: "Information technology - Coding of audio-visual objects - Part 15: Advanced Video Coding (AVC) file format".

[i.13] ETSI TR 102 824: “Digital Video Broadcasting (DVB); Content Protection and Copy Management Specification”

[i.14] ISO/IEC 14496-10: "Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding"

[i.15] ISO/IEC 14496-15: "Information technology - Coding of audio-visual objects - Part 15: Advanced Video Coding (AVC) file format".

[i.16] ETSI TR 102 824-12: “Digital Video Broadcasting (DVB); Content Protection and Copy Management Specification; Part 11: CPCM Content Management scenarios”

[i.17] IETF RFC 3711: “The Secure Real-time Transport Protocol (SRTP)”

[i.18] DLNA, “DLNA Guidelines; Volume 2: Media Format Profiles”

3 Definitions, symbols and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ISO-FF and DVB-FF and the following apply:

DVB file: a file authored to the DVB File Format

DVB File Format: the DVB file format specification, as defined in [i.1].

PES decoding time: the media essence decoding time as encoded in the PES DTS header

NOTE: This should not be confused with decoding time, which refers to values encoded in the ISO File’s “`stts`” box.

3.2 Abbreviations

For the purposes of the present document, the abbreviations in ISO-FF and DVB-FF and the following apply:

AVC	Advanced Video Coding
CBR	Constant Bitrate
CPCM	Content Protection and Copy Management, as defined in [i.13]
DECE	Digitial Entertainment Content Ecosystem
DT	Decoding Time
DVB-H	DVB Handheld
HD	High Definition
IDR	Instantaneous Decoder Refresh
MPTS	Multiple Program Transport Stream
NAL	Network Adaptation Layer, as defined in [i.14]
PAT	Program Association Table, as defined in [i.12]
PCR	Program Clock Reference
PDU	Protocol Data Unit
PES	Packetised Elementary Stream, as defined in [i.12]
PIFF	Protected Interoperable File Format
PMT	Program Map Table, as defined in [i.12]
PTS	Presentation Time Stamp, a value encoded in the PES packet header.
RTCP	Real-time Transport Control Protocol as defined in [i.9]
RTP	Real-time Transport Protocol, as defined in [i.9]
SD	Standard Definition
SDP	Session Discovery Protocol
SPTS	Single Program Transport Stream
SRTCP	Secure RTCP as defined in [i.17]
SRTP	Secure Real-time Transport Protocol as defined in [i.17]
TS	Transport Stream, as defined in [i.12]
UDP	User Datagram
WLAN	Wireless Local Area Network

4 Branding

Brands in file formats derived from the ISO base media file format [i.3] are used to label a file and thus indicate the conformance of a file to a specification. Particularly, the brands defined by the DVB file format specification [i.1] serve the purpose of giving an indication that a DVB file format reader can make good use of the file.

DVB has defined two brands:

- The `'dvr1'` brand is mainly intended for environments using native RTP multimedia transport (e.g. DVB-H [i.8]).
- The `'dvt1'` brand is mainly intended for environments traditionally based on MPEG-2 Transport Stream for multimedia delivery (e.g. DVB-T [i.6], DVB-C [i.5], DVB-S [i.4], DVB-IP [i.7]).

A DVB file format reader is required to understand at least one of the DVB file format brands and to implement all features documented for this brand.

The `'dvr1'` and `'dvt1'` brands are supersets of the `'iso3'` brand, which is defined in the third edition of the ISO base media file format specification [i.3].

NOTE: The `'iso3'` brand includes all features of the `'iso2'`, `'isom'`, and the `'avc1'` brand.

4.1 Brand definitions within the DVB context

4.1.1 Major brand

The major brand identifies the best use of a file and is typically defined within the specification used for authoring for the best use. In many cases the major brand relates to the file extension. For DVB files [i.1] the file extension `.dvb` is used and one of the DVB brands (`'dvt1'` or `'dvr1'`) brands shall be used when a file is a DVB file. The DVB major brands are not to be used for indicating conformance to the DVB file format specification. Hence, the major brand must not be used to detect whether a file is a conformant DVB file. Instead, the compatible brands are intended to indicate conformance.

4.1.2 Minor version

The minor version field provides further information on the major brand, which is the version number of the DVB file format specification used for authoring the file in the case a DVB brand being the major brand.

4.1.3 Compatible brands

All brands that are listed as compatible brands provide an indication about conformance to a specification or a particular conformance point. It is possible that a file is conformant to multiple specifications. Then all brands indicating compatibility to these specifications should be listed, so that a reader only understanding a subset of the compatible brands can get an indication that the file can be parsed. A typical example for co-branding is e.g. a file which can be read by 3GPP file format readers and DVB readers.

Compatible brands are also used to indicate permission to read and interpret a file. Intentional omission of a compatible brand is therefore possible, to exclude parsers for a particular specification from consumption of the file.

The major brand shall also be listed as a compatible brand. It is recommended to list the brands defined in [i.1] as compatible brands.

4.2 Co-branding

A DVB file mainly authored for DVB use shall have one of the DVB brands as its major brand and a minor version set to the specification number used for authoring the file. The file extension shall be `.dvb`.

A file that is mainly authored for a different use but is also conforming to the DVB file format specification does not have the `.dvb` file extension but may have one of the DVB brands listed as a compatible brand.

4.2.1 Co-branding with 'dvr1' and 'dvt1'

It is possible to author a file readable by both 'dvr1' and 'dvt1' parsers. However, setting the major brand to either the 'dvr1' or the 'dvt1' brand specifies the best use. A file that conforms to both brands must follow all constraints set by the 'dvr1' and 'dvt1' brands. This requires either

- the presence of at least one media track, contained in both brands,
- or both a reception hint track for RTP and MPEG-2 TS,
- or a reception hint track of one brand and a media track contained in the other brand.

4.2.2 Co-branding with DVB brands and 3GPP brands

The media types supported by the 3GPP file format [i.2] brands and the DVB 'dvr1' brand are overlapping and hence files can be co-branded if both the 'dvr1' and the 3GPP brand constraints are followed. 3GPP defines several brands for different application scenarios, whereas DVB defines only two brands, currently, for the generic MPEG-2 TS based systems and the mobile TV systems based on RTP transport.

4.2.2.1 'dvr1' files without reception hint tracks

Files that contain media tracks but not reception hint tracks can often be co-branded with the DVB 'dvr1' and 3GPP's '3gpX' brands for the 3GPP basic profile. X is a number dependent on the support of specific media formats and file format features as specified in section 5.5 of [i.2].

4.2.2.2 'dvr1' files with reception hint tracks

Files that contain RTP (and optionally RTCP reception hint tracks) can be often co-branded with the DVB 'dvr1' and one or more of the 3GPP '3gtX' brands for the Media Stream Recording Profile (section 5.4.7 of [i.2]). X is a number dependent on the supported 3GPP specification, as specified in section 5.5 of [i.2].

NOTE: The 3GPP file format supports RTP reception hint tracks since Release 8.

The major difference of the 'dvr1' brand to the Media Stream Recording Profile for 3GPP files is the additional support of SRTP and SRTCP reception hint tracks [i.2] of the 3GPP file format, which are not supported by the DVB file format. The DVB file format features (e.g. indexing and metadata) are transparent for a 3GPP reader.

4.2.2.3 'dvt1' files

MPEG-2 TS reception hint tracks are not supported by the 3GPP file format [i.2] and hence co-branding is only possible if any of the other tracks of a file is also supported by a profile of the 3GPP file format.

4.3 Disabled tracks

When the major brand is one of the DVB brands 'dvt1' or 'dvr1', the file was authored according to the DVB file format specification. In this case some media tracks do not need to be supported by a DVB player, which are media tracks that conform to specifications outside the DVB ecosystem and shall hence be marked as disabled in the track header box (section 8.3.2 of [i.3]).

If an alternative media track in a format supported by the DVB ecosystem exists, the track needs not be marked as disabled, since a DVB player can then always select the DVB-supported media track. However, if the track does not have an alternative, then it shall be marked as disabled to accomplish maximum interoperability.

5 General guidelines on ISO boxes

5.1 Box parsing

This clause provides general guidelines for the handling of parsing boxes. As such, this section is primarily of relevance when reading a file.

5.1.1 Box size

The size of a box is part of every box. This size should be checked, and it should not be assumed or inferred from the 4CC that specifies the precise type of box.

When processing sized objects, good defensive programming practices should be used, including ensuring that the size is sensible and does not overflow expected buffers or fail to initialise values.

5.1.2 Box version

If an unknown or unexpected version number is encounted on a box, then the ISO Base Media File Specification definition of a box states that the box shall be skipped.

5.1.2 Unknown boxes

The specification provides a list of potential boxes, however any implementation should cope with the presence of unknown boxes. These may represent future extensions to specifications, boxes required by other specifications which interoperate with the DVB File Format [i.1] as well as proprietary extensions.

At various places a sequential sequence of boxes may be present; an unknown box may occur in any location in a sequence and the remainder of the sequence should be checked for known boxes.

When considering editing a file that contains unknown boxes, there are three options:

- remove unknown boxes and brands before making any edits
- do not make any edits
- take extreme care with edits as you may invalidate the unknown boxes (e.g. if they refer to tracks or specific times, etc).

The ordering of these points represents the preference of the choices. It is recommended that the option to use is selected consistently, and at design time.

When editing a file, a device should ensure all brands that it is aware of are correctly handled in the editing process, and other brands, boxes and tracks which are unknown to the device should be removed.

5.1.3 Box order

Although the DVB File Format [i.1] implies or specifies an order for boxes at various points, it is recommended that as much flexibility over the ordering of parsing of boxes as possible is provided.

5.2 "free" box

When a file is created, there are various points at which boxes that make up that file may grow, either in response to events that occur in the recording stream, or if a new recording is subsequently appended to the file. Examples of such events include a new PMT being received during a transport stream recording, or even the addition of new samples, or the need to insert an "mvex" box when movie fragments become necessary. Whilst it is possible to handle these occasions by inserting the data and moving all subsequence boxes in the file along, this is not an efficient operation.

Instead, when creating a file, it is sensible to reserve to minimise or eliminate the need to move data in the file. Space can be reserved via use of the "free"[1] box structure. When additional space is required the "free" box can be reduced in size, releasing space for use by preceding box(es). The primary location where this functionality is required is in the "moov" box. It is therefore recommended that each container box that is likely to grow, by additional or growing sub-boxes, contains a single "free" box as the last box. The boxes that are likely to grow are the "trak", "stbl" and in certain cases the "dinf" boxes, and sample entry boxes.

When a file is edited and additional space becomes available then a "free" box structure may be used to avoid a complete re-write of the file.

---

[1] The "skip" box is identical and parsers should cope with both forms.

Compaction of a file by shrinking or removal of "free" boxes is permitted; a device shall however ensure that all file offsets in the file are changed accordingly.

5.3 Box growth and chase play

One mode of operation is reading a file that is currently being written, as would occur in "chase play" where the viewer is watching content that is currently recording. The reading device should handle this operation, which implies that the file will change over time, both in the file size and potentially in the boxes present in the file. If movie fragments are not used, the boxes present in the "moov" box will change, and the player should ensure that it handles the possibilities of boxes growing, and their locations changing during play, or the values contained within them changing (e.g. counts).

Where chase play is to be implemented, it is recommended that movie fragments are used. The use of these simplifies the construction and parsing of the file, and significantly reduces the dependence of the parsing process on the creation process. For further discussion of the use of fragments and chase play, see clause 8.

5.4 Sample data layout

The simplest layout of a file will start with an "ftyp" box, followed with a "moov" box containing the descriptions of the relevant tracks. Finally one or more "mdat" boxes will be present that hold the actual sample data. A diagrammatic example of this for a file containing three tracks is shown in Figure 1. In this example, the samples for each track are shown in different colours, with white representing potential spaces in the "mdat" box.

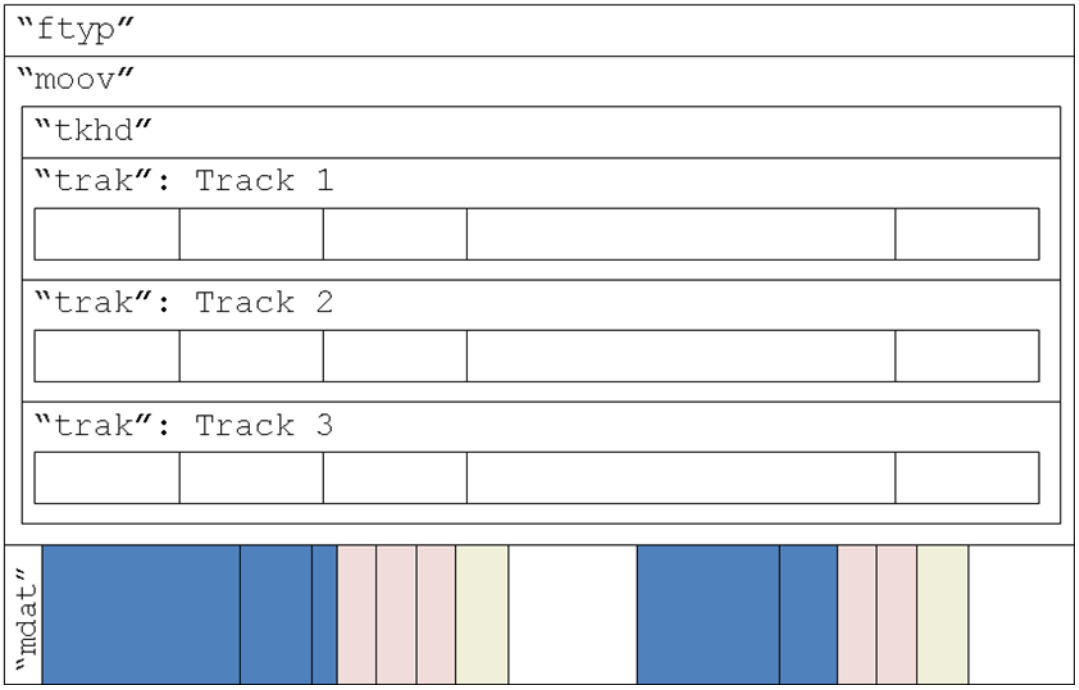


Figure 1: An example of basic file layout with "moov" box preceding single "mdat" box

An alternative and common structure is where the "moov" box is at the end of the file following all "mdat" boxes. A diagrammatic example of this is shown in Figure 2, similar to the example in Figure 1, but containing more samples.

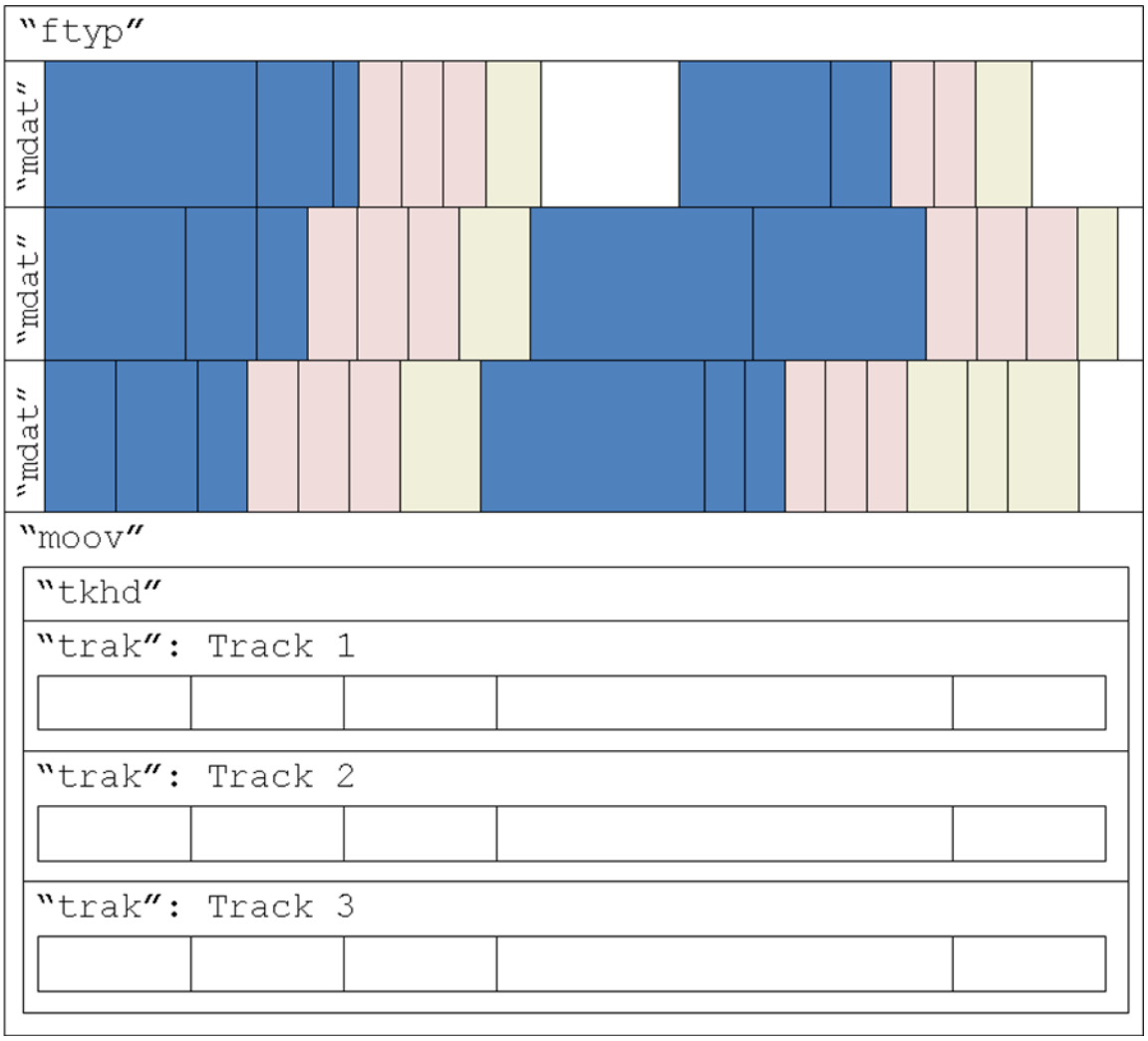


Figure 2: An example of basic file layout with "mdat" boxes preceding "moov" box

Players should cope with "mdat" boxes preceding "moov" boxes and support efficient skipping of these boxes to reach the information contained within the "moov" box.

NOTE: There may be sample data contained in the "mdat" box(es) preceeding the "moov" box; therefore the data in the "mdat" box(es) may need to be returned to when the file is processed.

For files that contain fragments, the simplest layout will also start with "ftyp" and "moov" boxes, but these will be followed by a sequence of "moof" and "mdat" boxes. A diagrammatic example of this is shown in Figure 3. Further discussion on fragments is provided in clause 8.

Figure 3: An example of basic file layout using fragments

6 Processing of RTP streams and reception hint tracks

6.1 Introduction

Clause 6 provides recommendations for recording of RTP streams and the use of recorded RTP streams for playback and re-sending.

Clause 6 is organized as follows:

- Clause 6.2 introduces the potential sources why the playback of RTP streams might become unsynchronized and provides an overview how proper synchronization is facilitated in recording and playback. It precedes the other clauses, because both the recording unit and the player have to take actions to achieve proper synchronization.
- Clause 6.3 provides recommendations for storing RTP streams in DVB files.
- Clause 6.4 provides recommendations how to play DVB files containing recorded RTP streams.
- Clause 6.5 provides recommendations for re-sending received RTP streams stored in DVB files as described in clause 6.3.

6.2 Synchronization of RTP streams

There are several potential sources of unsynchronized playback for received RTP streams. When RTP streams are recorded as RTP reception hint tracks, the necessary information for guaranteeing synchronized playback is also recorded. When RTP streams are recorded as media tracks, the synchronization of the playback of the media tracks has to be guaranteed by creating the composition times of the media samples appropriately. The following list describes the sources of unsynchronized playback for received RTP streams, summarizes the recommended synchronization means, and points to the relevant clauses for further information.

1) The RTP timestamp of the first packet of the stream has a random offset. Hence, the RTP timestamps of two streams are shifted by the difference of their initial random offsets even if the potentially different clock rate of the RTP timestamps of the different streams were compensated. The random offset should be reflected in the value of the offset field of the '`tsro`' box of the referred reception hint sample entry as described in clause 6.3.5.

2) The first received and recorded packet of the different streams may not have an identical playback time as discussed in clause 6.3.2. The unequal start time of the different recorded streams is compensated by parsing one or more RTCP Sender Reports to derive the playback time as the wallclock time of the sender and creating an initial offset of the playback using the Edit List box as described in clause 6.3.2. The Edit List box is interpreted by the player as described in clause 6.4.4.

3) There is no guarantee that the clock for producing the RTP timestamps of a certain RTP stream runs at the same pace as the wallclock time of the sender, which is used to create the RTCP Sender Reports. For example, the RTP timestamps may be generated on the basis of a constant sampling frequency, e.g. 44.1 kHz for audio, and hence governed by the clock rate of the audio capturing hardware. However, the RTP Sender Reports may be generated according to the system clock running at a different pace than the clock of the audio capturing hardware. Moreover, the clock used to generate RTP timestamps for audio might run at a different pace than the clock used to generate RTP timestamps for video (when both a normalized to the same clock tick frequency).

   A similar problem in the player arises if the clock pacing the output of a decoded stream runs at a different pace than the wallclock of the player or the clocks pacing the rendering of different decoded streams are not synchronized.

   The recommended approach for all these potential problems of clocks running at a different pace is to use RTCP Sender Reports to align the RTP timestamps of different streams onto the same wallclock timeline, which is used for inter-stream synchronization. This alignment can be done while recording the streams by modifying the representation of the recorded RTP timestamps or while playing the recorded streams by using the recorded RTCP Sender Reports as described in clause 6.3.6. Moreover, it is recommended to pace the playback according to the audio playout rate as described in clause 6.4.4.

4) The wallclock of the sender may run at a different pace than the wallclock of the player.

   It is recommended to play a recorded program at the pace of the wallclock of the player and to use the audio playout clock as the wallclock of the player. Consequently, the audio timescale does not typically have to modified. Even if the wallclock of the player ran at a different pace than the wallclock of the sender, it is typically unnoticeable.

   Pacing of the output of decoded media samples is described in clause 6.4.4.

6.3 Recording of RTP streams

6.3.1 Introduction

Recording of RTP streams can result into three basic file structures.

1) A file containing only RTP reception hint tracks. No media tracks are included. This file structure enables efficient processing of packet losses, but only players capable of parsing RTP reception hint tracks can play the file.

2) A file containing only media tracks. No RTP reception hint tracks are included. This file structure allows existing players compliant with the earlier versions of the ISO base media file format process recorded files as long as the media formats are also supported. However, sophisticated processing of transmission errors is not possible due to reasons explained in subsequent clauses.

3) A file containing both RTP reception hint tracks and media tracks. This file structure has both the benefits mentioned above and should be used when for as good interoperability as possible with other file formats derived from the ISO base media file format.

If an RTP stream being recorded is protected, a protected RTP reception hint track is used instead of an RTP reception hint track, while the operation of the recording unit remains unchanged otherwise. At the time of playback, the data included in the protected RTP reception hint track is unprotected first and then processed similarly to a conventional unprotected RTP stream. Alternatively, the RTP stream may be unprotected before storing it as a RTP reception hint track, but then care has to be taken that the rights to use the content in the protected RTP stream are obeyed.

Some of the recording operations are common for all the three file structures, while others differ. Table 1 indicates which recording operations are required for the basic file structures.

Table 1: Recording Operations for RTP Recordings

	File containing only RTP reception hint tracks	File containing only media tracks	File containing both RTP reception hint tracks and media tracks
Compensation for unequal starting position of received RTP streams (clause 6.3.2)	no, when RTCP reception hint tracks are stored; yes, otherwise	yes	no, when RTCP reception hint tracks are stored; yes, otherwise
Recording of SDP (clause 6.3.3)	yes	no	yes, for RTP reception hint tracks only
Creation of a sample within an RTP reception hint track (clause 6.3.4)	yes	no	yes, for RTP reception hint tracks only
Representation of RTP timestamps (clause 6.3.5)	yes	no	yes, for RTP reception hint tracks only
Recording operations to facilitate inter-stream synchronization in playback (clause 6.3.6)	yes	yes, the composition times of media tracks should be compensated as described in clause 6.3.6.3	yes
Representation of reception times (clause 6.3.7)	yes	no	yes, for RTP reception hint tracks only
Creation of media samples (clause 6.3.8)	no	yes	yes, for media tracks only
Creation of hint samples referring to media samples (clause 6.3.9)	no	no	yes

Some implementations may record first to RTP reception hint tracks only and create a file with a combination of media tracks and RTP reception hint tracks off-line.

Finally, clause 6.3.10 includes some recommendations how to record the streams of a DVB-H service item.

6.3.2 Compensation for unequal starting for position of received RTP streams

When the recording of RTP streams is started, it can happen that the presentation time of the first media sample in one RTP stream is not equal to the presentation time of the first media sample in another RTP stream at least due to the following reasons:

- The sampling frequency of audio and video typically differ.
- Audio and video streams may not be perfectly interleaved in terms of presentation times in transmission order.

If RTCP reception hint tracks are stored, the compensation for unequal starting position of received RTP streams should be done at playback time and no Edit List box concerning RTP reception hint tracks should be created. If RTCP reception hint tracks are not stored or if media tracks are stored, it is essential that the recording unit indicates the relative initial delay of the streams in order to synchronize audio and video correctly at the beginning of the playback of the streams as described subsequently in this clause. The recording unit should perform the following operations.

1. An RTCP Sender Report indicates which RTP timestamp corresponds to the wallclock time of the time instant the report was sent. At least the first RTCP Sender Report for each RTP stream should be parsed in order to establish an equivalence of an RTP timestamp of each RTP stream and a wallclock time of the sender. The wallclock timestamp of the earliest received RTP packet, in presentation order, is derived for each RTP stream by simple linear extrapolation.
2. The smallest wallclock timestamp derived above among all the received RTP streams is mapped to presentation timestamp zero in the movie timeline, i.e., is presented immediately at the beginning of the playback of the recorded file. The movie timeline is the master timeline for the playback of the file.

3. The media timeline for each track starts from 0. In order to shift the media timeline to a correct starting position in the movie timeline, an Edit box and an Edit List box are created for each of the other RTP tracks (which do not contain a packet having the earliest wallclock timestamp) as follows:

   The Edit List box contains two entries:

   a) The first entry is an empty edit (indicated by `media_time` equal to -1), and its duration (`segment_duration`) is equal to the difference of the presentation times of the earliest media sample among all the RTP streams and the earliest media sample of the track. Figure 4 presents an example of how the `segment_duration` of the first entry in an Edit List box is derived.

   b) The value of `media_time` of the second entry is equal to the composition time of the earliest sample in presentation order, and the value of `segment_duration` of the second entry spans over the entire track. As the actual duration of the track might not be known at the time of creating the Edit List box, it is recommended to set the `segment_duration` equal to the maximum possible value (either the maximum 32-bit unsigned integer or the maximum 64-bit unsigned integer, depending on which version of the box is used).

   The value of `media_rate_integer` is equal to 1 in both the entries of the Edit List box.

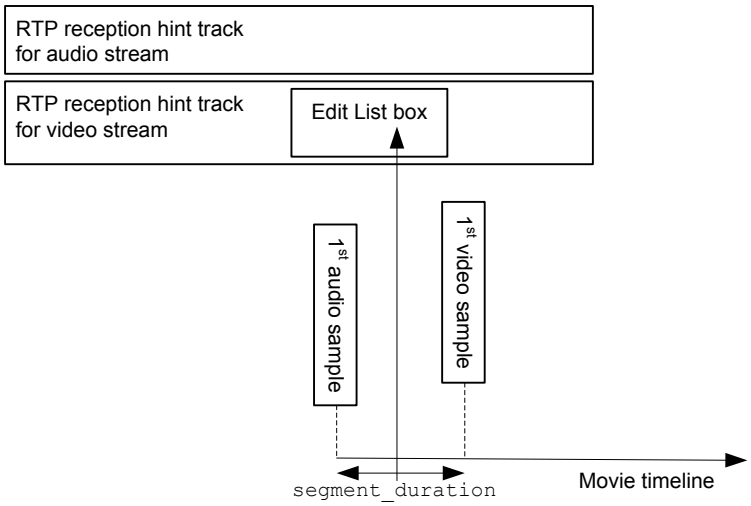


Figure 4. An example of creation of an Edit List box to compensate the unequal starting of the received RTP streams. `segment_duration` is copied to the first entry of the Edit List box.

Some recording units may detect packets from which decoding can be started, such as IDR pictures of H.264/AVC streams, which are here referred to as random access points. If a stream contains a packet having the earliest wallclock timestamp among all the received streams and the same stream contains packets preceding, in decoding order, the first random access point of the stream, it is recommended not to store the packets preceding the first random access point of the stream and not to consider them when determining the earliest wallclock timestamp among all the received streams.

6.3.3 Recording of SDP

It is required that there is one movie-level index track containing SDP indexes when a file contains one or more RTP reception hint tracks. The format and the use of the SDP index payloads is specified in clause 5.3.5.7 of the DVB file format [i.1].

If the initial SDP description is valid for the entire file, the SDP should be additionally stored as follows in order to obtain as good interoperability as possible with other file formats derived from the ISO base media format. Session-level SDP, i.e., all lines before the first media-specific line (“m=” line), should be stored as Movie SDP information within the User Data box, as specified in clause 9.1.4.1 of the ISO base media file format [i.3]. Each media-level section within the SDP description starts with an 'm=' line and continues to the next media-level section or the end of the whole session description. Each media-level section should be stored as Track SDP information within the User Data box of the corresponding RTP reception hint track.

6.3.4 Creation of a sample within an RTP reception hint track

It is recommended that each sample represents all received RTP packets that have the same RTP timestamp, i.e., consecutive packets in RTP sequence number order with a common RTP timestamp. The `RTPsample` structure is set to contain one `RTPpacket` structure per each received RTP packet having the same RTP timestamp. Each `RTPpacket` is recommended to contain one packet constructor of type 2 (`RTPsampleconstructor`). An `RTPsampleconstructor` copies a particular byte range, indicated by the `sampleoffset` and `length` fields of the constructor, of a particular sample, indicated by the `samplenumber` field of the constructor, by reference into the packet payload being constructed. The payload of each received RTP packet having the same RTP timestamp is copied to the `extradata` section of the sample. The track reference of each constructor is set to point to the hint track itself, i.e., is set equal to -1, and `sampleoffset` and `length` are set to match to the location and size of the packet payload within the sample.

Figure 5 presents a pseudo-code example of an RTP reception hint sample, which contains two RTP packets.

```
aligned(8) class RTPsample {
    unsigned int(16)    packetcount = 2;
    unsigned int(16)    reserved;
    RTPpacket           packets[packetcount]
    {
        RTPpacket {
            int(32)             relative_time;
            ...
            unsigned int(16)    entrycount = 1;
            RTPconstructor(2)
            {
                signed int(8)       trackrefindex = -1;
                unsigned int(16)    length; // number of bytes in the payload
                unsigned int(32)    samplenumber; // samplenumber of this sample
                unsigned int(32)    sampleoffset;
                unsigned int(16)    bytesperblock = 1;
                unsigned int(16)    samplesperblock = 1;
            }
        }
        RTPpacket {
            int(32)             relative_time;
            ...
            unsigned int(16)    entrycount = 1;
            RTPconstructor(2)
            {
                signed int(8)       trackrefindex = -1;
                unsigned int(16)    length; // number of bytes in the payload
                unsigned int(32)    samplenumber; // samplenumber of this sample
                unsigned int(32)    sampleoffset;
                unsigned int(16)    bytesperblock = 1;
                unsigned int(16)    samplesperblock = 1;
            }
        }
    }
    byte                extradata
    {
        byte rtppayload1[];
        byte rtppayload2[];
    }
}
```

Figure 5. An example of a RTP reception hint sample containing two packets (their header and payload).

The use of an error occurrence indexing event to indicate an RTP packet loss is not recommended, because the `RTPsequenceseed` field can be used for detecting packet losses without any increase in the storage space. Furthermore, the minimum unit the error occurrence event can refer to is a sample (in an RTP reception hint track). Since a sample can contain many packets, it is ambiguous which ones of these packets the error occurrence indexing event concerns.

6.3.5 Representation of RTP timestamps

RTP timestamps are represented in a RTP reception hint track by a sum of three values, one of which is the decoding time DT in the media timeline of the track. The decoding time is run-length coded into the Decoding Time to Sample box and additionally to one or more Track Fragment Run boxes, if a sample resides in a movie fragment. The Decoding

Time to Sample box includes a number of `sample_count` and `sample_delta` pairs, where `sample_delta` is the decoding time increment (i.e., the sample duration in terms of decoding time) for each sample in a set of consecutive samples, the number of which equals to `sample_count`. The Track Fragment Run box indicates one pair of `sample_count` and `sample_duration`, where `sample_duration` is the decoding time increment (i.e., the sample duration) for each sample in a set of consecutive samples, the number of which equals to `sample_count`. Each Track Fragment box can contain a number of Track Fragment Run boxes. The decoding time DT(i) for sample number i is derived by summing up the sample durations of all the samples preceding sample i from the Decoding Time to Sample box and, if needed, the Track Fragment Run boxes referring to any sample preceding sample i.

The RTP timestamp for sample i, RTPTS(i), is represented by a sum of three values specified as follows:

$$RTPTS(i) = (DT(i) + \texttt{tsro.offset} + \texttt{offset}) \bmod 2^{32} \quad (1)$$

where `tsro.offset` is the value of offset in the '`tsro`' box of the referred reception hint sample entry and `offset` is the value included in the `rtpoffsetTLV` box in the RTPpacket structure, and mod is the modulo operation.

A `'tsro'` box should be present in RTP reception hint sample entries. The value of offset in any `'tsro'` box of a track should be equal to the RTP timestamp of the first packet of the respective stream in RTP sequence number order.

Provided that no wrap-around of the RTP timestamp values over the maximum 32-bit unsigned integer happened between sample i-1 and i, the difference between consecutive unequal RTP timestamps, in RTP sequence number order, is

$$RTPTS\_DIFF(i) = RTPTS(i) - RTPTS(i-1) \text{ for any } i > 1 \quad (2)$$

RTPTS_DIFF(i) remains unchanged, when the frame rate is constant, the number of frames in any packet is constant, and the transmission order is the same as the presentation order. These constraints are typically met by audio streams and temporally non-scalable video streams. If RTPTS_DIFF(i) is a constant denoted as RTPTS_DIFF, the following is recommended. The value of `sample_delta` in the Decoding Time to Sample box and, if movie fragments are used, the value of `sample_duration` in the Track Fragment Run box or boxes are set to RTPTS_DIFF, which results into compact Decoding Time to Sample and Track Fragment Run boxes. The `rtpoffsetTLV` box should not be used within the RTP reception hint samples, if RTCP reception hint tracks are used (see clause 6.3.6). Otherwise (if RTCP reception hint tracks are not used), `offset` in the `rtpoffsetTLV` box should be set to 0.

When temporal scalability is used in a video stream, the transmission order and the playback order of packets are not identical, RTP timestamps do not increase as a function of RTP sequence number, and RTPTS_DIFF(i) is not constant. However, RTP timestamps typically have a constant behaviour in periods determined by the GOP_size, which is one plus the number of pictures between two consecutive pictures in the lowest temporal level in RTP sequence number order. For example, if two non-reference pictures are coded for each pair of reference pictures as illustrated in Figure 6, GOP_size is equal to 3. Figure 7 presents an example of a hierarchically temporally scalable bitstream with GOP_size equal to 4.

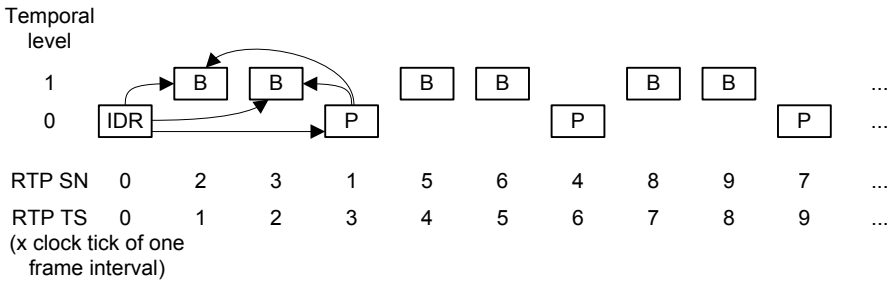


Figure 6. An example of a temporally scalable bitstream with GOP_size equal to 3.

NOTE 1: In this example, RTP sequence numbers (SN) are normalized to start from 0, and one packet per frame is assumed.
RTP timestamps (TS) are normalized to start from 0 and indicated as clock ticks lasting one frame interval. Inter prediction arrows are indicated for the first GOP only, while pictures in other GOPs are predicted similarly.

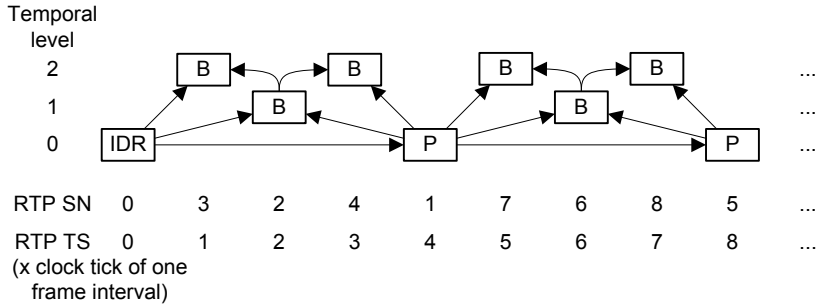


Figure 7. An example of a hierarchically temporally scalable bitstream with GOP_size equal to 4.

NOTE 2: In this example, RTP sequence numbers (SN) are normalized to start from 0, and one packet per frame is assumed.
RTP timestamps (TS) are normalized to start from 0 and indicated as clock ticks lasting one frame interval.

The RTP timestamp increment caused by one GOP is derived as follows, when no wrap-around of the RTP timestamp values over the maximum 32-bit unsigned integer happened between sample i and i + GOP_size, inclusive:

$$RTPTS\_GOP\_DIFF(i) = RTPTS(i + GOP\_size) - RTPTS(i) \quad (3)$$

If RTPTS_GOP_DIFF(i) is a constant equal to RTPTS_GOP_DIFF, when no sample i, i + 1, …, i + GOP_size is a picture starting a so-called closed group of pictures, such as an IDR picture of H.264/AVC streams, the following is recommended. The value of `sample_delta` in the Decoding Time to Sample Box and, if movie fragments are used, the value of `sample_duration` in the Track Fragment Run box or boxes are set to RTPTS_GOP_DIFF / GOP_size. The `rtpoffsetTLV` box should not be used for pictures in the lowest temporal level, if RTCP reception hint tracks are used (see clause 6.3.6). Otherwise (if RTCP reception hint tracks are not used), `offset` in the `rtpoffsetTLV` box should be set to 0. The value of `offset` in the `rtpoffsetTLV` box should be set for pictures in other temporal levels to such that Equation (1) is fulfilled. Figure 8 indicates how the decoding time and `offset` are set for a hierarchically temporally scalable video bitstream presented in Figure 7.

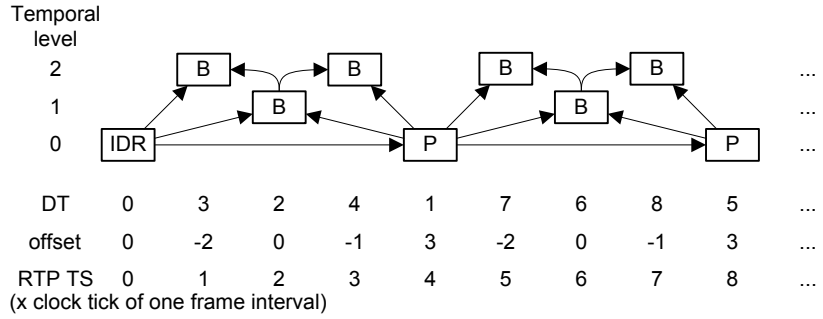


Figure 8. An example of setting the decoding time (DT) and the value of offset in the rtpoffsetTLV box of a hierarchically temporally scalable bitstream with GOP_size equal to 4.

NOTE 3: In this example, the decoding time increment between samples is set equal to RTPTS_GOP_DIFF / GOP_size to have a compact encoding decoding times. The value of offset in the rtpoffsetTLV box is adjusted for each sample to store a representation of the RTP timestamp. For this illustration, RTP timestamps and decoding times are normalized to start from 0 and indicated as clock ticks lasting one frame interval.

If no linear and periodical behaviour of RTP timestamps is detected from the received packets and no two received packets of different samples have the same reception time, it is recommended to set the value of `sample_delta` in the Decoding Time to Sample Box and, if movie fragments are used, the value of `sample_duration` in the Track Fragment Run box or boxes to represent the reception time of the first packet of the sample. That is, the derived decoding time DT(i) should be equal to the reception time of the first packet of the sample subtracted by the reception time of the first packet of the first received sample of the stream.

It is noted that composition timestamps are not explicitly indicated in the file for samples in any hint tracks. Consequently, for RTP reception hint tracks, the composition timestamps are inferred from the information related the RTP timestamps indicated in the stored packet stream. For an RTP reception hint track that is not associated with an RTCP reception hint track, the composition time of a received RTP packet is inferred to be the sum of the sample time DT(i) and the value of the `offset` field in the `rtpoffsetTLV` box including the sample. For an RTP reception hint track that is associated with an RTCP reception hint track, the composition time is inferred as follows. Let the received RTP packet having the earliest RTP timestamp within the same track have composition time equal to 0. Any remaining RTP packet has a composition time equal to the RTP timestamp difference of the present RTP packet and the earliest RTP packet in presentation order with clock drift correction similar to clause 6.3.6.3. The composition time refers to the media timeline of the track.

6.3.6 Recording operations to facilitate inter-stream synchronization in playback

6.3.6.1 General

Lip synchronization, i.e., correct synchronization between recorded RTP streams, during playback can be facilitated at least with the following two means:

1) An RTCP reception hint track is generated for each RTP reception hint track. The potential clock drift between the RTP timestamp clocks of different streams is corrected at the time when the file is parsed and the media streams included in the file are decoded and played. The clock drift correction is done similarly to as would be done for RTP streams that are received and played simultaneously. This mode of operation is straightforward for the recording units. However, accessing a file from an exact playback position might be more cumbersome, because it requires compensation of the clock drift of all the recorded streams at the time of the access.

2) The potential clock drift between recorded RTP streams is corrected by modifying the RTP timestamps of one or more recorded streams. This mode of operation is requires processing of RTCP Sender Reports at the time of recording and is hence more tedious for the recording units than creation of RTCP reception hint tracks. However, the operation of the player is straightforward.

Recording units should use the timestamp synchrony box (clause 9.4.1.2 of [i.3]) to indicate which lip synchronization approach has been used. The timestamp synchrony box includes the `timestamp_sync` field. `timestamp_sync` equal to 1 indicates that players should use RTCP reception hint tracks for lip synchronization. `timestamp_sync` equal to 2 indicates that players should use composition timestamps for lip synchronization.

Some implementations may create RTCP reception hint tracks first during the real-time recording operation and then compensate the clock drift by modifying RTP timestamps as an off-line post-processing step.

The following clauses provide more details about both approaches.

6.3.6.2 Facilitating lip synchronization based on RTCP Sender Reports

A recording unit stores all RTCP Sender Reports for a particular RTP stream as samples in the respective RTCP reception hint track.

6.3.6.3 Compensating clock drift in timestamps

It is not recommended to modify the RTP timestamps of the recorded audio streams. Such a modification would cause an audio timescale modification in the player, which is a non-trivial operation.

The recorded representation of the RTP timestamps of the video and other non-audio streams should be modified using the following procedure.

1) First, the wallclock timestamp a of a video frame is derived from the RTP timestamp corresponding to the video frame as a sum of the wallclock timestamp of the previous video frame and the difference of the RTP timestamps of the current and previous video frames in the units of the wallclock timeline.

2) Second, the playback time b for the video frame on the wallclock time is derived based on the RTCP Sender Reports. If no RTCP Sender Report that exactly indicates the wallclock time for the video frame is available, the wallclock time can be extrapolated assuming that the rate at which the RTP timestamp clock and the sender wallclock in RTCP Sender Reports deviates stays unchanged.

3) Third, based on the RTCP Sender Reports for audio, the audio RTP timestamp that is played simultaneously with the video frame at time *b* of the wallclock timeline is derived. There need not be an audio frame having exactly the derived audio RTP timestamp. The wallclock timestamp *c* of an audio sample is calculated from the derived audio RTP timestamp as a sum of the wallclock timestamp of the preceding audio frame and the difference of the RTP timestamps of the derived audio RTP timestamp and the RTP timestamp of the preceding audio frame.

The difference between *a* and *c*, if any, should be compensated in the fields that represent the video RTP timestamp in the file. In practice, the easiest way might be to add the difference to the `offset` field in the rtpoffsetTLV box, which is illustrated in Figure 9. The other option, rewriting the Decoding Time to Sample box and the Track Fragment Run boxes (if any), might be more cumbersome to implement, because of particular way of coding the sample times by a combination of sample counts and durations, and might require more storage space too.

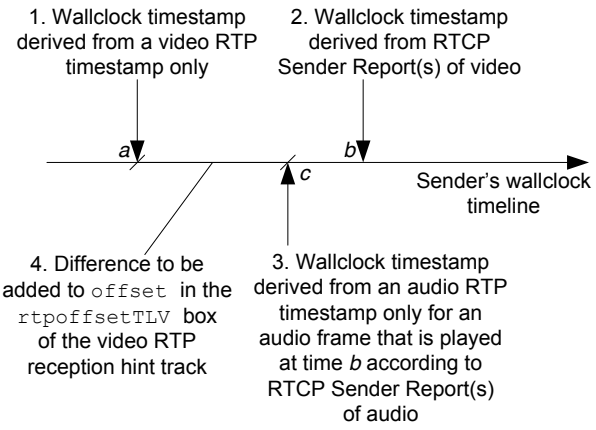


Figure 9. An example of correcting the lip synchronization in the RTP timestamp representation.

6.3.7 Representation of reception times

As specified in[i.3], the reception time of a packet is indicated by the sum of the decoding time of the sample containing the packet and the value of `relative_time` of the RTPpacket structure of the packet.

The reception time of the earliest received RTP packet should be zero, and the reception times of all subsequent packets should be relative to the reception time of the earliest received RTP packet.

The clock source for the reception time is undefined and may be, for instance, the wallclock of the receiver. If the range of reception times of a reception hint track overlaps entirely or partly with the range of reception times of another reception hint track, the clock sources for these hint tracks shall be the same.

The reception time of a packet should correspond to the time instant when the protocol stack layer underneath RTP, typically UDP, outputs the packet.

6.3.8 Creation of media samples

Media samples are created from the received RTP packets as instructed by the relevant RTP payload specification and RTP itself [i.9]. However, most media coding standards only specify the decoding of error-free streams and consequently it should be ensured that the content in media tracks can be correctly decoded by any standard-compliant media decoder. Handling of transmission errors therefore requires two steps: detection of transmission errors and inference of samples that can be decoded correctly. These steps are described in the subsequent paragraphs.

Lost RTP packets can be detected from a gap in RTP sequence number values. RTP packets containing bit errors are usually not forwarded to the application as their UDP checksum fails and packets are discarded in the protocol stack of the receiver. Consequently, bit-erroneous packets are usually treated as packet losses in the receiver.

The inference of media samples that can be correctly decoded depends on the media coding format and is therefore not described here in details. Generally, inter-sample prediction is weak or non-existing in audio coding formats, whereas most video coding formats utilize inter prediction heavily. Consequently, a lost sample in many audio formats can often be replaced by a silent or error-concealed audio sample. It should be analyzed whether a loss of a video packet concerned a non-reference picture or a reference picture, or, more generally, in which level of the temporal scalability

hierarchy the loss occurred. It should then be concluded which pictures may not be correctly decodable. For example, a loss of a non-reference picture does not affect the decoding of any other pictures, whereas a loss of a reference picture in the base temporal level typically affects all pictures until the next picture for random access, such as an IDR picture in H.264/AVC. Video tracks must not contain any samples dependent on any lost video sample.

6.3.9 Creation of hint samples referring to media samples

Media samples are created from the received RTP packets as explained in clause 6.3.8. RTP reception hint tracks are created as explained in clause 6.3.4, but the contents of the `RTPpacket` structure depend on the existence of the corresponding media sample as follows.

- If the packet payload of the received RTP packet is represented in a media track, the track reference of the relevant packet constructors are set to point to the media track and include the packet payload by reference. It is not recommended to have a copy of the packet payload in the `extradata` section of the received RTP sample in order to save storage space and make file editing operations easier to implement.
- If the packet payload of the received RTP packet is not represented in a media track, the instance of the `RTPpacket` structure is created as explained in clause 6.3.4.

6.3.10 Recording of a DVB-H service item

It is recommended to store the RTP streams of a recorded DVB-H service as RTP reception hint tracks or media tracks or a combination thereof. Hence, it is recommended to run the protocol stack of the receiver up to, but excluding, RTP. For example, MPE-FEC decoding [i.10], if applicable, should be performed at the time of recording, before storing the respective data streams into the file. The stored reception time for all RTP packets of the same MPE-FEC frame should be identical and match the moment when the MPE-FEC decoding has been completed. If movie fragments are created in the file being recorded, it is recommended to generate one movie fragment per each MPE-FEC frame.

6.4 Playing of recorded RTP streams

6.4.1 Introduction

Clause 6.4 describes operations required for playback of a DVB file containing recorded RTP streams. Clause 6.4 is organized as follows:

- Before RTP streams can be played, the contents of the files should be analyzed. Particularly, alternative tracks representing the same media stream should be identified and one of these tracks should be selected for decoding and playback. The coding format should be detected in order to conclude up front that it can be decoded by the player. These preparation operations are described in more details in clause 6.4.2.
- If an RTP reception hint track is being processed, there are a few things to be taken into account as described in clause 6.4.3. For example, packet losses should be detected and handled appropriately.
- The synchronization of the decoded media samples should be handled properly as described in clause 6.4.4.
- If the RTP streams stored in a file are accessed from a position other than the beginning of the streams, proper inter-stream synchronization and decoder initialization are needed as described in clause 6.4.5.

6.4.2 Preparation for the playback

In the preparation phase for playback, the player selects which tracks are played. The basic track structure of the file is parsed first. The tracks are grouped according to which alternate group they belong to. Tracks that belong to the same alternate group are indicated by the same value of `alternate_group` in the track header box. One track from each alternate group is selected for playback as follows.

If there is an RTP reception hint track in the alternate group, it is preferred for playback, because it contains an entire representation of the received RTP stream, unlike media tracks derived from the received RTP streams, which might use such subset of the received RTP packets that can be decoded by any standard-compliant decoder without capability for handling packet losses.

The compatibility of the player with the selected track should be ensured. For example, it should be examined whether the codec, the profile, and the level used in the track are such that the player is able to support.

The codec, profile, and level used for the coded bitstream in an RTP reception hint track can be concluded from the SDP description of the RTP stream. The SDP descriptions are stored in the movie-level index track. If SDP is unchanged throughout the file, it may be additionally stored as Movie SDP information and Track SDP information within User Data boxes. If Track SDP information is present, it may be parsed to find out the codec, profile, and level used for the bitstream contained in the RTP reception hint track. If Movie SDP information or Track SDP information is not present, the move-level index track is traversed to find and parse each SDP index and, consequently, the codec, profile, and level used for the bitstream contained in the RTP reception hint track.

If no RTP reception hint track exists in an alternate group, the sample entry or sample entries of the media tracks in the alternate group should be examined to find out which ones of them the player is able to support.

6.4.3 Decoding of a sample within an RTP reception hint track

The original RTP packets may be reconstructed from an RTP reception hint sample by creating the RTP packet header from the RTPpacket structures and by resolving the constructors of the RTPpacket structures. Hence, one approach for file players to process RTP reception hint tracks is to re-create the packet stream that was received and process the re-created packet stream as if it was newly received.

The `relative_time` field included in the RTPpacket structure may be used to schedule the insertion of the packet into the buffer for the RTP receiver. However, it may be more advisable to modify the decoding process of recorded RTP streams such a manner that the decoder output buffers are kept as full as possible in order to avoid interruptions or jerky playback caused by late packets or occasional problems in real-time decoding in systems running other processes in addition to the player.

Packet losses should be detected from gaps in the RTP sequence number. The reaction to packet losses depends on the particular media decoder implementation and may also depend on user preferences.

6.4.4 Lip synchronization

The following steps are required for achieving correct synchronization between streams:

1) Inter-track synchronization at the start of the playback.

   The starting position of the media timeline of a track may be shifted in the movie timeline of the file as described in the following two paragraphs.

   For a media track and an RTP reception hint track that is not associated with an RTCP reception hint track, an Edit List box is used to shift the starting position of the media timeline within the move timeline as described in clause 6.3.2. In DVB files, the Edit List box of a track can only be used for selecting the starting position of the media timeline within the movie timeline. The playback of each media track and each RTP reception hint track that is not associated with an RTCP reception hint track starts at the movie timeline position indicated in the Edit List box of the track or from the beginning of the movie timeline, if no Edit List box exists for the track.

   For RTP reception hint tracks that are associated with respective RTCP reception hint tracks, the shifting of the starting position of the media timeline within the movie timeline is inferred as follows. The media timeline of the RTP reception hint track containing the earliest RTP packet (in presentation time on the sender wallclock timeline) among all RTP reception hint tracks is not shifted within the movie timeline (i.e., starts at time 0 on the movie timeline). The starting time of the media timeline of the any other RTP reception hint track is equal to the timestamp difference of the earliest RTP packets of the present track and the track containing the earliest RTP packet among all RTP reception hint tracks.

2) Reconstruction of RTP timestamps and composition times on the media timeline (clause 6.3.5).

3) Correction of RTP timestamps and composition times based on RTCP Sender Reports, if RTCP reception hint tracks are used.

   The correction is done similarly to what is described in clause 6.3.6.3. However, instead of adding the difference between times a and c into the representation of the RTP timestamps in the file, the difference is added during the playback to the presentation times of the video frames on the movie timeline.

4) Pacing the output of the decoded media samples.

It is recommended to play a recorded program at the pace of the wallclock of the player and to use the audio playout clock as the wallclock of the player. The audio playback is arranged to be continuous at the native sampling frequency of the audio signal. A presentation clock of the player runs at the pace of the audio playback, i.e., its value is always equal to the (the number of the most frequent uncompressed audio sample that was played out) × (sampling frequency of the audio signal). The playback of the video track (and potential other continuous media tracks) is synchronized to the presentation clock of the player. In other words, when the presentation clock of the player meets the composition time of a video sample on the movie timeline, the video sample is played out.

Only if a file being simultaneously recorded and played back and if the receiver wallclocks runs faster than the sender wallclock, pacing the playback according to the rate of the receiver wallclock might not be recommended and synchronizing the rate of the receiver wallclock to the rate of the sender wallclock may be done as follows.

The pace of the sender clock is recovered by creating a relationship between the reception times (according to the receiver clock) and the respective wallclock timestamps of the sender, which are reconstructed from RTCP Sender Reports. It is recommended to use the audio playout clock as the receiver clock. As the delay in the network and in the receiver may be varying, the relation between the reception times and the respective timestamps of the sender should be averaged over a large number of received packets. A timescale multiplication factor is concluded as a result of the averaging of the relation between the reception times and the respective timestamps of the sender.

A presentation time on a timeline of the receiver clock is derived for each sample. If RTCP reception hint tracks are in use, the presentation time is the composition time of the sample on the movie timeline, also including clock drift correction as described in step 3 above. If RTCP reception hint tracks are not in use, the presentation time is directly the composition time of the sample on the movie timeline. Then, for playback purposes only, the presentation times of the samples in all tracks being played should be multiplied by the timescale multiplication factor.

Time stretching of the signal should be done accordingly. Samples are played out at their presentation times.

In practice, the timescale multiplication factor and the mapping from the RTP timeline to the wallclock of the sender (step 3 above) may be implemented as a single operation.

6.4.5 Random access

Random access refers to a non-linear access to the media streams represented in the file. In other words, in a random access operation the file is accessed from another sample than that which was previously played or the file is initially accessed from a position that is not the beginning of the movie timeline.

It is recommended to provide the random access functionality to the user relative to the movie timeline of the file rather than any other timelines, such as the sender wallclock timeline. By using the movie timeline as the basis, the number of steps for a random access operation is kept low.

First, it is derived which media frames are at a desired random access position (or closest to it, if there are none exactly at the desired random access position). In the case of media tracks, RTP reception hint tracks for audio, and any RTP reception hint tracks having the timestamp_sync field equal to 2 (indicating pre-compensated lip synchronization), the media frame closest to the desired random access position can be directly derived based on the composition timestamps (on the media timeline) shifted by the initial starting position indicated in the Edit List box, if any. In the case of non-audio RTP reception hint tracks having the timestamp_sync field equal to 1 (indicating the use of RTCP reception hint tracks), the presentation times of samples should be derived as described in clause 6.4.4 until the closest presentation time to the desired random access position is found.

Second, decoding of many media bitstreams can be started only from frames of a particular type, such an IDR picture of H.264/AVC. Player implementations may therefore have different approaches, including the following:

1) Discover the closest frame at or preceding the desired random access position from which decoding can be started, start decoding from that frame, and start rendering only from the desired random access point. This approach may imply some processing delay before the rendering is started.

2) Start decoding and rendering at or after the desired random access point using the earliest frame from which decoding can be started. Typically, audio playback would start earlier than video playback, but the processing delay before the rendering is started is smaller than in the previous option.

6.5 Re-sending recorded RTP streams

6.5.1 Introduction

It may be a desirable operation to re-send the RTP streams that have been recorded earlier to a DVB file. For example, if RTP streams are received through a DVB service and recorded into a DVB file, it may be desirable to re-send them from one device to another device in a home environment using a WLAN connection. Clause 6.5 provides recommendations for re-sending of recorded RTP streams.

A communication system based on RTP includes a source endpoint (a.k.a., a sender) and a destination endpoint (a.k.a., a receiver) and may contain one or more mixers and translators. The sender and the receiver are the endpoints of the RTP and RTCP sessions. The behaviour of RTP translators and mixers is specified in [i.9] and clarified in [i.11]. In general, the recording unit receiving RTP streams and storing them into a file acts as a destination endpoint, and a re-sending unit reading stored RTP streams from a file and sending them acts as a source. Typically, the payloads of the re-sent RTP stream are not modified, which makes a combination of a recording unit and a re-sending unit acting similarly to a transport translator as described in [i.11]. However, the essential characteristic of a translator is that receivers cannot detect its presence. Consequently, a combination of a recording unit and a re-sending unit cannot act as a transport translator, unless re-sending happens simultaneously with the recording of the original streams. As this case is considered rare, the discussion in this clause regards a recording unit as a destination terminating the original RTP and RTCP sessions and a re-sending unit as a source of new RTP and RTCP sessions.

Clause 6.5 is organized as follows:

- Clause 6.5.2 includes recommendations how to compose RTP packets from RTP reception hint tracks and how to schedule the transmission of the RTP packets.
- Clause 6.5.3 discusses how RTCP packets should be generated and how received RTCP packets should be processed.

6.5.2 Re-sending RTP packets

The packets are recommended to be constructed and transmitted as follows.

The packet payloads are recommended to be constructed according to the constructors stored in the reception hint track, i.e., the packet payloads are recommended to be identical to those received, unless a different packet size is crucial for the network to which the packets are re-sent.

The values of the header fields for the RTP packets created as suggested by an RTP reception hint track should be kept the same as in the respective RTPpacket structure except for the following cases:

- The initial RTP timestamp offset and the RTP sequence number offset should be selected randomly regardless of the values stored in the offset field of the '`tsro`' box of the referred reception hint sample entry or the values of the `RTPsequenceseed` field of the RTPpacket structure of any for any of the packets of the respective RTP reception hint track.
- The value of the RTP timestamp field should be a sum of the random initial offset, the value of `offset` in the `RTPpacket` structure, and the decoding time of the respective RTP sample. If the sum exceeds the maximum unsigned 32-bit integer, it should be wrapped over.
- The relative increments of the RTP sequence number should be the same as those recorded in the values of the `RTPsequenceseed` fields. Consequently, if there was a packet loss in the stream that was recorded, the stream that is re-sent also has a respective gap in the RTP sequence number, and the receiver is able to deduce a packet loss.
- The value of the CSRC count field should always be zero, because no contributing sources of the previous RTP session that was recorded are actively modifying the streams for the RTP session for the stream being re-sent. The source identifier space (for both SSRC and CSRC) is session specific. Consequently, the CSRC list of the RTP header should be empty regardless of the potentially stored CSRC values for the received streams, which are included in the `receivedCSRC` TLV box in the `RTPpacket` structure.
- The value of the payload type field may be dynamically selected depending on the signalling scheme in use.

- The value of the SSRC field should be randomly selected and potential collisions should be handled as specified in [i.9]. The SSRC value of a received stream may be stored in the `ReceivedSsrcBox` of the referred reception hint sample entry but it should be ignored when the stream is re-sent.

- The recorded RTP header extensions, stored in `rtphdrextTLV` in the `RTPpacket` structure, if any, should be re-sent only if the re-sending unit can verify that they are valid for the re-sent stream. If the re-sending unit is not able to parse the semantics of the recorded RTP header extensions, they should not be re-sent.

The reception time of a packet, represented by the sum of the decoding time of the RTP reception hint sample containing the packet and the value of the `relative_time` of the `RTPpacket` structure, equals to the transmission time of the packet with a skew caused by the transmission delay and the processing delay in the protocol stack of the receiver. The skew of adjacent packets might not be equal due to transmission delay jitter and varying processing delay. Moreover, the protocol stack used when receiving the stream might differ from the protocol stack used for re-sending the stream. For example, the stream might have been received from a DVB-H service, where time-slicing is used and reception times have therefore a bursty nature, but it may be re-sent over a connection of a steady throughput. Due to these reasons, the reception times are often not applicable as such to pace the transmission of the re-sent packets. In all cases, the re-sending unit should verify that the re-sent packet stream complies with the buffering model in use, if any. If the re-sending unit can conclude that the network environments and protocol stacks used when receiving the stream and when re-sending the recorded stream are similar, reception times may be used as a basis for scheduling the packet transmission. The re-sending unit should make an effort to remove or conceal the transmission delay jitter in the recorded stream. If the re-sending unit is unable to conclude that the network environments and protocol stacks used when receiving the stream and when re-sending the recorded stream are similar or is uncertain which kind of packet scheduling is appropriate, it may use the decoding time as the basis for scheduling.

6.5.3 RTCP processing

RTCP Sender Reports and other RTCP messages are regenerated following the constraints specified in [i.9] rather than directly using the RTCP messages recorded in RTCP reception hint tracks, if any.

An RTCP Sender Report contains the wallclock time when the report was sent and the RTP timestamp corresponding to the same time as the indicated wallclock time. The RTP timestamp for an RTCP Sender Report is generated as follows. A presentation time on a timeline of a reference clock is derived for the sample corresponding the indicated wallclock time in the RTCP Sender Report. The reference clock may be the wallclock of the re-sending unit initialized to 0 at the beginning of the session. The sample corresponding to the indicated wallclock time might not exist in the corresponding RTP reception hint track, because the sampling instants of the samples in the RTP reception hint tracks might not match with the transmission instants of the RTCP Sender Reports. However, as instructed by [i.9], the RTP timestamp is derived as if there was a sample in the RTP stream corresponding to the indicated wallclock time. The RTP timestamp for an RTCP Sender Report should be linearly interpolated from the RTP timestamps of the samples immediately preceding and following the wallclock time indicated in the RTCP Sender Report. In order to conclude the samples immediately preceding and following the wallclock time indicated in the RTCP Sender Report, presentation times on the timeline of the reference clock should be derived until the closest samples are discovered. If RTCP reception hint tracks are present for the RTP reception hint track being re-sent, the presentation time is the composition time of the sample on the movie timeline, also including clock drift correction as described in step 3 of clause 6.4.4. If RTCP reception hint tracks are not present, the presentation time is directly the composition time of the sample on the movie timeline.

When handling the received RTCP Receiver Reports, it should be noticed that the reported cumulative number of packets lost includes also the unsent packets that were never originally received and correspond to the gaps in the RTP sequence number in the RTP reception hint tracks. Any congestion management, retransmission, or other packet loss resilience method should take this into account.

7 Processing of transport streams and reception hint tracks

7.1 Introduction to transport streams

This clause provides recommendations about Transport Streams within the file format. With the exception of clause 7.6, it is assumed that the Transport Streams recorded are single program transport streams, with all components stored within a single multiplexed transport stream reception hint track. It is not recommended to record multi-program

transport streams (MPTS) within DVB files. It is not recommended that multiple transport stream reception hint tracks are used for the separate PIDs that make up a transport stream.

This clause is organised as follows:

- Clause 7.2 provides an overview of the timing issues for transport streams and provides recommendations on how to handle timing within Transport Stream reception hint tracks in DVB files
- Clause 7.3 provides recommendations on the recording of transport streams in DVB files.
- Clause 7.4 provides recommendations on the playback of transport streams in DVB files.
- Clause 7.5 provides recommendations on the streaming of transport streams in DVB files.
- Clause 7.6 discusses issues with multi-program transport streams.
- Clause 7.7 discusses the use of external files for storing samples in the context of a dvt1 brand.

7.2 Timing and transport streams

7.2.1 Clock rate for "rm2t" tracks

It is strongly recommended that the timescale (i.e. clock rate) used for files containing an "rm2t" track is 90000. Whilst the use of alternate frequency clocks may reduce potential timing errors, most systems that support transport streams do so using 27MHz clocks, or the derived 90kHz clock.

Care should be taken when handling the timing values, especially since wraparound when using 32-bit values internally can occur.

A 32 bit timer operating at the 90kHz clock frequency wraps round in approximately 13 hours. The MPEG-2 PCRs use a 33-bit value for the 90kHz component of the clock. Care should be taken to handle the wraparound values and different value lengths. 64-bit values for the clock are supported by the DVB File Format, and may be used as necessary. A reader should be able to support these, and a writer must use them as necessary.

NOTE 1: The 27MHz timing value may be used where the increased accuracy is essential, and parsers should cope with this mode of operation.

NOTE 2: The "mvhd" and"tkhd" boxes can grow when 64-bit values are used to replace 32-bit values. It is recommended that a "free" box is used to ensure space is available for this growth, which is 12-bytes for all boxes.

7.2.2 Timing sources

The DVB File Format has a timing source in the media time of the file. Since the MPEG-2 transport stream represents a system layer, this also contains a timing source in the form of the system clock encoded through PCR samples present in the stream[2]. The file should be created so that these two clocks can be used interchangeably, via a constant offset applied which corresponds to the system clock value of the transport stream at the start of the reception hint track.

However, a degree of jitter is possible between these clocks, as discussed below. It is recommended that implementations are designed to cope with jitter between these clocks.

7.2.3 Calculating sample duration

The transport stream reception hint track stores MPEG-2 Transport Streams with each packet as a single sample. This means has consequences for the mechanisms used within the file format for recording the details of the samples. Firstly, and most obviously, the number of samples is often higher than with media tracks. Secondly the size of a sample is fixed (as it often is with audio media tracks), but the duration is variable. These facts combine to make the handling of the timing of samples in the MPEG-2 transport stream critical to the performance and size overheads of DVB files.

[2] An MPTS can contain multiple different system clocks, as discussed in clause 7.6.

Often, an "rm2t" track will contain only a single program or service of a variable bitrate that has been extracted from a constant bitrate multiplex. In this case, the duration of each transport packet can be calculated using one of three ways:

1) Reception Time: in this mechanism the arrival time of each packet is noted, and the packet is assumed to finish arriving when the next packet that is part of the same service arrives. This means that the duration time of each transport packet can vary.

2) Piecewise Linear CBR[3] (PLCBR): in this mechanism the receiving device to monitors the PCRs for the received service, and calculate the duration of the packets between a pair of PCRs by assuming a constant bitrate for the packets received between the PCRs. The consequences of this is that the timings of a packet cannot be calculated until after the next PCR is received, but that the packets between PCRs all have the same sample duration. Since the timing values are stored as an integer duration for each individual packet, the accuracy limits of an integer value may result in a timing jitter between the media time and that of the PCR, when the integer value is used for a number of samples.

NOTE 1: 13818-1:2007 uses the piecewise linear CBR calculations as the means of calculating time for the purposes of buffer modelling, as defined in 2.4.2.2 of the MPEG-2 Systems Specification [i.12].

3) Extended PLCBR (E-PLCBR): this mechanism is essentially identical to mechanism 2 above, except that the calculations may take place over more than two received PCRs.

Figure 10 below shows an example of the different timings that might be generated by the first two mechanisms for a sequence of packets.

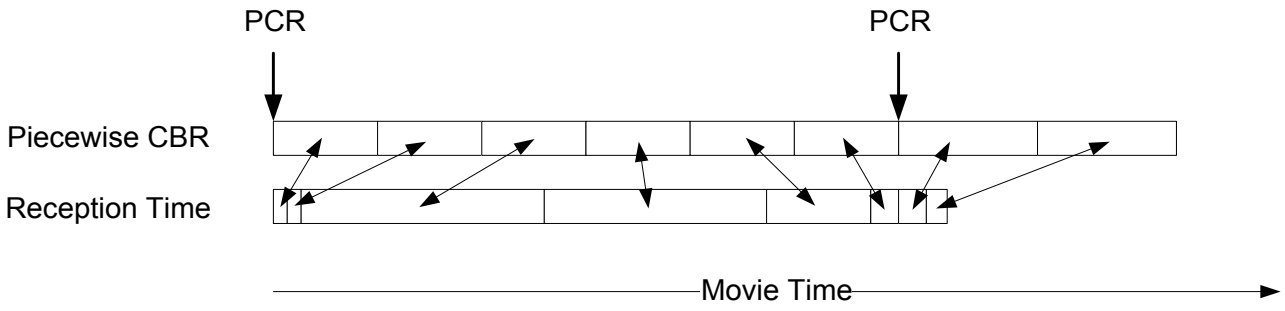


Figure 10: Piecewise CBR versus Reception Time

The three mechanisms differ in the trade-off they introduce between movie time versus reception time jitter and the size of the metadata stored to hold the timing values. Figure 11 below shows the jitter between the sample times as derived from the values stored in the "stts" box(es) and those calculated accurately by PLCBR without rounding to integer values. Although it is possible to eliminate the jitter at the PCR locations by generating two "stts" box entries, this is not considered necessary as a level of jitter should be accepted, and the consequential increase in metadata size is considered unacceptable. As the timing values are stored as two 32-bit values, one for the duration and one for the number of samples, increasing the number different values increases the size of the timing information to be stored and hence the size of the "stts" box (and hence "moov" box) or the equivalent information in the "moof" box.

[3] Also sometimes called Piecewise Linearity Between PCRs.

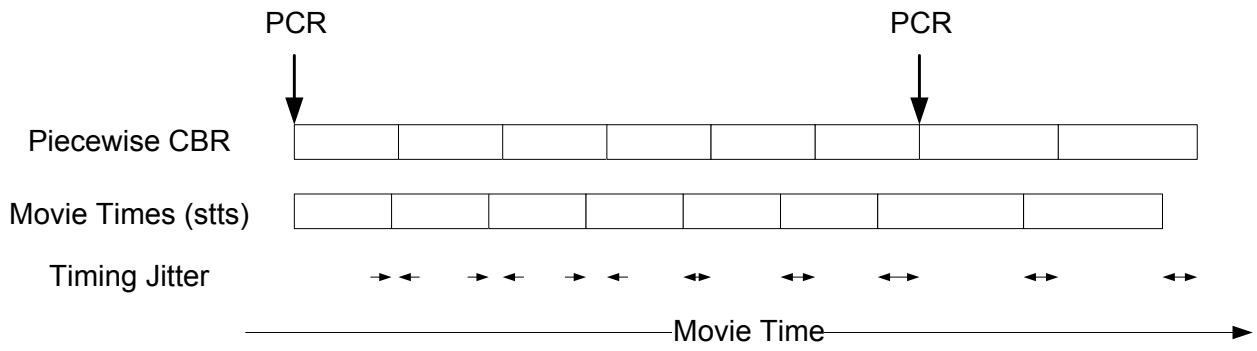


Figure 11: Timing jitter between PLCBR and values stored in "stts" box

It is recommended that the third mechanism, E-PLCBR, is used, but with an acknowledged jitter value.

The method used for timing calculation is signalled by the 'tsti' box in the track header. It is recommended that the tsti box is present.

The number of PCRs over which the averaging can take place represents a design trade-off between:

- The size reduction of the "stts" box, and hence the "moov" box (or the "moof" equivalents),
- The introduced jitter between the media time and the transport stream's system clock,
- The delay before sample metadata can be written into the file[4].

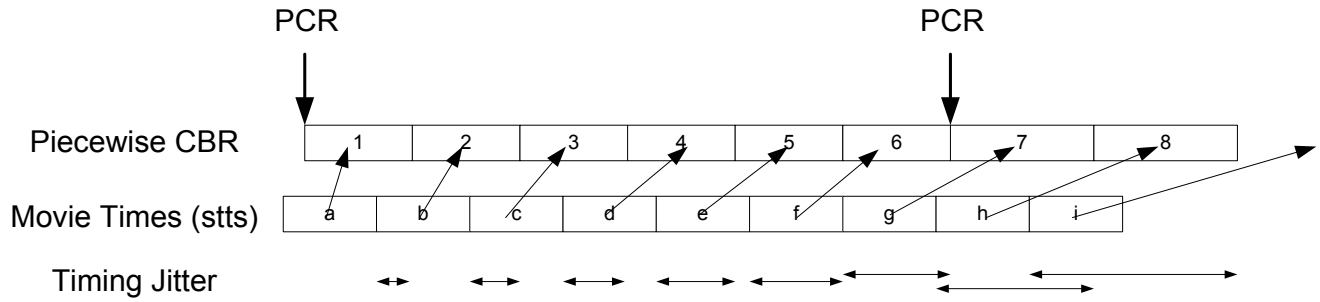
The DVB File Format specification defines a recommended maximum jitter of 40ms. It is also recommended that the delay is kept below 2 seconds.

The maximum jitter can be signalled via the "tjit" box in the track header. It is recommended that this box is present and contains the maximum potential jitter present.

NOTE 2: This maximum value may not be achieved. An implementation may choose to write out the maximum jitter value at which it will force a new entry for the sample duration, but this value may not be reached. When feasible, good practice involves re-writing this value with the actual maximum jitter value.

7.2.4 Implications of jitter

The above proposals can increase the timing jitter between the media time and the sample time to a level at which a given transport packet is mis-identified. An example of this is shown in Figure 12 below, where the top row labelled "Piecewise CBR" shows the transport stream system clock timing based on a piecewise linear constant bitrate interpolation between PCR samples, and the bottom line shows a potential movie timing using E-PLCBR. This shows how the jitter can result in a mis-identification of a sample, as the movie timing would indicate that the timing period *g* relates to sample *6* when in fact it should relate to sample *7.*



[4] The delay is introduced since the sample duration cannot be written until the end of the extended PLCBR period as it remains unknown until then. Although the sample data can we written into the file in advance of the metadata, the samples will remain inaccessible until the metadata is written.

Figure 12: Example showing large values of jitter

It is strongly recommended that implementations can cope with the jitter consequences outlined.

7.2.5 Jitter and synchronisation

In most cases of "`rm2t`" tracks, there will not be other tracks to be synchronised with which the reception hint track is synchronised, and so jitter is not a significant issue for synchronisation purposes.

However, where other tracks are to be synchronised with the MPEG-2 Transport Stream Reception hint track, the synchronisation should be with the presentation and display times of the content contained within the transport stream, as encoded within the PES headers.

7.3 Recording of transport streams

7.3.1 Introduction

A recording of a transport stream may result in a file that contains:

1. Pure transport stream packets only (a precomputed transport stream)
2. Media with no transport stream packets at all
3. Both Transport Streams and media
4. Transport Streams constructed from media tracks.

The first represents the most common mode of operation. Files containing pure transport stream packets only are the simplest to create when recording. Furthermore, such files can be handled with little or no additional processing by devices capable of demultiplexing and decoding plain Transport Streams.

The use of media tracks (options 2, 3 and 4 above) allows existing players compliant with earlier versions of the ISO base media file format to process recorded files as long as the media formats are also supported. Furthermore, files containing media tracks may be conform to both 'dvt1' and 'dvr1' brands and hence increase the likelihood of a file being playable by various types of DVB compliant devices.

Devices optimised for Transport Stream playback may not be able to efficiently support the final case, which requires the reconstruction of transport packets from media samples. As a consequence this mode of operation should be used with care considering the balance of usability of the file by devices that can only support transport streams and those that can only support media tracks together with the complexity of extracting media samples from transport stream packets. Discussion of this is given in clause 7.3.3.

NOTE: All dvt1 devices must be able to support media tracks.

Files containing both a precomputed transport stream reception hint track and respective media tracks (option 3 above) can be used to enable efficient playback of Transport Streams, while the interoperability to players without MPEG-2 TS reception hint track support is maintained. The downside is the increased size of the file, as all media data is practically duplicated in the file. When constructors are used in hint samples (option 4 above), the file size increase compared to storing the transport stream packets only (option 1) or media only (option 2) is moderate.

Storing the original transport headers (options 1, 3 and 4) retains valuable information for error concealment.

All but the first case may be limited to those media formats that have defined representations in the file format. For instance, neither MPEG-2 video nor DVB subtitles have defined encapsulations compatible with the ISO Base Media File Format on which the DVB File Format is based.

Table 2 below shows how the following clauses apply to the different modes of recording.

Table 2: Applicability of functionality to different recording modes

	Transport Stream Packets Only	Media Only	Transport Stream and Media	Transport Stream Packets Constructed from Media
Creation of Transport Stream Samples (clause 7.3.2)	Yes	No	Yes	Possibly
Creation of Media Samples (clause 7.3.3)	No	Yes	Yes	Yes
Creation of hint samples referring to media samples (clause 7.3.4)	No	No	No	Yes
Transport Stream Timing Jitter Trade-off Handling (clause 7.3.5)	Yes	No	Yes	Yes
PMT Update Handling (clause 7.3.6)	Yes	Possibly	Yes	Yes
Handling Box Size Limitations (clause 7.3.7)	Yes	Yes	Yes	Yes
Transport Stream Random Access Samples (Clause 7.3.8)	Yes	No	Yes	Yes

7.3.1.1 "rm2t" and "pm2t" track structure

Conceptually, a transport stream could be split into multiple tracks, one for each component of the program. However the strongly recommended mode of operation is with a single track that holds the multiplex components of the stream in their original format.

7.3.1.2 Protected tracks

Transport streams may be protected by CPCM [i.13] which is signalled by using the "pm2t" track type as defined in the ISO Base Media File Format Specification [i.3] and in the DVB File Format specification [i.1]. In the case of CPCM protected content, the information and recommendations discussed still apply, however as the content may be scrambled certain operations or the generation of data may not be possible.

It is possible that other proprietary protection mechanisms may also be applied, and this may be signalled with one of two mechanisms

- using the "pm2t" track type combined with proprietary extensions to signal the specifics of the protection mechanism, or
- via transport stream signalling such as the PMT and scrambling control bits of the transport packet samples (i.e. independent of the "pm2t" signalling).

Whenever possible, it is recommended that the "pm2t" mechanism is used.

7.3.2 Creation of transport stream samples

A transport stream sample in the DVB File Format consists of a single transport stream packet. This packet will normally be exactly as received—there is no requirement for the packet to be processed or adapted prior to storage in the file.

7.3.2.1 Sample size

It is recommended that the sample size used is 188 bytes[5] and that it remains fixed for the duration of the file. This allows for a single entry in the "stsz" box. However, for robustness, and to allow access to the file during creation, it is recommended that the count value associated with the sample size in the "stsz" box is continually updated. Alternatively, although not recommended, new entries are added[6] to the "stsz" box as new sample duration values are added to the "stts" box.

[5] This is not applicable for the creation of samples using the constructor mechanism as this requires variable size samples to reflect the number of constructors used to recreate the sample. In such cases the sample size will vary throughout the file.

[6] The disadvantage of adding new entries to the "stsz" box is that the "stsz" box will grow in size, and this in turn will limit the maximum duration of file that can be stored without exceeding the 1Mbyte size limit of the "moov" box.

7.3.2.2 Sample duration

The sample duration has been discussed above in clause 7.2.3 and the trade-offs in accuracy versus metadata size are discussed below in 7.3.5.

7.3.2.3 Start and end samples

It is recommended that the recording starts or ends with a sample containing a PCR. Given the PCRs should be present in every broadcast at least once every 100ms, this does not represent a significant alteration in the timing.

However, where this is not practical the recording should extrapolate the first duration period back to the start of the file, or the last duration on to the end of the file. The extrapolation should use the timing values from the following samples, for the start of the file, or the preceding samples, for the end of the file.

There is no implication or expectation that the media contained within the transport stream starts with or ends in random access media samples, or even at the start or end of a sample. This means that a device playing back a file will need to look through the reception hint samples until it encounters a media data that allows it to start playback. Equivalently, there is no expectation that the recording process starts with a transport packet that contains the start of a media random access sample.

7.3.2.4 Sample format

The transport stream packet sample starts with a header of 32 bits. This header carries a range of values describing properties of the sample. These include two scrambling control bits. There is no requirement that these values should be zero, and if they are not zero they represent a scrambled packet. In the absence of any other signalling, e.g. via the "`pm2t`" sample description, details relating to this scrambling will be included in the PMT that is applicable to this sample, as per normal practice for transport streams.

7.3.3 Creation of media samples from transport stream

In order to extract media samples from the transport stream, the transport stream must be descrambled as there is no defined mapping of the broadcast scrambling into a protected media stream (and it would be challenging to define such a mapping for a transport stream scrambling scheme). This either means that the transport stream must have been transmitted in the clear, or must be available to the recording process in the clear. As another example of the need for access to the descrambled content, creating media samples requires access to the timestamps present in the transport stream to enable synchronisation between tracks, e.g. to preserve lipsync.

Each component of interest with a defined media storage format that is present in the broadcast should be converted into its own media track.

NOTE 1: Some media encodings that are present in the transport stream, such as MPEG-2 video or DVB subtitles do not have a media track representation, and so cannot be stored in a media sample format.

NOTE 2: The reconstruction of reception hint samples (MPEG-2 Transport Stream Packets) from media samples may not be an efficient or easy operation, because it may require additional memory buffer to store the reconstructed MPEG2-TS packets, and reception hint samples and the respective media samples need not be stored adjacently and hence may require non-sequential file access.

7.3.3.1 First sample

The transport stream may not start with a complete media sample, or with a random access media sample. When the media samples are extracted for a media track, all sample data prior to the first random access sample should be ignored. This process will affect the media track start time, and should be accounted for using the mechanism discussed in 7.3.3.4.

7.3.3.2 Sample size

The sample size must be calculated by parsing the received data. Care should be taken to transform the received data as required for the storage specification. For example, an H.264 stream is broadcast in byte_stream format which has startcodes, but is stored in the file as a set of length delimited NAL units.

It is not reliable to assume alignment between MPEG-2 Packetised Elementary Stream Packets and the media samples, as there are cases where this relationship may not apply. Specifically, an access unit may not start at the beginning of a PES packet.

Transmissions can also include stuffing data to ensure buffer model compliance, and this can be represented in the media sample. Whilst this may be removed to save space when storing the media samples, in may increase the complexity of reconstructing transport stream reception hint samples.

7.3.3.3 Sample duration

The sample duration can be extracted from numerous sources, including an inspection of the media itself. The difference between packetized elementary stream (PES) headers can be used, together with parsing the underlying media into sample units, to identify the duration of samples.

Care must also be taken as there are numerous occasions when a single PES packet can include multiple samples, and where the number of samples per PES packet can change.

7.3.3.4 Compensation for different media track start times

When media samples are extracted from the transport stream, the PES timestamps specifying the presentation of the first media sample in each track will not be aligned. The following recommended method is suggested:

1) The track which has the earliest (first) presentation time is identified. This time is treated as zero in media time

2) For all other tracks, the delay required between their first sample and that of the track identified in step 1 is calculated.

3) For each other track, and Edit Box and Edit List Box is created to shift the media timeline to maintain the synchronisation present in the transport stream. This is done as follows:

   a) The first entry in the edit list box is an empty edit (media_time equal to -1), with a duration equivalent to the delay required as calculated at stage 2.

   b) The media_time of the second entry is set to the composition time (or presentation time in MPEG-2 transport stream terms) of the first entry in the track.

NOTE 1: This mechanism is identical to that described for extracting media samples from RTP recordings, as detailed in clause 6.3.2.

NOTE 2: The first sample in a media track should enable the start of decoding and hence be a sync sample.

7.3.4 Creation of hint samples referring to media samples

This section describes the issues related to generating hint samples that refer to media samples from received transport stream samples. The reverse process, that of recreating the original reception samples as received, can be inferred from the descriptions.

Use of hint samples referring to media samples is not recommended, as many devices may not be able to natively process this format.

When hint samples must be used, they should enable a perfect reconstruction (except for scrambling) of the original received transport stream packets.

It is to be expected that a mixture of constructor samples and pre-computed samples will be present in such a track. In a similar fashion, it is to be expected that a single sample will use a mixture of constructor types.

7.3.4.1 Adaptation data and PES headers

The hint sample provides details of the transport stream packet header. Any adaptation layer data in the original transport stream should be stored here.

The PES header, when present in the original received sample, should be stored as the first constructor associated with the reception hint packet, using the immediate constructor format, i.e. constructor type 1. This will then normally be followed by one or more MPEG2Sample constructors.

As necessary, additional or combined, immediate constructors may occur to hold data that is required for a transport stream encoding, but that are not present in the sample format (see 7.3.4.6).

7.3.4.2 Transport stream packets holding more than one media sample

A transport packet can hold data from more than one media sample. An example of this is where a PES packet holds multiple access units, where one access unit may end part way through a packet and be followed immediately by the next packet. In the case of extremely small media samples (such as supported by H.264), it is possible that more than two media samples will be present in a single transport packet. As a consequence there may be multiple constructors present in a reception hint track sample

7.3.4.3 Non-media sample data

Where a recording in a media track format has been made, there may be data for which no media representation exists, such as DVB subtitles, or makes sense to exist, such as SI and PSI data. Packets containing this data, that have been selected as part of the recording, should be stored as pre-computed media samples.

7.3.4.5 Sample size

The size of a hint sample that is constructed from media samples is determined by the number of constructors present, and their type. As such the reception hint track sample size is no longer 188 bytes, which means that the run length coding of the value table in the "`stsz`" box will not work as efficiently. This implies that the "`stsz`" box becomes larger than the "`stsz`" box for a precomputed reception hint track.

7.3.4.6 Differing media representations

The media encapsulation for certain codecs, notable H.264/AVC [i.14], differs between that used in a transport stream [i.12] and in an ISO Base Media File [i.15]. This means that creating media samples from the reception hint samples can involve codec specific processing. In the case of H.264/AVC this is a conversion from byte_stream NAL representation into the sample representation which involves:

- Searching for, and removal of, start codes (including removal of zero_byte values preceding startcode)
- Addition of NAL unit sizes
- Grouping of appropriate NAL units into single sample.
- Identifying and removing parameter set NAL units from the bitstream and storing them in the decoder configuration record of the sample description of the media track or in a parameter set elementary stream track.

7.3.5 Transport stream timing jitter trade-off

When creating the sample durations for the transport stream packets, using E-PLCBR increases the introduced jitter in exchange for the size of the "`stts`" box. When determining the duration between timing values, the following mechanism is recommended.

A new sample duration entry in the "`stts`" box should be generated when:

- The difference between the media time and the system clock reference time (as received in the PCRs) exceeds the threshold value, recommended to be 40ms, or
- When last entry was generated at a media time of more than 2 seconds ago (the delay constraint discussed in 7.2.3)
- At the end of the recording

7.3.6 PMTs, updates and sample description changes

It is recommended that the PMT is included in the sample description, as this simplifies processing by a file parser to ensure it has the necessary capabilities to decode and display content from a transport stream. As a PMT included in the sample description takes precedence over any within the transport stream, the recording process needs to continually monitor the received PMTs for changes.

The "`rm2t`" sample description (and by inclusion, the "`pm2t`" sample description) can include a copy of the PMT. This means that when a PMT change occurs, a new sample description is required to ensure a valid PMT data for the new segment.

A change in the PMT is signalled by a new version number for the PMT. It is recommended that when a new version is seen:

- the PMT is compared against the current PMT to ensure that changes have actually occurred, and
- any changes that occur are relevant to the recording (e.g. if they refer to a component that is not being recorded)

Only when a PMT with a new version meets the above conditions should a new sample description be generated.

If the recording device is excluding certain components from the file, or performing alterations to the received content, it is recommended that a version of the PMT is stored in the sample description box that that reflects the changes made to the recorded stream, i.e. that contains only the components present in the track.

7.3.7 Size limitations and use of "moof" box

The "moov" box of a DVB file has a 1Mbyte limit on its maximum size. As the "moov" box can hold details of samples sizes and durations, this imposes a limit on the maximum duration of a file. For example a 25Hz video stored as media will require a new sample size stored every 40ms; as the size is 32-bits, approximately 3 hours of video samples will require over 1Mbyte of size information.

Where a file needs to be generated that is larger than 1Mbyte the solution is to use movie fragments. This requires the presence of an "mvex" box (containing the relevant number of "trex" boxes) in the "moov" box. Therefore an appropriate amount of space should be reserved to allow for the insertion of this box, which can occur at the end of the "moov" box.

More details on the use of fragments is provided below in clause 8.

Other specifications may not have a maximum size of the "moov" box. Such a file should not be co-branded as a DVB file.

7.3.8 Sync sample table

A sync sample is one that contains an MPEG2-TS packet which in turn contains the first byte of an independently decodable media packet (e.g. MPEG-2 video I-frame, or an H.264/AVC IDR frame) for the primary media of the recording[7].

For an SPTS, it is recommended that an "stss" box is present and that it contains details of the sync samples described above. However, if it is not possible to determine the location of these samples, then the "stss" box should remain empty. In this case a playback application will need to search for these samples which may impact playback performance, hence the recommendation that the "stss" box is present. However, a playback application should not assume the presence of a populated "stss" box.

NOTE 1: The media time of a sample identified by a sample number entry in the "stss" box represents a version of its reception time and not its display time. For operations seeking a random access point based on display time, the timestamp associated with the sample associated with the entry in the "stss" box will need to be inspected to identify the appropriate random access point or sync sample.

If the stream is an MPTS, it is recommended that the sync sample table is present, but that it is empty.

For certain codecs such as H.264/AVC the random access indicator flag on the transport packet header of a video packet can be used to identify a suitable random access point more easily than scanning the content of the video packet.

NOTE 2: An application searching for a key frame can start reading at that location, but in general it also has to read further MPEG2-TS packets (that in the file format these are subsequent samples) so that the decoder can decode a complete frame.

[7] Typically, though not necessarily, this will be the main video component of the file.

7.3.9 Interaction with ISO-defined boxes

7.3.9.1 Hint media header box

In the hint media header box "hmhd", the maxPDUsize and avgPDUsize are the packet size which for MPEG-2 transport stream packets is fixed at 188, as specified in the MPEG-2 Transport Stream sample entry.

NOTE 1: As this is the size of the PDU, and not the sample, this value applies when constructors are used to re-create the transport stream packets.

NOTE 2: The size and length of the the preceding or trailing data is an implementation specific choice. However in many cases either, or both, will be zero length.

7.4 Playback of transport streams

7.4.1 Introduction

Playback of transport streams assumes that the track contains pre-computed samples. If this is not the case, then, in general, the samples exist as a media track and the playback of the media track format is likely to be simpler than reconstructing the samples for playback through the transport stream processing, assuming that all required components of the playback exist in media tracks. As such, there is a single playback mode discussed which applies to both the first and third cases of the four discussed in section 7.3 on recording transport streams.

At its simplest, the playback operation involves the reading of the transport stream samples from the file format and passing them into a transport stream capable playback engine. The rate at which samples are read from the file is paced by the decoding process, which operates at a rate determined by the playback device's clock. Ideally, the transport stream playback engine should be configured based on the PMT from the sample description.

7.4.2 Clocks and synchronisation

With a typical single track transport stream file, there is no synchronisation performed at the level of the ISO media file. Instead the synchronisation between the components is performed using information encoded into the transport stream in the timestamp headers of the packetized elementary stream (PES) headers. In this mode the media time of the file is primarily useful for approximate jumps through the file, or for synchronisation with other metadata tracks such as content descriptions.

Pacing of the playback is controlled by the clock on the playback device, with content read as required from the file.

7.4.2.1 Synchronisation between tracks

Where multiple tracks are present in the file, the synchronisation between the tracks should be performed using the presentation time-line as governed by the Edit Box, if any. There is no requirement that the system clocks between the tracks are synchronized. It is possible that the clocks between the tracks operate at marginally different clock rates. The player selects one of the tracks as a master track, and other tracks will effectively be re-clocked to match the rate of the master track[8].

7.4.2.2 Media time, PES timestamps and synchronisation

For a transport stream track, the decoding time (as defined in the ISO Base Media file, and not that encoded into the PES timestamp) of the transport stream samples relate to their reception time as measured by the reconstructed PCR from the transmission, but starting at with the value zero at the start of the file. Therefore the decoding time is not the value encoded in the PES headers of the flows within the transport stream (i.e. the PES presentation and decode times). The offset between the decode time (reception time) and the presentation time, that is the PTS values encoded in PES headers, is usually the buffering time of the original transmission[9]. As such this offset is constrained by the various definitions of the buffers for the codecs, and will vary during the recording.

---

[8] The requirements on the relationship between media time and the MPEG-2 Transport System time prevent two tracks from operating at different clock rates.

[9] When the TS is carried over RTP or HTTP the buffering time, the difference between arrival time and display, may differ from that implied by a piecewise linear CBR interpretation or that of a TS stream delivered and received natively.

Synchronisation between components within a transport stream must be performed on the basis of the timestamps encoded into the PES in which the components are transmitted, rather than at the decode time which represents when the transport packet samples were received. Similarly, should synchronisation need to occur between tracks, this synchronisation should be based on composition times mapped to the movie timeline with Edit List boxes, if any. A composition time for a transport stream packet is the presentation time encoded into the PES subtracted by the very first presentation time of the track. No "ctts" boxes or sample_composition_time_offset in "trun" boxes are used to indicate composition times for reception hint tracks, but rather composition times are implied from the PES and Edit List boxes, if any, if there is a need to synchronise between tracks.

7.4.3 Handling of PMTs and multiple sample descriptions

Although the PMT may be present in the transport stream, the creator of a file may have stored a modified version in the sample description. An example of the reasoning for this may be to remove unrecorded components or unnecessary CA information. Thus if a PMT is found in the sample description, the processing of the PMT should be based on the one contained in the sample description and not those found in the media samples.

When playing back content and a new sample description is encountered, it does not necessarily mean that changes are necessary to the media decoders that are in use. PMT changes can signal changes to components that are not in use, or to a subset of components, or indicate that new components are now available. It may well be the case that the audio and video decoding processes can continue unaltered between different sample descriptions. When a sample description changes, an implementation should check to see if any alterations to the display and decoding are required, and attempt to maintain a seamless playback.

Where content is protected by CPCM, changes to the CPCM protection can also result in changes to the sample description. Assuming that these changes do not indicate that playback must be altered, a device should cope seamlessly with CPCM changes.

It may be beneficial for a playback device to study upcoming sample descriptions in advance of their being referenced by a sample. This may improve behaviour when a sample description change occurs.

7.4.4 Presence of errors

The recording contained in the file may contain errors that occurred during the recording. This may result in jumps in time value, missing samples, discontinuities in the transport stream, or even errors that were not detected at the transport level (but that manifest themselves during decoding). The playback device should take all steps to be tolerant of errors in the file. The device should not rely on having errors signalled in the file.

7.4.5. Handling jitter

The "`stts`" box (and equivalent information in the Movie Fragment box or boxes) is used to provide a mapping between the file's movie time and a given transport packet. This means that a range of time values will map down to a given transport packet. However, at times corresponding to the start of end of a packet, the mapping may identify the wrong packet (for small jitter values, this will be the preceding or following packet), as discussed in section Y.2 above. Thus when using decoding time, as derived from the "`stts`" box (or equivalent information in the Movie Fragment box or boxes) to identify a specific packet, data should be read in a block that includes samples in advance of the media time indicated value based on the jitter value signalled in the "`tjit`" box (or using the maximum value of 40ms). The block of data read should be searched for actual PCR values. These values can then be used to calculate the actual packet timing (subject to the PCR to media time offset value).

7.4.6 Fragmentation support

Fragments may be used for multiple reasons, including overcoming the 1Mbyte "`moov`" box size limit discussed above in 7.3.7, and for allowing simpler and more robust file growth as discussed by in section 8. A playback device should support files using only fragments, not using fragments and those that only use fragments after a period of non-fragment content. The playback device should seamlessly move between fragments and into fragment playback from non-fragment playback. Further discussion on fragments is given in section 8.

7.4.7 Sample location

The ISO Base Media File Format structures (on which the DVB File Format is built) that indicate the location of transport packet samples within the file should be followed. Specifically, there should be no assumptions that the transport packet samples are always contiguous in the "mdat" box(es). Samples are only contiguous within a chunk. It is perfectly valid, and indeed very possible, that the "mdat" box(es) may contain data (chunks) from other tracks such as metadata tracks interleaved with the chunks from one track. Also, the requirements of efficient transfers may well have resulted in "holes" between chunks in the "mdat" box(es).

For a discussion on Sample Location in Fragmented Movie Files see clause 8.3.7.

7.5 Retransmission of transport streams

Retransmission of the transport stream is the use case where the transport stream is pushed to another device at a real-time transfer rate, where the transport stream is not encapsulated within the DVB File Format. In this mode of operation, to prevent the receiving device suffering overflow or underflow of the decoder elementary buffers, it is important that the transmitter and receiver operate with locked clocks. As the receiver should derive its clock from the received transport stream, provided the transport stream is transmitted locked to the transmitting device's clock, then both devices will operate in a synchronised fashion.

Since the receiving device locks to the values in the transport stream, the sending device should pace the transmission of the transport stream packets (the samples from the file) based on the actual PCR samples contained in the track, and not the media time of the file (as this would introduce a jitter between the devices). However, as per the MPEG-2 systems specification [i.12], the piecewise linear constant bitrate method should be used to interpolate timing values between PCRs to support accurate transmission time of the transport stream packets.

It is recommended that the transmitting device ensures that both the PAT and the PMT are present in the transmitted stream, re-inserting them into the stream as necessary. Where these tables are inserted, it is recommended that they reflect the components and properties of the transmitted stream.

7.6 Multi-program transport streams

Storage of multi-program transport streams (MPTS) is not recommended. Instead, storage of multiple files, one per program, is the preferred mode of operation.

When a MPTS must be stored, the PAT should be present, preferably in the sample description. As with the PMT, it is recommended that this is modified to reflect the programs present in the stream. If there is a preferred, or default, program to display when this file is played, it is recommended that this is the first program listed in the PAT.

7.7 Use of external sample locations

The ISO Base Media File format allows media data to be stored in separate files, via the use of "dref" box contained in the "dinf" box. When applied to MPEG-2 transport streams, this allows a file containing the transport stream packets to be referenced by a DVB file, without the need to include the transport stream packets in either "mdat" boxes or in the DVB file itself. Such functionality is useful in several cases, including where legacy support for files containing only transport streams is required, or where a file includes both media samples and reception hint samples, and wishes to separate those for efficiency reasons.

However, when creating DVB files that make use of external files for sample storage, the benefits of this approach should be balanced against the risk of the DVB file and the external sample storage file becoming separated or one or other becoming lost. Likewise, the choice of the URL should be carefully considered, but a normal recommendation is that this is of the form "file:" and a relative position URL is used, e.g. "file:./tsdata789.ts".

NOTE: Care should be taken with relative filenames as they can present security issues, especially where relative filenames that go "up" the hierarchy.

Players that encounter the use of external locations for sample data should be robust against the lost of external sample data files. If the external data is spread over several files and not all files are missing, the player should play such content as is available. When making use of a relative position URL contained in the "dref" box, the player should take into account the source route of the DVB file, and amend the URL as required to form the actual data location.

8 Movie fragments

8.1 Introduction

Movie fragments are an essential part of the ISO Base Media File Format on which the DVB File Format is built. They provide key functionality enabling efficient (or more efficient) solutions to a range of use cases some important examples of which are described below. At the highest level, movie fragments provide a means of dividing a file up into largely self-contained units that hold both the samples and most of the associated data that varies with samples.

This clause provides:

- Some example use cases showing the potential usage and benefits of movie fragments
- Guidelines on creating fragmented movies
- Guidelines on reading fragmented movies

The remainder of this sub-clause provides a brief introduction to the key boxes used in the implementation of movie fragments, and references some other ISO Base Media File Format derived specifications that make use of fragments.

8.1.1 Movie fragment boxes

The following boxes are defined by the ISO Base Media File Format [i.3] and implement the movie fragment features:

- "mvex", "trex" and "mehd": the "mvex" box is present in the "moov" box to signal that movie fragments may be present. It is recommended that this box is *only* present when fragments are used. The "mehd" box contained within the "mvex" box is optional.
- "moof", "mfhd", "traf", "tfhd" and "trun": these boxes can, among other things, hold the information normally stored in the "stsz", "stts", "ctts" contained in the "moov" box. However, the form used to store this information in the "moof" box is different, and default values may be taken from the "mvex" box.
- "mfra" and "tfra": these boxes provide random access to the fragments, and are stored only at the end of the file. This location presents some challenges, and as such they may not be used, or may be superseded by alternative mechanisms.

8.1.2 Typical data layout of fragmented movie file

Figure 13 below shows part of a potential example fragmented file. Only one fragment is shown in detail in this picture, and this fragment contains samples from both tracks in the file, i.e the movie fragment contains two track runs. Each track fragment, in turn, consists of two track runs, which are colour coded to show the samples they refer to. The "moof" box is immediately followed by the "mdat" box that contains the samples indicated by the track runs. As with ordinary files, this shows how track runs can be used to interleave samples from different tracks in a fashion analogous with the use of chunks.

The picture also shows an absence of samples referenced from the "moov" box, and shows additional fragments and their sample data following the first fragment that is shown in more detail.

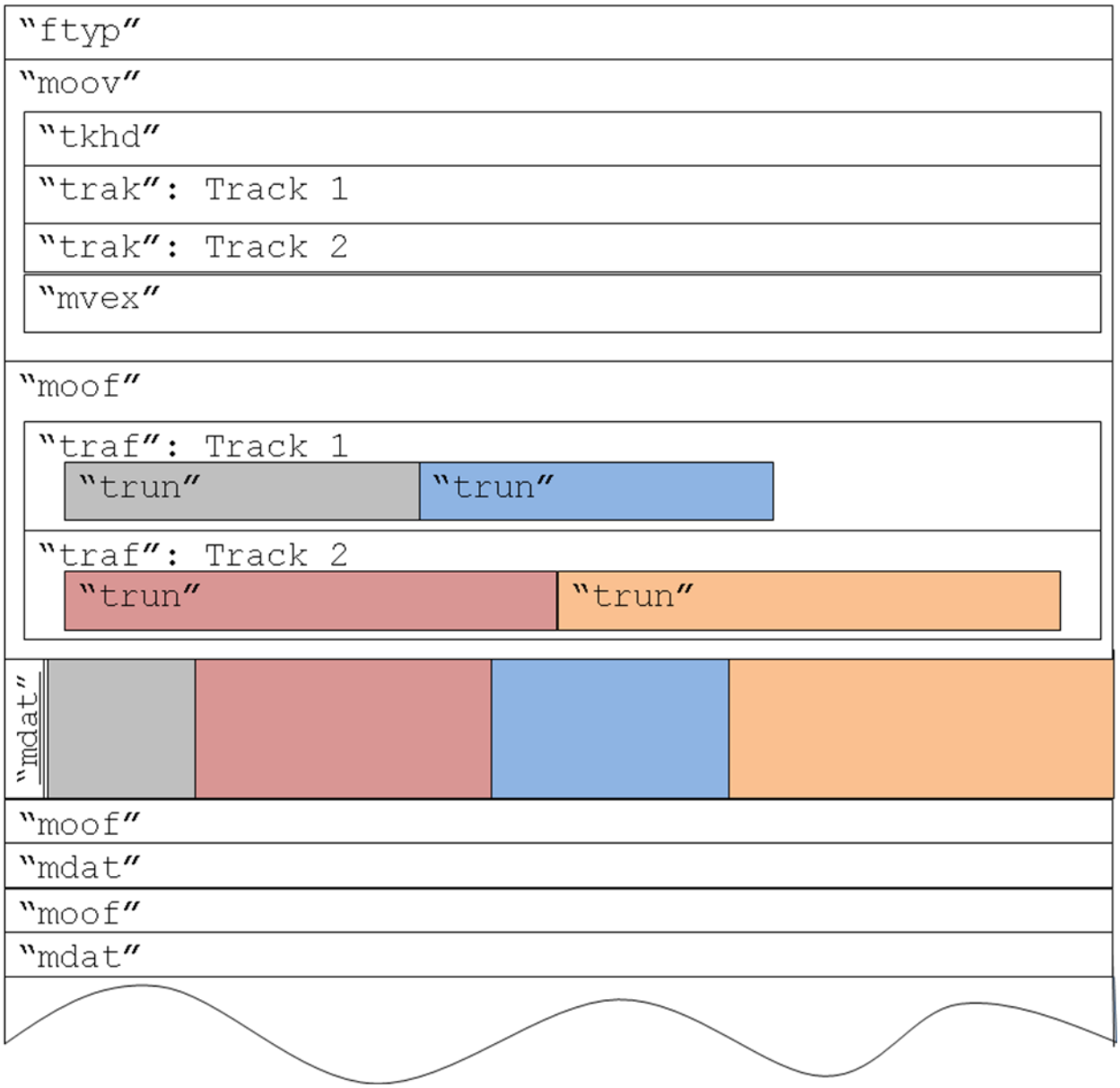


Figure 13: Example Layout of Fragmented File

8.1.3 ISO guidance on fragments

In Annex C.7 of the ISO Base Media File Format [i.3] contains basic recommendations on the construction of movie fragments. Whilst it is recommended that these are read, they need not always be complied with since:

1) the file may contain reception hint tracks rather than media tracks,

2) the "`mdat`" box(es), if used, may not be located in the file with the "`moof`" box (though normally it will be),

3) the first "`moof`" box sample may not be a random access sample.

8.1.4 Movie fragments in other standards

Movie fragments are explicitly referenced in a number of other standards. The 3GPP file format [i.2] allows the use of movie fragments, but does not recommend to use them for the Basic Profile due to compatibility with 3GPP Release 4 and 5 devices. Moreover, the adaptive streaming profile of the 3GPP file format, meant for adaptive streaming over HTTP, mandates the use movie fragments.

Other standards connected with the ISO Base Media File Format that reference movie fragments include:

- DECE's Ultraviolet
- DLNA's Media Format Profiles [i.18]
- PIFF, Microsoft's Protected Interoperable File Format

8.2 Example fragmentation use cases

8.2.1 Overview

This section provides some sample use cases that illustrate situations where the use of fragments is beneficial in comparison to non-fragmented files. Although the following indicate some use cases, it is likely that fragmented movie files will be used for all recordings by at least some recording devices.

8.2.2 Recording robustness

A recording can fail at an arbitrary point in advance of its normal completion, in a fashion that does not allow for the recording devices to generate a complete and clean file. The most common example of this is a power failure, but others may include (unexpected) lack of disk space or even software failures. With non-fragmented movie files, the samples that are recorded into the file can be accessed only via the information stored in the metadata. At best, when a failure occurs, a certain amount of the written data will be lost. However, the requirement to update this metadata may, at worst, result in the whole file being unusable if this metadata becomes corrupted. Likewise, some implementations of non-fragmented movie files may choose to only write the metadata at the completion of the recording, resulting in a loss of the entire recording when recordings do not complete successfully.

By comparison with a normal movie file, a fragmented movie file can be thought of as a collection of self contained parts of the file, since each fragment is self contained. Thus fragments place no requirement to update or alter existing data[10], and can be written sequentially to the file leaving existing data intact. This process means that in the event of a failure, the worst case data loss will be the final fragment that is written, and by selection of short duration fragments, this loss can be contained to an acceptable solution.

8.2.3 Chase play and file growth

A common feature of many systems is chase play where a file is read and its contents presented whilst the file is still being written. An example of this is where a viewer has started viewing a program before the program has finished recording. In today's systems, generally, it is not considered acceptable to prevent viewing of a recording until it has completed, so the playback device must be able to extract and display content from a file whilst this file is still having content added to the file.

When non-fragmented movies are used, this requires the "`moov`" box to be shared and continually updated between the reading and writing processes. Especially if these processes are on different devices, this can be challenging or costly. The effectiveness of sharing a file for this operation is highly depended on the precise details of both the way the file is managed by the writing process (e.g. are items "shifted" in the file) and the underlying filesystem support.

By comparison, using a fragmented movie to support chase play is relatively simple, where fragments are written as a single operation in and a completed fashion. Similarly, the support from the underlying filesystem or sharing mechanism is potentially simplified.

8.2.4 Large or long duration files

In the DVB file format specification, and several other file format specifications, size limits are either recommended or imposed on various boxes such as "`moov`". In the case of the DVB file format there is a size limit recommended of 1Mbyte for the "`moov`" box and 300kbytes for the "`moof`" box. Given that these boxes contain details on samples, this overall duration of a file that does not use movie fragments is limited. Whilst this size is sufficient for many files, it will not necessarily be large enough for all recordings. Movie fragments provide a solution to the size limits of the "`moov`" box by removing from the "`moov`" box all the items of metadata that grow as the number of samples grow.

---

[10] The one exception is where updates to the sample description are required, an update to the "moov" that holds the sample descriptions are needed. However, these are expected to be rare operations.

8.2.5 Adaptive streaming interaction

There are several methods for transferring content over the Internet, the "over the top" delivery method. Some of these make use of movie fragments to allow small, self contained sections to be transferred, and for those sections to be replaced with lower or higher bitrate versions to match the delivery bandwidth available to the target destination. Whilst it is unlikely that a device will create the multiple bitrate versions of a fragment that are at the heart of adaptive streaming, both the ability to playback fragmented movie files, and the ability to provide fragments for transfer allow for better and easier integration adaptive streaming functionality and devices that support or expect this format.

8.3 Guidelines on creation of fragmented files

8.3.1 "moov" box

8.3.1.1 Introduction

The "moov" box contains signalling indicating the potential presence of "moof" boxes, and fragments should not be used without this signalling. Specifically, the use of movie fragments in a file is signalled by the "mvex" box, which itself contains track specific fragment data in the "trex" box.

8.3.1.2 "mvex" box

Where a file makes use of movie fragments the "mvex" box must be present, and it is recommended that it is placed following the "trak" boxes. It is recommended that space is reserved, using a "free" box in the "moov" for an "mvex" box and the "mehd" and all the "trex" boxes it may need to contain. This applies even if fragments are not initially used in the creation of the file.

NOTE: The reservation of this space greatly simplifies the addition of fragments, should this be needed later on in the life of the file.

8.3.1.3 "trex" and default values

The "trex" box must be present for every track in the file which is to be extended. It provides default values for a variety of properties for sample values, such as size and duration. It is recommended that these values are correctly populated and used with the most common values to ensure that the most efficient use of space is made within the "moof" box. Since, at creation time, the most common value may not be known, either the "best guess" of a value should be made (where this can be done with reasonable confidence) or the value should be left unused.

NOTE: There is no explicit method to signal an unused value.

The values of the "trex" boxes should not be changed once they are written. Changing these values would cause significant problems for chase play operations, and require potential updates for all "moof" boxes that have been written.

8.3.2 Mixed "moov" and "moof" located samples

Where a recorder expects to use movie fragments, it is suggested that sample timing and sample size metadata is contained only in "moof" boxes and not in the "moov" box, i.e. the "stts", "ctts" (if present), "stsz", "stss" (if present), "stsc" and "stco" or "co64" boxes (and related boxes) are empty and the movie box contains no samples in its tracks.

8.3.3 Data layout and ordering

As discussed in Annex C of the ISO File Format specification [i.3], the layout of a fragmented movie file should start with the "ftyp" and "moov" boxes, and then followed by a sequence of "moof" and "mdat" pairs. There should be no boxes that occur between the "moof" box and its matching "mdat" box. Where additional boxes are required, they should be placed after the "mdat" box.

As implied by the ISO File Format specification, the sequence of "moof" and "mdat" pairs are time ordered as the sequence numbers in the "mfhd" box should increase through the file.

Where additional metadata is stored in the file, such as the DVB mandatory metadata, this should be placed before the first "moof" box, where possible.

The sample data for movie fragments should be located in the same file as the sample data for the "moov" box. In some cases, the sample data for a movie fragment may be stored in a file separate from the file(s) containing the preceding sample data. This is achieved by placing a "dinf" box with matching "stsd" box, and a new sample_description_index in the "tfhd" box. Sample data stored externally in this fashion need not be held in "mdat" boxes.

8.3.4 "moof" contained boxes

8.3.4.1 "mfhd" box

It is strongly recommended that the sequence number both starts at 1 and increments by 1 for each fragment.

8.3.4.2 "tfhd" box

It is recommended that where possible, default values from the "trex" box, as discussed in 8.3.1.3 are used. Where this is not possible, i.e. where values may change between movie fragments or are not know when the "trex" box is created, but that remain constant for samples in this fragment, it is recommended that they are signalled as default values and stored as such in the "tfhd" box. This helps to minimise the size of the "trun" box.

It is recommended that the base_data_offset field is not present, as this implies that all samples are referenced from the first byte of the "moof" box that encloses the "tfhd" box.

NOTE: This recommendation does not, and cannot, apply where the movie data is stored in an external file.

8.3.4.3 "trun" box

When recording a DVB file that consists of a single track it is recommended that the sample data for track runs are stored sequentially and in the same order as the "trun" box(es), i.e. without free space between the track runs being present.

8.3.5 Size and time duration

8.3.5.1 Generic guidelines on size and time duration

Although the specification provides a maximum size for the "moof" box, it makes no other size requirements, such as minimum sizes or on the size of the "mdat" box. This is deliberate to allow for flexibility: e.g. to avoid the need to rewrite the final fragment if the file has content added to it that would be required if only the final fragment was allowed to be less than a specified size. However, the use of fragments can have a significant impact on various aspects of performance. The following guidelines are intended to assist in the generation of efficient files.

There is a common size limitation on the "moof" box of 300kbytes. This size constraint is not a serious issue, since when the limit is approached a new "moof" box is started.

When operating in chase play mode, as discussed in 8.2.3, the time period between the recording starting and the first time at which playback can start is limited by the time duration of a movie fragment. Another impact of chase play is if the time duration of a movie fragment varies over the duration of the fragment. This can cause playback to reach, read and consume a fragment before the writing process has written the next fragment.

Thus it is recommended that fragments are written at a consistent, constant duration. A suggested value for the constant duration is application dependent, but it is suggested to be between 1 and 10 seconds, and it is recommended that this constant value is no larger than 1 minute, and no less than 1 second. This ensures that the "moof" box limit is not exceeded, unless there are an abnormally large number of tracks.

NOTE 1: The two second recommendation matches that of certain adaptive streaming solutions. Since these solutions prefer slightly shorter fragments to allow for faster adaptation to changes in delivery networks, certain implementations may choose a slightly longer fragment duration.

NOTE 2: An implementation may choose different constant durations for different values, reflecting knowledge of the stream. For example, a high bitrate HD stream may have a shorter duration than a low bitrate SD stream. The variation in durations may allow for more efficient use of memory and of transfers between the recording process and the DVB file.

8.3.5.2 Movie fragment sizes and MPEG-2 transport streams

In the case on an "rm2t" track (and its equivalent "pm2t" track), the bitrate timing calculations should be performed over a single fragment, i.e. they should not straddle fragments. This means that ideally fragments should start with PCRs, and the duration calculations to use the first PCR in the following fragment. Ideally, a single "sample duration" value should be used for each fragment, and that this duration should be recalculated for each fragment.

Where use of a single timing value for the entire fragment would break the timing jitter limits described by either the DVB File or the DVB File Format, two, or more durations need to be used. If this was to be done with a single "trun" box, this would require that each sample of that track in the fragment to have its own time. By using two "trun" boxes, one can use the default value for all samples associated with that track run, and only those samples in the other "trun" box require their own duration values. This is the preferred solution, where the default duration represents the most common duration.

NOTE: Splitting the fragment into two is not recommended as this would introduce variable sized fragments which can introduce problems for chase play, as discussed above.

8.3.6 Tracks and movie fragments

In structuring a file based on movie fragments there are a number of choices. For example, a movie fragment may contain fragments from all tracks, or it may contain only a subset of tracks. When structuring the file, the following recommendations are made:

- Each fragment contains all tracks currently active

NOTE 1: It is possible that some tracks do not exist for the full duration of the movie.

- Each fragment, within the timing constraints of the samples, contains tracks that are time aligned.

NOTE 2: However, a reasonable mode of operation for recording of DVB-H broadcasts is to create one movie fragment per one Application Data Table (ADT). It is therefore up to the service provider how accurately the different streams are time-aligned within ADTs.

- All fragments are approximately the same duration.

NOTE 3: If a file is subsequently has content appended to it, the final fragment of the original recording will no longer be the final fragment. To support such use cases it is beneficial if the final fragment is extended to the same duration as other fragments.

- The number of track runs (i.e. "trun" boxes) is kept to a minimum whilst also minimising the size of the "traf" box.

NOTE 4: This is not intended to mean that there is necessarily a single "trun" box. As discussed in 8.3.3 above, the most efficient solution may be with two "trun" boxes. Where, for instance, no sample values change during a track run, or where every sample requires a distinct value, then it is expected that a single track run would be present. However, where there is a single common value set that is used by adjacent samples, then there would be two or three track runs, one for the common value set using the defaults in the "tfra" box, and one or two with the other, less common values.

When creating a file that may be used for chase play, creating a file only after there are two or more fragments that can be placed into it improves chase play behaviour. This is because guaranteeing a number of fragments in a file before playback can start minimises the chance of chase play starting sufficiently close to the recording time and thereby cause stuttering or failed playback.

Fragments should also be written out to the file as soon as they are completed. If they were buffered, it is more likely that a playing client could catch up to the end of the file and either incorrectly treat the recording as finished, or have to pause playback.

8.3.7 "moov" box

The use of fragments means that the "moov" box no longer contains the boxes that typically grow as samples are added to a file (e.g. "stss" and "stsz" boxes). However, this does not mean that the "moov" box is necessarily static as the file grows. There are a number of events that can require additional information to be stored in the "moov" box:

- Addition of new tracks, for instance when a new component starts during the recording of a broadcast such as a new language track.
- A new sample description for a track, for instance when the PMT of an MPEG-2 TS is updated.
- A new secondary file (in cases where data is not all stored within a single file), for instance if an MPEG-2 TS recording is split between multiple files.
- Updates to sample protection information (in schemes that use "schi" boxes), for instance where changes in CPCM permissions require a new license or auxiliary data.

It is useful to use a "free" box to reserve space for growth of track information at the appropriate location(s) in the "moov" box.

8.3.8 Sample location

When comparing the layout of samples in a fragmented file with that of a non-fragmented file, a single track run is equivalent to a chunk. Specifically, a track run comprises of a sequence of contiguous samples of the same track. As with chunks, track runs can be used to interleave samples from different tracks for more efficient transfers. However, given the small recommended size of fragments, the number of track runs should be fairly small. Very small track runs are not efficient, and should be avoided where practical.

It is recommended that the "mdat" box containing the data for a "moof" box immediately follows the "moof" box.

8.4 Guidelines on playback of fragmented files

8.4.1 Introduction

This sub-clause provides guidelines when playing back fragmented files. Several items covered in the previous sub-clause, 8.5, are also informative when considering playback. However, as mentioned below, the playback process should be as tolerant as possible, and not assume that the recommendations are necessarily followed.

8.4.2 Identification of fragmented movies

Although all files that make use of movie fragments should signal this through the presence of a "mvex" box in the "moov" box, a playback process should be tolerant of the absence of the "mvex" or "trex" boxes and where there is no missing information still be able to playback fragmented content. Thus, ideally, a file should be checked for the presence of "moof" boxes, though this needs only be done as the media that is indexed from the "moov" box is exhausted.

8.4.3 Time alignment between tracks within a fragment

As samples in different tracks may have different durations, it is not possible to make a requirement that all tracks within a fragment start at the same time. To maintain time alignment between tracks (e.g. to preserve lipsync) it is important to maintain details of the timings of each track at the end of the previous fragment, and use these timings as the start timings for the next fragment.

Where a file consists of exclusively fragments, the same is true of the first fragment. In this case the relative alignment of the tracks may be signalled by the presence of edit boxes, as discussed above in clauses 6.3.2 and 7.3.3.4.

Where random access is used all previous fragments must be scanned to preserve accurate synchronisation between tracks.

Mechanisms for carrying relative track alignment timing information are provided in boxes introduced by 3GPP TS 26.244 release 9 specification [i.2], specifically the "tfad" box. It is recommended that these boxes are used where possible.

8.4.4 Chase play operation

When playing back a file, the playback operation should be able to cope with chase play. Practically, this means that it should allow for a file that grows during the process of playback.

The clock of the playback device may differ from that of the received stream, and as such it is possible that the playback process will consume and present information from the file faster than it is written. This means that it is possible that a playback process will catch up with the end of the file. Unfortunately, without additional information from the recording process to the playback process that is external to the file, it is not possible to determine whether the recording has completed, or if data will be written in a short time period. To minimise the chances of this occurring, it is recommended that playback does not start until at least two, and ideally more, fragments are present in the file. By ensuring there is a reasonable buffer of media samples, the differences in playback rates are unlikely to result in playback completely catching up with writing.

NOTE: The presence of the "mfra" box or "mfro" boxes should not be used to indicate a completed file. Firstly they are not required to be present, and secondly, they may be continuously updated by overwriting during the creation of a file.

When a playback starts only when a number of fragments exist before the end of the file, this can be used to increase confidence that the end of the file has really been reached. If the end of the file does not change over, for example, the last four fragments of playback where playback started with five or more fragments in the file, it is almost certain that the writing process has stopped.

8.4.5 Supporting mixed "moov" box and "moof" box based files

Although it is recommended to create files that either exclusively use movie fragments or use the "moov" boxes, a playback device should cope with a file that starts with content described in the "moov" box, but that subsequently switches to using movie fragments. This switch should be handled seamlessly.

As discussed in 8.4.3, it is not possible to assume that tracks in a movie fragment start at the same time, as is the case between fragments. Therefore the "moov" box will provide details of the relative time alignment of the tracks through the end positions of the tracks when the finish in the "moov" box. This will include any initial edit boxes.

8.4.6 Flexibility of parsing

As mentioned earlier, flexibility over the range of inputs is important. A non-exhaustive list of some key areas of flexibility follows.

- Although it is recommended that "moof" boxes are kept under 300kbytes in size, there may be instances where a "moof" box is larger than 300kbytes and supporting these will widen the range of compatible files that the player may successfully process.
- The number of "trun" boxes within a fragment can vary, especially for longer duration fragments, as such there should be no playback limits to the number of "trun" boxes.
- Although fragment durations are typically short, the player should cope with long duration fragments.

History

Document history		
<Version>	<Date>	<Milestone>

2010-05-07