File Format Specification for the Storage and Playback of DVB Services

DVB Document A121

June 2008

Contents

Foreword

Founded in September 1993, the DVB Project is a market led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG 2 based digital television services. Now comprising over 280 organizations from more than 25 countries around the world, DVB fosters market led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

Introduction

This present document provides specifications for the format of a file used to store DVB services, the "DVB File Format".

The specifications provided in this present document provide for a file that can contain the full richness of experience that is offered by DVB compliant services. The file may be used to contain content received from a transmission, or it may contain content authored direct into the file which has been delivered by a mechanism out of the scope of this document.

The file format is derived from the ISO base media file format [30], and as such provides much commonality with many other similarly derived file formats, such as the MP4 file format [31] or the 3GP file format [15]. Indeed, where appropriate, the file formats are interoperable to allow for the maximal mobility of content.

The format allows the storage of audio, video and other content in any of three main ways:

- encapsulated in a MPEG-2 Systems Transport Stream [27], stored as a reception hint track;
- encapsulated in an RTP stream, stored as a reception hint track; and
- stored directly as media tracks.

In addition, optional mechanisms are described that allow for the conversion of content between reception hint tracks and media tracks.

To provide useful descriptive information to the user of what is contained within the file, a small amount of mandatory descriptive metadata is included in the file. This allows any device to get a description of the content, even if it is not able to display the content. Additional metadata can be provided either in the extensible mandatory format, or in formats specific to certain DVB specifications.

Content contained in files that follow this specification may be protected by CPCM [14]. This allows the DVB File Format to be used for the storage and retrieval of CPCM protected content.

1 Scope

The present document specifies the File Format used to store and playback DVB services. The audio/video properties of the services that may be contained in this format are defined by specifications TS 101 154 [3] for MPEG-2 Systems services, and TS 102 005 [4] for RTP carried elementary stream services. This present document also specifies the storage of a range of additional data relevant to the services. This includes:

- specification TS 102 825 [14] when used to protect MPEG-2 Systems services;
- TV-Anytime Related metadata, as defined by TS 102 471 [7], TS 102 323 [5], TS 102 539 [10];
- any data that is carried with a transport stream as part of the service; and
- 3GPP Timed Text [16].

Through the defined brands, this specification provides definitions of what may be contained in the file. However, the specification does not preclude any extensions from also being present, provided they do not conflict with the requirements of this specification.

This specification does not require that the File Format be used internally by a device, and as such the specification can be viewed as an interoperability point.

The use of the File Format is outside the scope of this present document.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific.

- For a specific reference, subsequent revisions do not apply.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

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

For online referenced documents, information sufficient to identify and locate the source shall be provided. Preferably, the primary source of the referenced document should be cited, in order to ensure traceability. Furthermore, the reference should, as far as possible, remain valid for the expected life of the document. The reference shall include the method of access to the referenced document and the full network address, with the same punctuation and use of upper case and lower case letters.

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

[1] ETSI EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[2] ETSI ETR 162: "Digital Video Broadcasting (DVB); Allocation of Service Information (SI) codes for DVB systems".

[3] ETSI TS 101 154: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in Broadcasting Applications based on the MPEG-2 Transport Stream".

[4] ETSI TS 102 005: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in DVB services delivered directly over IP protocols".

[5] ETSI TS 102 323: "Digital Video Broadcasting (DVB); Carriage and signalling of TV-Anytime information in DVB transport streams".

[6] ETSI TS 102 366: "Digital Audio Compression (AC-3, Enhanced AC-3) Standard".

[7] ETSI TS 102 471: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Electronic Service Guide (ESG)".

[8] ETSI TS 102 472: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols".

[9] ETSI TS 102 474: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Service Purchase and Protection".

[10] ETSI TS 102 539: "Digital Video Broadcasting (DVB); Carriage of Broadband Content Guide (BCG) information over Internet Protocol (IP)".

[11] ETSI TS 102 591: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols (CDP) Implementation Guidelines".

[12] ETSI TS 102 822-3-1: "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 - Metadata schemas".

[13] ETSI TS 102 823: "Digital Video Broadcasting (DVB); Specification for the carriage of synchronized auxiliary data in DVB transport streams".

[14] ETSI TS 102 825: "Digital Video Broadcasting Content Protection & Copy Management (DVB-CPCM); Part 4: CPCM System Specification".

[15] ETSI TS 126 244: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP) (3GPP TS 26.244 version 7.2.0 Release 7)".

[16] ETSI TS 126 245: "Universal Mobile Telecommunications System (UMTS); Transparent end-to-end Packet switched Streaming Service (PSS); Timed text format (3GPP TS 26.245 version 7.0.0 Release 7)".

[17] ETSI TS 126 290: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); Audio codec processing functions; Extended Adaptive Multi-Rate - Wideband (AMR-WB+) codec; Transcoding functions (3GPP TS 26.290 Release 6)".

[18] IETF RFC 4234: "Augmented BNF for Syntax Specifications: ABNF".

[19] IETF RFC 4281: "The Codecs Parameter for 'Bucket' Media Types".

[20] IETF RFC 4566: "SDP: Session Description Protocol".

[21] IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings".

[22] IETF RFC 3550: "RTP, A Transport Protocol for Real Time Applications".

[23] ISMACryp: "Internet Streaming Media Alliance Implementation Specification – Encryption and Authentication".

[24] ISO 639-2: "Codes for the representation of names of languages – Part 2: Alpha-3 code".

[25] IEC 62455: "Internet protocol (IP) and transport stream (TS) based service access".

[26] ISO/IEC 9834-8: "Information technology - Open Systems Interconnection - Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 Object Identifier components".

[27] ISO/IEC 13818-1: "Information technology – General coding of moving pictures and associated audio information: Systems – Transmission of non-telephone signals".

[28] ISO/IEC 14496-1: "Information technology – Coding of audio-visual objects - Part 1: Systems".

[29] ISO/IEC 14496-10:2005: "Information technology – Coding of audio-visual objects - Part 10: Advanced Video Coding", second edition.

[30] ISO/IEC 14496-12:2008: "Information Technology - Coding of audio-visual objects – Part 12: ISO base media file format", third edition.

[31] ISO/IEC 14496-14: "Information technology – Coding of audio-visual objects - Part 14: MP4 file format".

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

[33] ISO/IEC 14496-3:2005: "Information technology – Coding of audio-visual objects - Part 3: Audio", third edition.

[34] Open Mobile Alliance OMA-DRM-DCF: "DRM Content Format".

[35] SMPTE 421M: "VC-1 Compressed Video Bitstream Format and Decoding Process".

[36] SMPTE RP2025: " SMPTE Recommended Practice: VC-1 Bitstream Storage in the ISO Base Media File Format".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in ISO/IEC 14496-12 [30] and ISO/IEC 13818-1 [27], and the following apply:

DVB File Format: The format as contained within this specification.

Descriptive metadata: Metadata which carries descriptions of the content that is useful to a human.

NOTE 1: This is also sometimes referred to as "attractor" or "attractor metadata" as its purpose is to attract viewers to the content.

Key Message: Messages that contain information that allows access to media content.

NOTE 2: The name comes from the normal usage of carrying keys that are used to decrypt media.

Indexing metadata: Metadata that provides mappings between content location(s) and events in the stream.

Parallel index: An index which contains only one type of index event.

Pre-computed hint sample: a sample in a hint track that contains the relevant data.

NOTE 3: Normally a hint sample contains a pointer to the (media) data it contains, rather than containing the data itself.

Reception hint track: a hint track that is created from a received packetized stream, and that may include the data.

NOTE 4: Reception hint tracks can be used both for media and non-media data.

Serial index: An index which contains multiple index types, sorted by the time at which they occur.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

3GPP	
4CC	Four Character Code
AC-3	Dolby AC-3 audio coding system according to TS 102 366
AMR-WB+	Adaptive Multirate Wideband+
AVC	Advanced Video Coding (see also H.264/AVC)

BCG	Broadband Content Guide, as defined in TS 102 539
CA	Conditional Access
CPCM	Content Protection and Copy Management
CT()	Composition Time, as defined in 14496-12
DT()	Decoding Time, as defined in 14496-12
ESG	Electronic Services Guide, as defined in TS 102 471
H.264/AVC	Advanced Video Coding for Generic Audiovisual Services according to ITU-T Recommendation H.264
IPDC	Internet Protocol Datacast
MIME	Multipurpose Internet Mail Extensions
MP4	MPEG-4 File Format, as defined in 14496-14
MPEG	Moving Pictures Expert Group
MPEG2-TS	MPEG-2 Transport Stream (as defined in 13818-1)
MPTS	Multiple Program Transport Stream
NTP	Network Time Protocol
PAT	Program Association Table (as defined in 13818-1)
PCR	Program Clock reference
PMT	Program Map Table (as defined in 13818-1)
PSI	Program Specific Information
RTP	Real-Time Protocol, as defined in RFC3550
RTCP	Real-Time Control Protocol
SAD	Synchronised Auxiliary Data
SDP	Session Description Protocol, as defined in RFCXXX
SI	System Information, as defined in EN 300 468
SPTS	Single Program Transport Stream
SRM	System Renew ability Messages
STC	System Time Clock
TLV	Type, Length, Value
TS	Transport Stream
TV-Anytime	The set of specifications generated by the TV-Anytime Forum
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
UUID	Universally Unique Identifier, as defined within RFC4122 or ITU-T Rec. X.667 | ISO/IEC 9834-8
VC-1	Advanced Video coding according to SMPTE Standard 421M
XML	Extensible Markup Language

4 Use of the DVB File Format

4.1 File identification

4.1.1 File extension

The extension '`.dvb`' shall be used for files where the major brand is one of the DVB File Format brands as specified in clause 4.2. Readers should allow mixed case for the characters in the extension.

4.1.2 MIME types

See Annex A.

4.1.3 Brands

A DVB file shall have the file type box '`ftyp`' at the beginning. If the file type box is not present at the beginning of a file, it shall be assumed that this file was not generated according to this specification.

NOTE: If the file type box is present at the beginning, a reader should parse it first and decide depending on the listed brands whether it is able to parse the file.

4.2 Branding

4.2.1 General

Brands are indicated in the file type box '`ftyp`', defined in clause 4.3 of ISO/IEC 14496-12 [30], which shall be present in conforming files. The fields of the file type box shall be used as follows.

`major_brand` identifies the "best use" of the file and should match the file extension. For files with extension '`.dvb`' and conforming to this specification, the major brand shall be one of the brands specified in this clause.

`minor_version` identifies the minor version of the brand. For files conforming to version x.y.z of this specification, this field takes the value x*256 + y.

`compatible_brands` is a list of brand identifiers (to the end of the box). Files that conform to the constraints for DVB brands specified subsequently in this clause should have the respective brand identifier as one of the compatible brands (unless the brand identifier is the major brand).

4.2.2 Common constraints for all DVB brands

The following constraints shall be followed in all DVB files regardless of which DVB brands are listed.

For all brands defined within the present specification:

- The size of '`moov`' box shall be less than or equal to 1 Mbytes.
- The size of any '`moof`' box shall be less than or equal to 300 kbytes.

NOTE 1: These size limits allow devices to determine internal buffer sizes for processing movie fragments.

NOTE 2: DVB file players have to support movie fragments, but some small DVB files might not contain fragments. Movie fragments, which are an integral part of the ISO base media file format [30], provide a way of adding additional media to a file. Potential examples of this including adding recordings to the end of an existing file, or allowing a player to view a file whilst the file is being recorded, or minimising data-loss in the case of power failure. By using movie fragments the changes to the file are minimised, i.e., the movie box '`moov`' and its contents do not need to be re-written when content is added. For these reasons, movie fragments are an important tool that may be used by DVB files.

NOTE 3: DVB file players should not assume that a fragment starts with a random access point. However, where possible, it is recommended that a writer creates fragments starting with random access points.

- MPEG-4 systems specific information [28] should not be stored in a DVB file. It cannot be assumed that a reader implements the MPEG-4 system architectural elements; a compliant reader shall ignore these elements if one of the DVB file format brands is the major brand.
- When '`dvt1`' or '`dvr1`' is the major brand, the contents of all edit list boxes, if present, are constrained as follows. When present, an edit list box shall contain two entries. The first entry shall be an empty edit (indicated by `media_time` equal to -1), the value of `media_time` of the second entry shall be equal to 0, and value of `segment_duration` of the second entry shall span over the entire track. The value of `media_rate_integer` shall be equal to 1 in both entries.

NOTE 4: An empty edit can be used when the recording of a new RTP stream is started later than the recording of some other RTP streams. This can happen, for example, when sub-titling or narration of a language that was not provided in the previous content item becomes available.

4.2.3 The '`dvt1`' brand

The four-character code '`dvt1`' is used for the DVB Transport Stream brand. Either the major brand or one of compatible brands shall be '`dvt1`' in DVB file recordings from a transmission conforming to TS 101 154 [3].

Files including the '`dvt1`' brand shall be constrained as follows.

- There shall be at least one MPEG2-TS reception hint track or media track present in the file.

- When 'dvt1' is the major brand, all contained media streams in any MPEG2-TS reception hint track present in the file shall conform to TS 101 154 [3].

- When 'dvt1' is the major brand, the following shall apply.

  - If a media track is enabled (track_enabled flag is equal to 1 in header of the Track header box) and the alternate_group for the media track is equal to 0, the media track shall conform to one of the following specifications: ISO/IEC 14496-15 [32], SMPTE RP2025 [36], TS 102 366 [6],or ISO/IEC 14496-14 [31].

  - Otherwise, if the alternate_group for a set of tracks is greater than 0, there shall be an enabled media track in the alternate group that conforms to one of the following specifications: ISO/IEC 14496-15 [32], SMPTE RP2025 [36], TS 102 366 [6],or ISO/IEC 14496-14 [31].

  - Otherwise, a media track shall be marked as disabled (track_enabled flag equal to 0 in header of the Track header box).

NOTE: A track marked as disabled need not be played by a DVB player but can be used for proprietary extensions among other things.

The DVB Transport Stream brand allows the use of any features of the 'iso3' brand as defined in ISO/IEC 14496-12 [30].

When 'dvt1' is the major brand, each sample entry (coding names, protocols) of enabled tracks shall be one of those stated in table 1. When 'dvt1' is the major brand, if a sample entry is not one of those stated in table 1 then the track shall be marked as disabled. When 'dvt1' is a compatible brand and is not the major brand, the formats stated in table 1 are allowed in the sample entries (coding names, protocols) of enabled tracks, but other formats may be present too. The presence of a track with any specific type in table 1 is not guaranteed in any DVB File.

Table 1: Sample entries of enabled tracks in the DVB Transport Stream brand

4CC	Entry name	Clause
rm2t	MPEG-2 Transport Stream reception hint track	5.2.1.2
pm2t	Protected MPEG 2 Transport Stream reception hint track	5.2.1.3
mlix	Movie-level index track	5.3.2.1.3
ixse	DVB serial index track	5.3.3.2
mp4a	MPEG-4 Audio as defined in ISO/IEC 14496-14	5.2.2.2.1
avc1	MPEG/ITU-T AVC/H.264 as defined in ISO/IEC 14496-15	5.2.2.1.1
vc-1	SMPTE VC-1 as defined in SMPTE RP2025	5.2.2.1.2
ac-3	Dolby AC-3 Audio as defined in TS 102 366	5.2.2.2.3

4.2.4 The 'dvr1' brand

The four-character code 'dvr1' is used for the DVB RTP brand. Either the major brand or one of compatible brands shall be 'dvr1' in DVB file recordings from a transmission conforming to ETSI TS 102 005 [4].

Files including the 'dvr1' brand shall be constrained as follows.

- There shall be no references to external data outside the file except, potentially, for content protection specific information that is not transmitted as a parallel stream. Typical examples of this include:

  - rights objects; or

  - long-term keys for content protection.

- There shall be at least one RTP reception hint track or media track present in the file.

- When 'dvr1' is the major brand, the following shall apply.

  - If a media track is enabled (track_enabled flag is equal to 1 in header of the track header box) and the alternate_group for the media track is equal to 0, the media track shall either conform to one of the specifications ISO/IEC 14496-15 [32], SMPTE RP2025 [36], TS 102 366 [6], ISO/IEC 14496-14 [31],

TS 126 244 [15], or 126 245 [16], or be protected with ISMACryp [23] and have an original format among these listed specifications.

- Otherwise, if an RTP reception hint track is enabled and the `alternate_group` for the RTP reception hint track is equal to 0, the contained RTP stream shall conform to TS 102 005 [4], TS 102 472 [8], either protected or not protected by ISMACryp [23].
- Otherwise, if the `alternate_group` for a set of tracks is greater than 0, there shall be an enabled track in the alternate group for which one the following applies.
  - The enabled track is an RTP reception hint track that conforms to TS 102 005 [4], TS 102 472 [8], either protected or not protected by ISMACryp [23].
  - The enabled track is a media track that either conforms to one of the specifications ISO/IEC 14496-15 [32], SMPTE RP2025 [36], TS 102 366 [6], ISO/IEC 14496-14 [31], TS 126 244 [15], or 126 245 [16], or is protected with ISMACryp [23] and has an original format among these listed specifications.
- Otherwise, a media track or an RTP reception hint track shall be marked as disabled (`track_enabled` flag equal to 0 in header of the track header box).

NOTE: A track marked as disabled need not be played by a DVB player but can be used for proprietary extensions among other things.

The brand '`dvr1`' allows the use of any features of the '`iso3`' brand as defined in ISO/IEC 14496-12 [30].

When '`dvr1`' is the major brand, each sample entry (coding names, protocols) of enabled tracks shall be one of those stated in table 2. When '`dvr1`' is the major brand, if a sample entry is not one of those stated in table 2 then the track shall be marked as disabled. When '`dvr1`' is a compatible brand and is not the major brand, the formats stated in table 2 are allowed in the sample entries (coding names, protocols) of enabled tracks, but other formats may be present too. The presence of a track with any specific type in table 2 is not guaranteed in any DVB File.

Table 2: Sample entries of enabled tracks in the DVB RTP brand

4CC	Entry name	Clause
`rrtp`	RTP reception hint track	5.2.1.4
`rtcp`	RTCP reception hint track	5.2.1.6
`prtp`	Protected RTP reception hint track	5.2.1.5
`keym`	Key Message reception track	5.4.2.2
`mp4a`	MPEG-4 Audio as defined in ISO/IEC 14496-3	5.2.2.2.1
`avc1`	MPEG/ITU-T AVC/H.264 as defined in ISO/IEC 14496-10	5.2.2.1.1
`vc-1`	SMPTE VC-1 as defined in SMPTE RP2025	5.2.2.1.2
`sawp`	3GPP AMR-WB+ as defined in TS 126 244	5.2.2.2.2
`tx3g`	3GPP Timed Text as defined in TS 126 244	5.2.2.3.1
`ac-3`	Dolby AC-3 Audio as defined in TS 102 366	5.2.2.2.3
`mlix`	Movie-level index track	5.3.2.1.3
`ixse`	DVB serial index track	5.3.3.2

4.2.5 Use of branding

Every DVB file shall contain the compatible brand '`iso3`' as defined in ISO/IEC 14496-12 [30], which indicates that each DVB file is conformant to the third edition of the ISO base media file format. When media tracks are present, the brand corresponding to the media format shall be present in the list of compatible brands, as listed non-exhaustively in table 3.

A file that has additional compatible brands not mentioned here shall comply with these brands, i.e., a file that has a compatible brand of e.g., '`mp42`' must comply to the MP4 file format specification [31] as it must comply with all other specifications for which a compatible brand is in the file.

Table 3: Compatible brands for various media tracks

Contained in a TS reception hint track	Contained in an RTP reception hint track	As a media track	Media description
dvt1	n/a	n/a	MPEG-2 Video
dvt1	n/a	n/a	MPEG-2 Audio
dvt1	dvr1	n/a	Dolby AC-3 Audio
dvt1	dvr1	avc1	MPEG/ITU-T AVC/H.264
dvt1	dvr1	mp42	MPEG-2/MPEG-4 AAC
dvt1	dvr1	vc-1	SMPTE VC-1
dvt1	dvr1	sawp	3GPP AMR-WB+

4.3 Overview of newly introduced boxes

This document defines the new boxes listed in table 4. Also, a number of existing boxes of the ISO base media file format are extended.

Table 4: New boxes defined in this specification

						Box description	Clause
					dstg	DVB sample to group box	5.3.4.2
					dsgd	DVB sample group description box	5.3.4.3
srmc						System Renewability Message container box	5.4.6.2
	srmb					System Renewability message box	5.4.6.3

4.4 Overview of new track types

This document specifies the new sample entries for track types used in this specification, listed in table 55.

Table 5: Track types defined in this specification

Index	Index description	Clause
rm2t	MPEG-2 Transport Stream reception hint track	5.2.1.2
pm2t	Protected MPEG 2 Transport Stream reception hint track	5.2.1.3
rrtp	RTP reception hint track	5.2.1.4
prtp	Protected RTP reception hint track	5.2.1.5
rtcp	RTCP reception hint track	5.2.1.6
mlix	Movie-level index track	5.3.2.1.3
ixse	DVB serial index track	5.3.3.2
keym	Key Message reception track	5.4.2.2

5 Box definitions and usage

5.1 Descriptive metadata

5.1.1 Introduction

The DVB File Format stores content related, descriptive or annotative metadata in common file format structures, agnostic to the actual metadata schema(s) used. In this clause, the storage mechanism for such metadata is described. A metadata schema for storing a minimal mandatory description is defined. Furthermore, the storage of descriptive metadata adhering to existing, DVB-defined metadata schemas in the DVB File Format is explained. The details of storing proprietary descriptive metadata formats is out of the scope of this clause, though the means of signalling stored metadata as proprietary is described.

This clause describes the storage of metadata in raw, textual, schema-valid XML documents. Henceforward, the term "descriptive metadata item" is used when referring to a single such XML document.

A DVB file may contain multiple descriptive metadata items, where each metadata item is separate textual, schema-valid XML document.

5.1.2 Usage of metadata box

5.1.2.1 Overview

The metadata box '`meta`' (defined in clause 8.11.1 of ISO/IEC 14496-12 [30]) serves as the main entry point for descriptive metadata. Exactly one metadata box shall occur at file level, called the primary metadata box. Additional metadata boxes referencing descriptive metadata defined in this specification shall only be contained in the additional metadata container box '`meco`' as defined in clause 8.11.7 of ISO/IEC 14496-12 [30]. The metadata container box containing metadata box that reference descriptive metadata defined in this specification shall occur only at file level. Each metadata box shall reference only descriptive metadata adhering to the same metadata specification (i.e., having the same handler type as detailed in clause 5.1.2.2).

The primary metadata box shall be used to reference the mandatory basic description as detailed in clause 5.1.4, where the major brand is one defined by this specification.

When the brands defined by this specification are listed only as compatible brands, it is recommend that the primary metadata box is used to reference the mandatory basic description as detailed in clause 5.1.4, unless this conflicts with requirements of the major brand. In this case, the mandatory basic description shall be present and should be referenced in the first metadata box (with a handler type of '`dmbd`') within the file-level metadata container box.

NOTE 1: The complete set of descriptive metadata available in the file can thus be found by following all references from the metadata box at file level as well as all references from all metadata boxes contained in the metadata container box at file level.

NOTE 2: It is possible that a metadata box occurs at levels other than the file level, but that boxes at these levels do not reference "DVB File Format" metadata.

The item location box '`iloc`' in the metadata box is used to locate each descriptive metadata item and to provide a unique number (`item_ID`), which can be used to refer to the descriptive metadata item. Descriptive metadata items adhering to this specification shall not be stored externally.

5.1.2.2 Handler reference box

The handler type specified in the mandatory handler reference box '`hdlr`' of a metadata box defines the format, encoding and interpretation of the descriptive metadata items referenced from that metadata box. Table 6 lists the formats defined and supported by the DVB File Format.

Table 6: Handler types of a metadata box in a DVB File

Specification name	Specification reference	Handler-type	Clause
DVB Mandatory basic description	current	'`dmbd`'	5.1.4
TV-Anytime (TVA)	ETSI TS 102 323 ETSI TS 102 822-3-2	'`dtva`'	5.1.2.2.1
IPDC Electronic Service Guide (ESG)	ETSI TS 102 471	'`ipdc`'	5.1.2.2.2
Broadband Content Guide (BCG)	ETSI TS 102 539	'`dtva`'	5.1.2.2.3
Proprietary descriptive metadata	N/A	'`uri `'	5.1.2.2.4

The handler type for the primary metadata box shall be '`dmbd`'.

5.1.2.2.1 TV-Anytime descriptive metadata

Descriptive metadata items with a handler type of '`dtva`' shall adhere to the TV-Anytime DVB profile, as defined in clause 8 of TS 102 323 [5], and shall be stored as a textual, schema-valid, XML document, as detailed in 5.1.2.6.

5.1.2.2.2 IPDC ESG descriptive metadata

Descriptive metadata items with a handler type of 'ipdc' shall adhere to TS 102 471 [7] and shall be stored as a textual, schema-valid, XML document, as detailed in 5.1.2.6.

5.1.2.2.3 BCG descriptive metadata

The format of descriptive metadata items in the Broadband Content Guide (BCG) metadata as defined in TS 102 539 [10] is identical to that of TV-Anytime metadata and so shall be stored as described in clause 5.1.2.2.1 of the present document.

5.1.2.2.4 Proprietary descriptive metadata

If a metadata box has a handler type of 'uri ', this metadata box shall contain a URI box 'uri ' that identifies the type and the interpretation of the descriptive metadata items referenced from that metadata box. The URI mechanism shall be used to provide proprietary extensions. If desired, a UUID may be encoded into the URI as per ISO/IEC 9834-8 [26].

The URI mechanism shall not be used for descriptive metadata formats with mechanisms defined elsewhere in this specification.

5.1.2.2 Primary item box

Each metadata box shall contain a primary item box 'pitm', defined in clause 8.11.4 of ISO/IEC 14496-12 [30]. The primary item box shall identify the item that contains the main descriptive metadata element. This shall be the default descriptive metadata item.

NOTE: The choice of which item is indicated by the primary item box is at the discretion of the creator of the file. However, as this item is most likely to be the first information presented to the user, it makes sense to select the most informative item.

5.1.2.3 Data information box

The data information box 'dinf' shall not be present in a metadata box with a handler type defined in this specification.

NOTE: This implies that the descriptive metadata for a handler type defined in this specification is contained within the same file.

5.1.2. 4 Item location box

For metadata items with a handler type defined in this specification, this sub-clause shall apply.

Each metadata box shall contain an item location box 'iloc', as defined in clause 8.11.3 of ISO/IEC 14496-12 [30]. For each metadata item present, there shall be an entry in the item location box that maps the descriptive metadata item to an item_ID. Each item_ID shall be unique within the file.

NOTE: There will always be at least one entry, corresponding to the item_ID in the primary item box.

The data-reference-index field shall take the value zero.

5.1.2.5 Location of the metadata box

It is recommended that the metadata box referencing the mandatory basic description (and thus having a handler type of 'dmbd') as well as the mandatory basic description itself is located in the file as soon after the file type box as is possible without contradicting brand requirements.

NOTE 1: In the case of a file whose primary brand is DVB, this means that the metadata box and the media data container box 'mdat' containing the mandatory basic description can occur immediately after the file type box.

NOTE 2: The media data container box containing the mandatory basic description need not be the same media data container box that holds media data.

5.1.2.6 Storage of XML

For metadata items with a handler type defined in this specification, this sub-clause shall apply.

The XML data shall be stored in one or more media data container boxes. The XML box as defined in clause 8.11.2 of ISO/IEC 14496-12 [30] shall not be used.

Each XML document shall be stored as a separate metadata item. A delivery system may provide the XML-based descriptions as a sequence of fragments. Each fragment may be stored either as a separate schema-valid XML document (thereby storing the set of received metadata as multiple metadata items), or all fragments may be combined into a single schema-valid XML document (and so stored as a single metadata item).

Each XML document that is stored shall be stored as a complete schema valid document.

Where an XML document refers to other XML documents, e.g., in the case of Classification Schemes, it is recommended that this information is also provided in the file.

Where an XML document is referred to, it is recommended that the item information box '`iinf`' is present and contains, within the `item_name` field, the reference string, usually a URI, that is used to link to this XML document.

5.1.2.7 Multiple concurrent metadata schemes

If relevant descriptive metadata is to be stored in schemas other than the schema of the mandatory basic description, this metadata shall be stored in metadata boxes in the additional metadata container box '`meco`'. Data formatted according to a specific schema shall be stored in a separate metadata box containing only data formatted to that schema. There shall be a maximum of one metadata box storing data formatted according to the same metadata schema.

If multiple metadata boxes are present, the relation between the descriptive metadata in the different boxes should be described by metadata relation boxes '`mere`', as defined in clause 8.11.8 of ISO/IEC 14496-12 [30].

5.1.3 Handling descriptive metadata updates

The file shall only contain one version of a descriptive metadata item as stored according to this specification (e.g., one version of the description of a TV program). It is possible that between creating a file and the recording completing the descriptive metadata item is updated. In this case the recording device should replace the old description with the new description.

There are numerous ways this can be achieved, which are outside the scope of this specification and implementation dependent.

NOTE: By way of an informative example, one possible mechanism is described. When the file is first created, a media data container box for the descriptive metadata is created that contains an appropriate amount of spare space, typically in the order of a few kilobytes. If an update arrives, this can be placed into the spare space in the media data container box. Then the relevant entries in the relevant item location boxes are updated. By parsing the file, a device is able to identify which locations are free in the media data container box containing the descriptive metadata, and thus the update to the item location box releases the space used by the old description. Should space management be required, the `extent_count` mechanism in the item location box can be used to fragment the description into multiple locations within the media data container box.

5.1.4 Mandatory basic description

5.1.4.1 Overview

The mandatory metadata description is designed to be easily generated from either TV-Anytime [12] or IPDC ESG [7] based metadata descriptions. This basic description provides an interoperability point to assist in the interchange of content between devices. The basic description can take one of two forms. The first format (`FileContentInformation`) contains information about a program, at a minimum the title, but it may be populated with additional information such as is typically provided by DVB SI [1], the IPDC ESG [77], the IPTV BCG [10] or TV-Anytime data carried over a transport stream [5]. The second form (`SelfRecordingInfo`) contains only a description of the recording and will typically be used where the user initiates a recording and may include such details as the time of the recording and a description of the channels recorded.

As with any XML document, additional elements or attributes may be provided utilising additional namespaces. For instance additional fields may be present in the `PurchaseInformation` element. Such additional fields shall not be removed, and may be ignored by receivers.

There is no requirement for a device to implement any of the optional elements set out below. The only mandatory information defined by the present document is the title of the content conveyed in the `FileContentItemInformation.ContentItemInformation.BasicDescription.Title` element which shall appear at least once and shall be processed by receivers.

5.1.4.2 XML schema

Mandatory metadata shall be a schema valid XML document with the schema whose namespace is `urn:dvb:metadata:schema:fileContentItemDescription:2007`. The relevant schema is provided as attachments to this document, and they are available from http://www.dvb.org/metadata/schema. The schema forms part of the normative specification.

5.1.4.3 FileContentItemInformation

5.1.4.3.1 Overview

The `FileContentItemInformation` element shall be used as the root element of the instance document where a recording has been made of a program, or primarily of a program. This element shall reflect the main program that is recorded. Where there are multiple main programs recorded, there may be multiple instance documents present, each with its own `FileContentInformation` for a different program.

NOTE: For example, if a recording starts with the end of program A, contains all of program B, and ends with a small part of program C, the `FileContentItemInformation` should refer to program B.

In situations where the content is self-produced, or primarily self-produced by a user, for example using a digital video camera, the `SelfRecordingInfo` element (see clause 5.1.4.4) may instead be used as the root element of an instance document.

5.1.4.3.2 ContentItemInformation

5.1.4.3.2.1 Overview

This mandatory element shall contain the description of the content item. This element is structurally similar to the `urn:tva:metadata:2007 ProgramInformationType` [12] and the `urn:dvb:ipdc:esg:2005 ContentType` [7]

5.1.4.3.2.2 BasicDescription

This mandatory element shall contain at least one `Title` element, which shall hold the title of the program to which this metadata description refers. The usage of the remaining elements is optional.

The element `RelatedMaterial` is defined using the `urn:tva:metadata:2007 RelatedMaterialType`, the corresponding `urn:dvb:ipdc:esg:2005 RelatedMaterialType` may be additionally present in situations where not all receiver devices can be expected to implement the TVA type.

NOTE: This means that, as with all XML instance documents, additional attributes and elements may be present.

5.1.4.3.2.3 Other elements

The other elements are all optional, and may be provided as available. The usage of these elements is as defined in the TV-Anytime over transport streams [5] or IPTV BCG [10] specifications.

5.1.4.3.3 BroadcastServiceName

If present, this optional element shall be used to hold the name of the service from which the recording was made.

5.1.4.3.4 ScheduleEvent

If present, this optional element shall be used to hold the scheduled event information of the item described in the `ContentItemInformation` element.

Although this element is defined using the `urn:tva:metadata:2007 ScheduleEventType`, the corresponding `urn:dvb:ipdc:esg:2005 ScheduleEventType` may be additionally present in situations where not all receiver devices can be expected to implement the TV-Anytime type.

NOTE: This means that, as with all XML instance documents, additional attributes and elements may be present.

Further details on the elements and attributes of the respective `ScheduleEventType` are provide in the IPDC ESG and TV-Anytime specifications

5.1.4.4 SelfRecordingInfo

5.1.4.4.1 Overview

Where the `FileContentItemInformation` cannot be used, then the `SelfRecordingInfo` shall be used.

5.1.4.4.2 RecordingDescription

This mandatory element shall contain a description of the recording.

NOTE: The details of this textual field are at the discretion of the recording device, but are intended to provide useful information on the recording, such as for instance the time and date it took place and the channel(s) that were recorded.

5.1.4.4.3 RecordingLocation

If present, this optional element shall be used to contain the location at which the recording was initiated.

5.1.5 Other descriptive metadata

5.1.5.1 SI

If an MPEG-2 Transport Stream [27] is encapsulated in a DVB File (as described in clause 5.2.1.2 of the present document), any DVB Service Information according to EN 300 468 [1] may be stored in this encapsulation as received.

Any descriptive information carried by the DVB Service Information that is relevant and that can be stored according to the mandatory basic description metadata schema (defined in clause 5.1.4 of the present document) should be extracted upon reception and stored in this form.

NOTE: If an MPEG-2 TS is encapsulated, the storage of SI information in the mandatory basic description is additional to the storage in the encapsulated format.

5.1.6 Associating descriptive metadata items with content

The time at which a particular descriptive metadata item is valid is signalled using the indexing mechanism defined in clause 5.3.5.5. If no such indexes are present in the file, then the primary metadata item shall be valid for the entire file. It is strongly recommended that if more than one descriptive metadata item is present, then the relevant indexes are provided to indicate where these items are valid.

5.2 Media

5.2.1 Reception hint tracks

5.2.1.1 General

Reception hint tracks may be used when one or more packet streams of data are recorded. Reception hint tracks indicate the order, reception timing, and contents of the received packets among other things.

NOTE 1: Players may reproduce the packet stream that was received based on the reception hint tracks and process the reproduced packet stream as if it was newly received.

Reception hint tracks have an identical structure compared to hint tracks for servers, as specified in the ISO base media file format [30] (in particular, clauses 7.2, 7.3, and 10 of ISO/IEC 14496-12 [30]).

NOTE 2: Consequently, reception hint tracks use the following common set of declarations and structures:

- reception hint tracks may be linked to the elementary stream tracks they carry, by track references of type '`hint`';
- they use a handler-type of '`hint`' in the handler reference box; and
- they use a hint media header box.

The format of the reception hint samples is indicated by the sample description for the reception hint track. Each protocol has its own reception hint sample format.

NOTE 3: Servers using reception hint tracks as hints for sending of the received streams should handle the potential degradations of the received streams, such as transmission delay jitter and packet losses, gracefully and ensure that the constraints of the protocols and contained data formats are obeyed regardless of the potential degradations of the received streams.

NOTE 4: As with server hint tracks, the sample formats of reception hint tracks may enable construction of packets by pulling data out of other tracks by reference. These other tracks may be hint tracks or media tracks. The exact form of these pointers is defined by the sample format for the protocol, but in general they consist of four pieces of information: a track reference index, a sample number, an offset, and a length. Some of these may be implicit for a particular protocol. These "pointers" always point to the actual source of the data, i.e., indirect data referencing is disallowed. If a hint track is built "on top" of another hint track, then the second hint track must have direct references to the media track(s) used by the first where data from those media tracks is placed in the stream.

NOTE 5: Conversion of received streams to media tracks allows existing players compliant with the ISO base media file format to process recorded files as long as the media formats are supported. 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. Players may utilize reception hint tracks for handling of degradations caused by the transmission, i.e., content that may not be correctly decoded is located only within reception hint tracks. The need for having a duplicate of the correct media samples in both a media track and a reception hint track can be avoided by including data from the media track by reference into the reception hint track.

The composition time to sample table '`ctts`' shall not be present in reception hint tracks. The decoding time of reception hint samples indicates reception times as described later in the subsequent clauses.

NOTE 6: Composition times are inferred for received RTP packets as specified in clause 5.2.1.4.4.3.

A reception hint sample having a zero sample size shall be ignored.

NOTE 7: A reception hint sample having a zero sample size can be used when a movie-level index base track (see clause 5.3.2.1.3) would otherwise be shorter than the duration of the movie. An index event in a DVB sample to group box can point to a zero-sized sample in the movie-level index base track.

5.2.1.2 Transport Stream reception hint track

5.2.1.2.1 Introduction (informative)

The MPEG-2 Transport Stream [27] (commonly abbreviated as MPEG2-TS or MPEG-2 TS) is used in many broadcast systems. It is a stream multiplex which can carry one or more programs consisting of audio, video and other media. Sometimes program data cannot be de-multiplexed and elementary streams cannot be created – especially when the transport stream is encrypted and is not allowed to be stored decrypted. Therefore, it is necessary to be able to store the MPEG2-TS as such as a reception hint track.

The MPEG2-TS reception hint track definition supports an approach which is called "pre-computed hints". Pre-computed hints make no use of including data by reference from other tracks, but rather MPEG2-TS packets are stored within the sample. This mechanism also allows having the MPEG2-TS packets stored in a separate file. Furthermore, pre-computed hints facilitate simple recoding operation.

In addition to pre-computed hint samples, it is possible to include media data by reference to media tracks into hint samples. As specified in clause 5.2.1.1, conversion of a received transport stream to media tracks 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. Storing the original transport headers retains valuable information for error concealment and the reconstruction of the original transport stream.

5.2.1.2.2 Sample description format

5.2.1.2.2.1 General

The sample description for an MPEG2-TS reception hint track contains all static metadata that describes the stream, especially the PSI/SI tables. MPEG2-TS reception hint tracks use an entry-format in the sample description of '`rm2t`'.

5.2.1.2.2.2 Syntax

```
class MPEG2TSReceptionSampleEntry extends HintSampleEntry('rm2t') {
    unsigned int(16)    hinttrackversion = 1;
    unsigned int(16)    highestcompatibleversion = 1;
    unsigned int(8)     precedingbyteslen;
    unsigned int(8)     trailingbyteslen;
    unsigned int(1)     precomputed_only_flag;
    unsigned int(7)     reserved;
    box                 additionaldata[];
}

class PATBox() extends Box('tPAT') {
    unsigned int(16)    PID = 0;
    unsigned int(8)     sectiondata[];
}

class PMTBox() extends Box('tPMT') {
    unsigned int(16)    PID;
    unsigned int(8)     sectiondata[];
}

class TSTimingBox() extends Box('tsti') {
    unsigned int(1)     timing_derivation_method;
    unsigned int(2)     reserved;
    unsigned int(13)    PID;
}
```

5.2.1.2.2.3 Semantics

`hinttrackversion` is currently 1; the `highestcompatibleversion` field specifies the oldest version with which this track is backward-compatible.

`precedingbyteslen` indicates the number of bytes that are preceding each MPEG2-TS packet (which may e.g., be a time-code from an external recording device).

`trailingbyteslen` indicates the number of bytes that are at the end of each MPEG2-TS packet (which may e.g., contain checksums or other data that was added by a recording device).

`precomputed_only_flag` indicates that the associated samples are purely pre-computed if set to 1.

additionaldata is a set of boxes. This set can contain boxes that describe one common version of the PSI/SI tables by means of the 'tPAT' box or the 'tPMT' box or other data, e.g., boxes that describe the timing derivation method of the received TS packets.

The 'tPAT' box contains the section data of the PAT and each 'tPMT' box contains the section data of one of the PMTs.

If a box is present that contains data that is also present in the sample data, it shall take precedence over that contained in the sample data. For example, if a 'tPMT' box is present, it shall take precedence over PMT data in the sample data.

NOTE 1: An application that places information in the additionaldata must therefore take care that it is always correct.

In the case of an SPTS, it is strongly recommended that the 'tPMT' box is present in the additionaldata. If the PMT is not present in the sample data, then it shall be present in the additionaldata. If the 'tPMT' box is present, it shall be the PMT for the program contained in the sample data (although the recorded stream may contain other programs and be an MPTS).

NOTE 2: There is no requirement that the 'tPAT' box is present.

In the case of an MPTS, there are no recommendations.

PID is the PID of the MPEG2-TS packets from which the data was extracted. In the case of the 'tPAT' box this value is always 0.

sectiondata is the complete MPEG2-TS table, containing the concatenated sections, of an identical version number.

If the version number of one of the tables stored in the additionaldata increases, another complete MPEG2TSReceptionSampleEntry shall be added to the sample table.

NOTE 3: When a new MPEG2TSReceptionSampleEntry is referred to, a new chunk must be started. The chunks are described by the MPEG2TSReceptionSampleEntry as indexed in the sample to chunk box 'stsc' or track fragment equivalent.

timing_derivation_method is a flag that specifies the method that was used to set the sample time for a given PID. The value for timing_derivation_method is interpreted as follows.

- A value of 0x0 indicates the "reception time" method. The sample timing is derived from the reception time. It is not guaranteed that the STC was recovered for derivation of the reception time.
- A value of 0x1 indicates the "piecewise linearity between PCRs" method. The sample time is derived from a reconstructed STC for this program. Piecewise linearity between adjacent PCRs is assumed and all TS packets in the samples have a constant duration in this range. In the case of an MPTS, "piecewise linearity between PCRs" does not apply and the meaning of the value 0x1 is undefined and its use is not recommended.

timing_derivation_method equal to 0 is inferred if the Transport Stream timing box is not present for a PID.

5.2.1.2.3 Sample format

5.2.1.2.3.1 General

Each sample of an MPEG2-TS reception hint track consists of exactly one raw MPEG2-TS packet (typically of size 188 bytes) including all headers or contains instructions to compose exactly one MPEG2-TS packet by pointing to data of another track, i.e., each sample is defined by the MPEG2TSPacketRepresentation. The MPEG2-TS packet in the sample may be preceded with, or followed by, extra data as detailed in the sample description format. This extra data is marked as precedingbytes and trailingbytes in the sample description.

5.2.1.2.3.2 Syntax

```
aligned(8) abstract class MPEG2TSConstructor (unsigned int(8) type) {
    unsigned int(8) constructor_type = type;
}

aligned(8) class MPEG2TSImmediateConstructor extends MPEG2TSConstructor(1) {
    unsigned int(8) immediatedatalen;
    unsigned int(8) data[immediatedatalen];
}
```

```
aligned(8) class MPEG2TSSampleConstructor extends MPEG2TSConstructor(2) {
    unsigned int(8)     sampledatalen;
    unsigned int(16)    trackrefindex;
    unsigned int(32)    samplenumber;
    unsigned int(32)    sampleoffset;
}

aligned(8) class MPEG2TSPacketRepresentation {
    unsigned int(8) precedingbytes[precedingbyteslen];
    unsigned int(8) sync_byte;
    if (sync_byte == 0x47) {
        unsigned int(8) packet[packetsize - 1];
    } else if (sync_byte == 0x00) {
        unsinged int(8)     headerdatalen;
        unsigned int(4)     reserved;
        unsigned int(4)     num_constructors;
        unsigned int(1)     transport_error_indicator;
        unsigned int(1)     payload_unit_start_indicator;
        unsigned int(1)     transport_priority;
        unsigned int(13)    PID;
        unsigned int(2)     transport_scrambling_control;
        unsigned int(2)     adaptation_field_control;
        unsigned int(4)     continuity_counter;
        if (adaptation_field == '10' || adaptation_field == '11') {
            unsigned int(8) adaptation_field[headerdatalen – 3];
        }
        MPEG2TSConstructor[]
    } else {
        unsigned int(8) reserved[packetsize – 1];
    }
    unsigned int(8) trailingbytes[trailingbyteslen];
}
```

5.2.1.2.3.3 Semantics

`precedingbytes` contains any extra data preceding the packet, typically provided by the recording device. For example, this may include a timestamp.

NOTE 1: The format of `precedingbytes` is not defined by the specification.

`sync_byte`: if this value is 0x47, then the sample is a transport stream packet (a pre-computed reception hint track sample), with the remaining bytes following in the field `packet`. If this value is `0x00`, it indicates that the associated sample points to a track indexed by `trackrefindex` in the track reference box with reference type '`hint`'. All other values are currently reserved.

`trackrefindex` indexes in the track reference box with reference type '`hint`' to indicate with which media track the current sample is associated. The `samplenumber` and `sampleoffset` fields in the `MPEG2TSSampleConstructor` point into this media track.

NOTE 2: It is possible for a mixture of pre-computed and constructed samples to occur in the same track.

NOTE 3: The `trackrefindex` starts from value 1. The value 0 is reserved for future use.

`packet` is the MPEG-2 transport stream packet, excluding the sync byte (0x47).

NOTE 4: The `packet_size` can be inferred from the sample size, as carried by the '`stsz`' box, and the sample description that carries the `precedingbyteslen` and `trailingbyteslen` values, but will be 188 for a normal MPEG-2 transport stream.

The `MPEG2TSConstructor` array is a collection of one or more constructor entries to allow for multiple access units in one transport stream packet. An `MPEG2TSImmediateConstructor` can contain, amongst others, the PES header. An `MPEG2TSSampleConstructor` references data in the associated media track. The sum of the `headerdatalen` and the `datalen` fields of all constructors of an `MPEG2TSPacket` must be equal to the length of the transport stream packet being constructed, minus 1 byte, which is 187.

NOTE 4: If padding of the transport stream packet is required, this shall be done explicitly by using the `MPEG2TSImmediateConstructor` as appropriate.

The remaining fields (`transport_error_indicator`, `payload_unit_start_indicator`, `transport_priority`, `PID`, `transport_scrambling_control`, `adaptation_field_control`, `continuity_counter`, `adaptation_field`) of the sample structure contain a copy of the packet header of the TS packet, as defined in ISO/IEC 13818-1 [27].

trailingbytes contains any extra data following the packet. For example, this may include a checksum.

NOTE 5: The format of trailingbytes is not defined by the specification.

samplenumber indicates the sample within the referred track contained in the packet and sampleoffset indicates the starting byte position of the referred media sample contained in the packet of which sampledatalen bytes are included. sampleoffset starts from value 0.

immediatedatalen indicates the number of bytes at the start of the MPEG2-TS packet payload that are included in the sample (within the field data) rather than data being included into the sample by reference to a media track.

headerdatalen indicates the length of the TS packet header (without the sync byte) in bytes.

5.2.1.2.4 Interaction with ISO-defined boxes

5.2.1.2.4.1 Introduction

By using the MPEG2-TS reception hint tracks, the decoding timing of the samples (i.e., the transport packets) is reception timing, which is typically the same as transport timing.

The following clauses describe the behaviour for a movie box 'moov'; the same shall apply to the equivalent syntax elements in the movie fragment box(es) 'moof', if present.

5.2.1.2.4.2 Sync sample table

A sample that contains an MPEG2-TS packet which contains the first byte of an independently decodable media packet (e.g., MPEG-2 video I-frames) should be marked as a sync sample in the sync sample table box 'stss' if the recorded TS is an SPTS. If the stream is an MPTS, the sync sample table shall be added to the track and be empty.

NOTE: 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 (regarding the file format these are subsequent samples) so that the decoder can decode a complete frame.

5.2.1.2.4.3 Decoding time to sample box

The decoding time to sample box 'stts' contains a run-length encoded representation of the duration of all samples of the media in the track. The decoding time of a sample in an MPEG-2 TS reception hint track is the reception/transmission time of the first bit of that packet, which is recommended to be derived from the PCR timestamps of the TS. If the PCR times are used, piece-wise linearity can be assumed and the decoding time to sample table compacts sensibly. The optional 'tsti' box in the sample description can be used to signal whether reception timing with STC clock recovery was used (in which case the value of timing_derivation_method is 1).

5.2.1.2.4.4 Hint media header box

In the hint media header box 'hmhd', maxPDUsize and avgPDUsize shall be the packet size (fixed at 188) plus the size of extra recurring preceding or trailing data, as specified in the MPEG-2 Transport Stream sample entry.

5.2.1.2.4.5 Sample to chunk box

As implied by the structure of the ISO base file format, for each new MPEG-2 Transport Stream sample entry taken into use, a new chunk must then be started. The chunks are described by the MPEG-2 Transport Stream sample entry as indexed in the sample to chunk box 'stsc'.

5.2.1.3 Protected MPEG 2 Transport Stream reception hint track

5.2.1.3.1 Introduction (informative)

The ISO base file format defines a mechanism for marking media streams as protected, as detailed in clause 8.12 of ISO/IEC 14496-12 [30]. This works by changing the 4CC of the sample entry, and appending boxes containing both details of the protection mechanism and the original 4CC. As reception hint tracks are derived from hint tracks, this same mechanism cannot be directly applied. This clause describes the how reception hint tracks should be marked as protected, and uses an essentially similar mechanism, utilising the boxes defined in the ISO base file format.

Apart from the new sample entry definition, all definitions from the MPEG 2 Transport Stream reception hint track apply here, e.g., the same sample format is used (though it will need to be "unprotected" before it can be used).

5.2.1.3.2 Syntax

```
class ProtectedMPEG2TransportStreamEntry extends MPEG2TSReceptionSampleEntry('pm2t') {
    ProtectionSchemeInfoBox     SchemeInformation;
}
```

5.2.1.3.3 Semantics

The scheme information box '`sinf`' shall contain details of the protection scheme applied. This shall include the original format box '`frma`', which shall contain the 4CC '`rm2t`' (the 4CC of the original MPEG-2 Transport Stream reception hint track sample entry box). If the scheme information box only contains the original format box, then the details of the protection mechanism(s) are contained in the Transport Stream. In the case of CPCM protected content, the scheme information box shall be as defined in clause 5.4.1.2.

NOTE: The scheme information box is defined in clause 8.12.1 of ISO/IEC 14496-12 [30].

5.2.1.4 RTP reception hint track

This clause specifies the reception hint track format for the real-time transport protocol (RTP), as defined in IETF RFC 3550 [22].

5.2.1.4.1 Introduction (informative)

RTP is used for real-time media transport over the Internet Protocol (IP). Each RTP stream carries one media type, and one RTP reception hint track carries one RTP stream. Hence, recording of an audio-visual program results into at least two RTP reception hint tracks.

The design of the RTP reception hint track format follows as much as possible the design of the RTP server hint track format. This design should ensure that RTP packet transmission operates very similarly regardless whether it is based on RTP reception hint tracks or RTP server hint tracks. Furthermore, the number of new data structures in the file format was consequently kept as small as possible.

The format of the RTP reception hint tracks allow storing of the packet payloads in the hint samples, or converting the RTP packet payloads to media samples and including them by reference to the hint samples, or combining both approaches. As noted earlier, conversion of received streams to media tracks 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. Storing the original RTP headers retains valuable information for error concealment and the reconstruction of the original RTP stream. It is noted that the conversion of packet payloads to media samples may happen "off-line" after recording of the streams in pre-computed RTP reception hint tracks has been completed.

5.2.1.4.2 Sample description format

5.2.1.4.2.1 General

The entry-format in the sample description for the RTP reception hint tracks is '`rrtp`'. The syntax of the sample entry is identical compared to the one of RTP hint tracks having the entry-format '`rtp `'.

NOTE: The entry-format in the sample description of the RTP reception hint track is different from the entry-format in the sample description of the RTP server hint track, in order not to accidentally use an RTP reception hint track that contains errors as a valid server hint track.

5.2.1.4.2.2 Syntax

```
class RtpReceptionHintSampleEntry() extends HintSampleEntry('rrtp') {
    unsigned int(16)    hinttrackversion = 1;
    unsigned int(16)    highestcompatibleversion = 1;
    unsigned int(32)    maxpacketsize;
    box                 additionaldata[];
}
```

5.2.1.4.2.3 Semantics

The `hinttrackversion` is currently 1; the `highestcompatibleversion` field specifies the oldest version with which this track is backward-compatible.

The `maxpacketsize` indicates the size of the largest packet that this track will generate.

The `additionaldata` is a set of boxes, which may include the timescale entry '`tims`' and time offset '`tsro`' boxes specified in the ISO base media file format [30]. Moreover, the `additionaldata` may contain a timestamp synchrony box specified subsequently.

NOTE 1: At present, the syntax of the '`tims`' and '`tsro`' boxes is as follows:

```
class timescaleentry() extends Box('tims') {
    unsigned int(32)    timescale;
}

class timeoffset() extends Box('tsro') {
    int(32) offset;
}
```

The timescale entry box shall be present and the value of timescale shall be set to match the clock frequency of the RTP timestamps of the stream captured in the reception hint track. The timescale in the media header box of an RTP reception hint track shall be identical to the timescale in the timescale entry box.

NOTE 2: The clock frequency of RTP timestamps is indicated by the following media-specific attribute in SDP:

```
a=rtpmap:<payload type> <encoding name>/<clock rate>
```

The track SDP information (see clause 5.3.5.7) therefore also contains clock frequency information that is identical with the information in the timescale entry box.

The time offset box may be present. If the time offset box is not present, the value of the field `offset` is inferred to be equal to 0. The value of the field `offset` is used for the derivation of the RTP timestamp, as specified in clause 5.2.1.4.4.3.

RTP timestamps typically do not start from zero, especially if an RTP receiver "tunes" into a stream. The time offset box should therefore be present in RTP reception hint tracks and the value of `offset` in the time offset box should be set equal to the first RTP timestamp of the RTP stream in reception order.

Zero or one timestamp synchrony boxes may be present in the `additionaldata` of an RTP reception hint track sample entry. If a timestamp synchrony box is not present, the value of `timestamp_sync` is inferred to be equal to 0.

```
class timestampsynchrony() extends Box('tssy') {
    unsigned int(6) reserved;
    unsigned int(2) timestamp_sync;
}
```

`timestamp_sync` equal to 0 indicates that the RTP timestamps of the present RTP reception hint track derived from equation 1 may or may not be synchronized with RTP timestamps of other RTP reception hint tracks.

`timestamp_sync` equal to 1 indicates that the RTP timestamps of the present RTP reception hint track derived from equation 1 reflect the received RTP timestamps exactly (without corrected synchronization to any other RTP reception hint track).

`timestamp_sync` equal to 2 indicates that RTP timestamps of the present RTP reception hint track derived from equation 1 are synchronized with RTP timestamps of other RTP reception hint tracks.

NOTE 3: When `timestamp_sync` is equal to 0 or 1, a player should correct the inter-stream synchronization using stored RTCP sender reports. When `timestamp_sync` is equal to 2, the media contained in the RTP reception hint tracks can be played out synchronously according to the reconstructed RTP timestamps without synchronization correction using RTCP Sender Reports.

`timestamp_sync` equal to 3 is reserved.

The value of `timestamp_sync` shall be identical for all RTP reception hint tracks that overlap in their presentation time as indicated by their edit list boxes.

5.2.1.4.3 Sample format

5.2.1.4.3.1 General

The sample format of RTP reception hint tracks is identical to the syntax of the sample format of the RTP hint tracks. Each sample in the reception hint track represents one or more received RTP packets. If each media frame is transmitted contiguously in an RTP stream, i.e., no data unit of a second media frame is transmitted between two data units of any media frame, 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. Otherwise, it is recommended that each RTP reception sample represents a received RTP packet.

Each RTP reception hint sample contains two areas: the instructions to compose the packet, and any extra data needed for composing the packet, such as a copy of the packet payload. Note that the size of the sample is known from the sample size table.

Since the reception time for the packets may vary, this variation can be signalled for each packet as specified subsequently.

5.2.1.4.3.2 Syntax

```
aligned(8) class receivedRTPsample {
    unsigned int(16)    packetcount;
    unsigned int(16)    reserved;
    receivedRTPpacket   packets[packetcount];
    byte                extradata[];
}
```

5.2.1.4.3.3 Semantics

`packetcount` indicates the number of received RTP packets contained in the sample.

`packets` contains the received RTP packets, as detailed in clause 5.2.1.4.4.

`extradata` contains additional data, such as packet payloads, i.e., the concatenated RTP payloads of pre-computed RTP packets.

5.2.1.4.4 Packet entry format

5.2.1.4.4.1 General

Each packet in the packet entry table has the following structure, which is compatible to the corresponding structure of the RTP hint tracks.

5.2.1.4.4.2 Syntax

```
aligned(8) class receivedRTPpacket {
    int(32)             relative_time;
    unsigned int(2)     RTP_version;
    unsigned int(1)     P_bit;
    unsigned int(1)     X_bit;
    unsigned int(4)     CSRC_count;
    unsigned int(1)     M_bit;
    unsigned int(7)     payload_type;
    unsigned int(16)    RTP_seqnum;
    unsigned int(13)    reserved = 0;
    unsigned int(1)     extra_flag;
    unsigned int(1)     reserved = 0;
    unsigned int(1)     reserved = 0;
    unsigned int(16)    entrycount;
    if (extra_flag) {
        unsigned int(32)    extra_information_length;
        box                 extra_data_tlv[];
    }
    dataentry   constructors[entrycount];
}
```

5.2.1.4.4.3 Semantics

When i is the sample number of the present sample, the sum of the sample time DT(i) as specified in clause 8.6.1.2.1 of ISO/IEC 14496-12 [30] and `relative_time` indicates the reception time of the packet. The clock source for the

reception time is undefined and may be, for instance, the wall clock 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.

NOTE 1: Receivers may use a constant value for `sample_delta` in the decoding time to sample box '`stts`' as much as reasonable and smooth out packet scheduling and end-to-end delay variation by setting `relative_time` adaptively in stored reception hint samples. This arrangement of setting the values of `sample_delta` and `relative_time` can facilitate a compact decoding time to sample box.

The values of `RTP_version`, `P_bit`, `X_bit`, `CSRC_count`, `M_bit`, `payload_type`, and `RTP_seqnum` shall be set equal to the V, P, X, CC, M, PT and sequence number fields of the RTP packet captured in the sample.

`extra_flag` equal to 1 indicates that there is extra information before the constructors, in the form of type-length-value sets.

`extra_information_length` indicates the length in bytes of all extra information before the constructors which includes the four bytes of the `extra information_length` field. The subsequent boxes before the constructors, referred to as the TLV boxes, are aligned on 32-bit boundaries. The box size of any TLV box indicates the actual bytes used, not the length required for padding to 32-bit boundaries. The value of `extra_information_length` includes the required padding for 32-bit boundaries.

The following TLV boxes are specified: `rtphdrextTLV`, `rtpoffsetTLV`.

If the `X_bit` is set a single `rtphdrextTLV` box shall be present for storing the received RTP header extension.

```
aligned(8) class rtphdrextTLV extends Box('rtpx') {
    unsigned int(8) data[];
}
```

`data` is the raw RTP header extension which is application-specific.

The syntax of the `rtpoffsetTLV` box, identified with the four-character code '`rtpo`', is specified in the ISO base media file format [30].

NOTE 2: At present, the syntax of the `rtpoffsetTLV` box is as follows:

```
class rtpoffsetTLV() extends Box('rtpo') {
    int(32) offset;
}
```

`offset` indicates a 32-bit signed integer offset to the RTP timestamp of the received RTP packet. Let i be the sample number of the present sample, DT(i) be equal to DT as specified in clause 8.6.1.2.1 of ISO/IEC 14496-12 [30] for sample number i, `tsro.offset` be the value of offset in the '`tsro`' box of the referred reception hint sample entry, and mod be the modulo operation. The value of `offset` shall be such that the following equation is true:

$$RTPtimestamp = (DT_i + tsro.offset + offset) \bmod 2^{32} \quad (1)$$

NOTE 3: When each reception hint sample represents all received RTP packets that have the same RTP timestamp, the value of `sample_delta` in the decoding time to sample box can be set to match the RTP timestamp. In other words, DT(i), as specified above, can be set equal to RTPtimestamp – `tsro.offset` – `offset` (assuming that the resulting value would be greater than or equal to 0).

NOTE 4: RTP timestamps do not necessarily increase as a function of RTP sequence number in all RTP streams, i.e., transmission order and playback order of packets may not be identical. For example, many video coding schemes allow bi-prediction from previous and succeeding pictures in playback order. As samples appear in tracks in their decoding order, i.e., in reception order in case of RTP reception hint tracks, `offset` in the `rtpoffsetTLV` box can be used to warp the RTP timestamp away from the sample time DT(i).

For edits in edit list boxes and association of movie-level index events to RTP reception hint samples, the composition time of a received RTP packet is inferred to be the sum of the sample time DT as specified in clause 8.6.1.2.1 of ISO/IEC 14496-12 [30] and `offset` as specified above.

5.2.1.4.5 Constructor format

The constructor formats for RTP reception hint tracks are specified in clause 9.1.3.2 of the ISO base media file format [30].

5.2.1.5 Protected RTP reception hint track

5.2.1.5.1 Introduction (informative)

The ISO base file format defines a mechanism for marking media streams as protected, as detailed in clause 8.12 of ISO/IEC 14496-12 [30]. This works by changing the 4CC of the sample entry, and appending boxes containing both details of the protection mechanism and the original 4CC. As reception hint tracks are derived from hint tracks, this same mechanism cannot be directly applied. This clause describes the how reception hint tracks should be marked as protected, and uses an essentially similar mechanism, utilising the boxes defined in the ISO base file format.

Apart from the new sample entry definition, all definitions from the RTP reception hint track apply here, e.g., the same sample format is used (though it will need to be "unprotected" before it can be used).

5.2.1.5.2 Syntax

```
class ProtectedRtpReceptionHintSampleEntry extends RtpReceptionHintSampleEntry ('prtp') {
    ProtectionSchemeInfoBox     SchemeInformation;
}
```

5.2.1.5.3 Semantics

The scheme information box '`sinf`' shall contain details of the protection scheme applied. This shall include the original format box '`frma`', which shall contain the 4CC '`rrtp`' (the 4CC of the original RTP reception hint track sample entry box). If the protection scheme information is already present in the SDP information, then this shall not be stored in the scheme information box (i.e., if there is no protection scheme information present in the scheme information box, it will be found in the SDP information).

Where a track contains RTP packets where any of the packets are protected, then the '`prtp`' definition should be used.

5.2.1.6 RTCP reception hint track

This clause specifies the reception hint track format for the real-time control protocol (RTCP), defined in IETF RFC 3550 [22].

5.2.1.6.1 Introduction (informative)

RTCP is used for real-time transport of control information for an RTP session over the Internet Protocol (IP). During streaming, each RTP stream typically has an accompanying RTCP stream that carries control information for the RTP stream. One RTCP reception hint track carries one RTCP stream and is associated to the corresponding RTP reception hint track through a track reference.

The format of the RTCP reception hint tracks allows the storage of RTCP Sender Reports in the hint samples.

The RTCP Sender Reports are of particular interest for stream recording, because they reflect the current status of the server, e.g., the relationship of the media timing (RTP timestamp of audio/video packets) to the server time (absolute time in NTP format). Knowledge of this relationship is also necessary for playback of recorded RTP reception hint tracks to be able to detect and correct clock drift and jitter.

The timestamp synchrony box as specified in clause 5.2.1.4.2.3 makes it possible to correct clock drift and jitter before playing a file and therefore recording of RTCP streams is optional.

NOTE: There is no server hint track equivalent for the RCTP reception hint track because RTCP messages are generated on-the-fly during transmission.

5.2.1.6.2 General

There shall be zero or one RTCP reception hint track for each RTP reception hint track. An RTCP reception hint track shall contain a track reference box including a reference of type '`cdsc`' to the associated RTP reception hint track.

NOTE: 'cdsc' was deliberately chosen over 'hint', as 'cdsc' indicates that the track contains metadata for the associated track, while 'hint' indicates that the associated track contains media data for to be included by reference into the hint track.

When i is the sample number of the present sample, the sample time DT(i) as specified in clause 8.6.1.2.1 of ISO/IEC 14496-12 [30] indicates the reception time of the packet. The clock source for the reception time shall be the same as for the associated RTP reception hint track. The value of timescale in the media header box of an RTCP reception hint track shall be equal to the value of timescale in the media header box of the associated RTP reception hint track.

5.2.1.6.3 Sample description format

5.2.1.6.3.1 General

The entry-format in the sample description for the RTCP reception hint tracks is 'rtcp'.

5.2.1.6.3.2 Syntax

```
class RtcpReceptionHintSampleEntry() extends HintSampleEntry('rtcp') {
    unsigned int(16)    hinttrackversion = 1;
    unsigned int(16)    highestcompatibleversion = 1;
    unsigned int(32)    maxpacketsize;
    box                 additionaldata[];
}
```

5.2.1.6.3.3 Semantics

The hinttrackversion is currently 1; the highestcompatibleversion field specifies the oldest version with which this track is backward-compatible.

The maxpacketsize indicates the size of the largest packet that this track will generate.

The additionaldata is a set of optional additional boxes.

5.2.1.6.4 Sample format

5.2.1.6.4.1 General

Each sample in the reception hint track represents one or more received RTCP packets. Each sample contains two areas: the raw RTCP packets and any extra data needed. Note that the size of the sample is known from the sample size table or equivalent movie fragment structures.

5.2.1.6.4.2 Syntax

```
aligned(8) class receivedRTCPpacket {
    unsigned int(8) data[];
}

aligned(8) class receivedRTCPsample {
    unsigned int(16)    packetcount;
    unsigned int(16)    reserved;
    receivedRTCPpacket  packets[packetcount];
    byte                extradata[];
}
```

5.2.1.6.4.3 Semantics

data contains a raw RTCP packet including the RTCP report header, the 20-byte sender information block and any number of report blocks. Note that the size of each RTCP packet is known by parsing the 16-bit length field of the RTCP header.

packetcount indicates the number of received RTCP packets contained in the sample.

packets contains the received RTCP packets.

extradata is additional data.

5.2.2 Elementary streams

5.2.2.1 Video

5.2.2.1.1 H.264/AVC

H.264/AVC [29] elementary streams stored in accordance with the present specification shall comply with ISO/IEC 14496-15 [32].

H.264/AVC elementary streams stored in accordance with the present specification shall also comply with coding specifications detailed in TS 101 154 [3] or TS 102 005 [4].

5.2.2.1.2 VC-1

VC-1 [35] elementary streams stored in accordance with the present specification shall comply with SMPTE RP2025 [36].

VC-1 elementary streams stored in accordance with the present specification shall also comply with coding specifications detailed in TS 101 154 [3] or TS 102 005 [4].

5.2.2.2 Audio

5.2.2.2.1 MPEG-4 AAC profile, MPEG-4 HE AAC profile and MPEG-4 HE AAC v2 profile

MPEG-4 AAC profile elementary streams, MPEG-4 HE AAC profile elementary streams, and MPEG-4 HE AAC v2 profile elementary streams [33] stored in accordance with the present specification shall comply with ISO/IEC 14496-14 [31].

MPEG-4 AAC profile elementary streams, MPEG-4 HE AAC profile elementary streams, and MPEG-4 HE AAC v2 profile elementary streams stored in accordance with the present specification shall also comply with coding specifications detailed in TS 101 154 [3] or TS 102 005 [4].

5.2.2.2.2 AMR-WB+

AMR-WB+ [17] elementary streams stored in accordance with the present specification shall comply with TS 126 244 [15].

AMR-WB+ elementary streams stored in accordance with the present specification shall also comply with coding specifications detailed in TS 102 005 [4].

5.2.2.2.3 AC-3 and Enhanced AC-3

AC-3 elementary streams and Enhanced AC-3 elementary streams [6] stored in accordance with the present specification shall comply with Annex F of TS 102 366 [6].

AC-3 elementary streams and Enhanced AC-3 elementary streams stored in accordance with the present specification shall also comply with coding specifications detailed in TS 101 154 [3] or TS 102 005 [4].

5.2.2.3 Other

5.2.2.3.1 Timed text

Timed text [16] elementary streams stored in accordance with the present specification shall comply with TS 126 244 [15].

Timed text elementary streams stored in accordance with the present specification shall also comply with coding specifications detailed in TS 102 472 [8].

5.3 Indexing

5.3.1 Introduction (informative)

Indexes are metadata that describe an existing track, which is usually a reception hint track. The metadata contained in indexes would otherwise be available only by parsing a rather large portion of the samples of this track (e.g., the polarity bits of scrambling fields) or is not available at all (e.g., packet corruption indication).

There are two indexing mechanisms specified in the DVB File Format. The serial indexing mechanism enables storage of different types of indexes in an interleaved manner in non-decreasing order of time. The parallel indexing mechanism enables type-wise organisation of indexes into a file. Both types of index structures can be present in the same file. The elementary structures for both mechanisms are identical, which enables conversion of indexes from one mechanism to the other. In many parser implementations, parallel indexes are more efficient for looking up a sample by a specific index, whereas serial indexes are more efficient when indexes for a specific sample are to be looked up. A parser can ignore a parallel indexing structure for a particular index type, if indexes of that index type are not relevant as an output of the parsing process.

Since an MPEG2-TS is a multiplex, the events may apply to only some components within the sample stream that is the MPEG2-TS, and so it is desirable to signal to which component the event applies. The indexing mechanisms therefore support storage of PID and organisation of indexes based on PID. Similarly, it can be beneficial to extract timed descriptive metadata of a particular metadata schema easily, and hence the indexing mechanism supports organisation of timed descriptive metadata indexes based on their metadata schema. Many index types can therefore be instantiated according to one or more key parameters, such as PID. An instance of an index type refers to a particular value of the instance information (`instance_info`) associated with the index type. The parallel indexing mechanism enables grouping of indexes sharing the same index type and instance information into the same data structure. The serial indexing mechanism enables indication of the index types and values of instance information.

It may not always be possible to generate accurate indexes of e.g., an MPEG-2 TS reception hint track, especially (though by no means exclusively) if the content is scrambled. This means that it is desirable to support both approximate and heuristic guesses as to the location of certain key events. These indexes are beneficial as they may often (e.g., in excess of 90% of the time) be accurate, or very close to accurate, and so provide a very useful location to start looking for an index event, saving the cost of searching through a significantly larger amount of data. Since the behaviour may be different between exact and heuristic indexes, it is desirable to indicate the accuracy of the index.

The nature of an MPEG2-TS reception hint track is that a sample corresponds to an MPEG2-TS packet, so any given event may persist over many samples. By contrast, in existing index structures of the ISO base media file format, such as the sync sample box, an index usually applies to a single sample. An index defined in this document relates to the first sample it applies to, but indexes that persist over many samples can be repeated e.g., at the start of a track fragment. Depending on the index type, an index may be implicitly valid for a longer period (e.g., until the next index event of the same class appears). The persistence of indexes can be indicated explicitly by the parallel indexing mechanism.

5.3.2 Common indexing structures

5.3.2.1 General

5.3.2.1.1 Index configuration and index characteristics

The serial indexing mechanism is based on timed metadata tracks and the parallel indexing mechanism is based on an extended version of sample groups. The indexes that can be included in a timed metadata track are described by sample entries included in the sample description box. The indexes represented by a sample grouping are described by an extended version of the sample group description box. Both descriptions use the same structure, called the index configuration structure. For a sample grouping, an index configuration structure contains one index characteristics structure, whereas an index configuration structure included in a sample entry may contain more than one index characteristics structure. An index characteristics structure identifies the index type, provides instance information (`instance_info`) for the index type, and includes static information that remains unchanged for all indexes associated with the index type and instance information. The static information specified in this Technical Specification is formatted according to the index inaccuracy structure, which indicates how accurately the indexes match with referred samples.

5.3.2.1.2 Index event and index element

An index event contains one occurrence of an index. An index event can be parsed independently without parsing other index events. The semantics of an index event are determined by its index type. The syntax of an index event may include instance information (`instance_info`), a grouping index, and a payload (`index_payload`). The grouping index and payload are mutually exclusive, and the grouping index shall only be used for parallel indexing. A grouping index points to an entry in the respective grouping description box, and the group description entry describes the index event. The number of bytes in the payload is provided in the index configuration and the syntax and semantics of the payload are determined by the index type.

An index element is a structure containing data for an index event. All data for one index event is included in one or more consecutive index elements. An index element is the elementary unit within the samples of serial indexing and loop entries used in sample groupings for parallel indexing. The length for an index element remains unchanged within a sample grouping or among samples that refer to the same sample entry (i.e., at least throughout one chunk of samples). Index elements can be accessed randomly due to their constant length.

A sample in an index track may contain any number of index events.

When an index event for particular values of index type and instance information is provided as a serial index, all other index events for the same values of index type and instance information, respectively, shall also be provided as serial indexes.

When an index event for particular values of index type and instance information is provided as a parallel index, all other index events for the same values of index type and instance information, respectively, shall also be provided as parallel indexes.

When index events for particular values of index type and instance information are provided both as a serial index and parallel index, both indexing mechanisms shall include the same index events and therefore they are consistent with each other.

NOTE: Consequently, if index events for particular values of index type and instance information are present both in an index track and a parallel indexing structure, a parser can read either the index track or the parallel indexing structure and need not switch between the indexing mechanisms at any point.

5.3.2.1.3 Movie-level indexes

An index may semantically apply to all reception hint tracks and media tracks present in a file. Such an index is referred to as a movie-level index, whereas indexes applied to a single track only are referred to as track-level indexes. All movie-level indexes shall be associated with a single track, referred to as the movie-level index base track. If any reception hint track is present, then one of them shall be assigned as the movie-level index base track. Otherwise (when no reception hint track is present), one media track shall be assigned as the movie-level index base track. An empy track reference type box with a `reference_type` of value '`mlib`' shall be included in the movie-level index base track, if any movie-level index is present in a file. Zero or one movie-level index tracks shall be present in a file. A movie-level index track is identified by the metadata sample entry type '`mlix`', whereas a track-level index track is identified by the metadata sample entry type '`ixse`' as specified in clause 5.3.3.2.2.

NOTE 1: When a file writer selects the movie-level index base track among RTP reception hint tracks or media tracks, the movie-level index base track should be such that no alternative track of the same media type is present among the group of tracks to increase the probability that the movie-level indexes are associated with a track that is played.

NOTE 2: A file writer should add zero-sized samples at the beginning or at the end of the movie-level index base track, if the base track otherwise would start later or end earlier (respectively) than any other track in the file. The sample duration of the zero-sized samples should be the default duration of the movie-level index base track, or, if no default sample duration exists, no greater than 1 second. Extending the movie-level index base track with zero-sized samples helps in adding parallel movie-level index events that occur close to the start or the end of the movie.

When parallel indexing is used for movie-level indexes, the DVB sample to group box should be present only in the movie box and should not be present in any track fragment box unless the referred `sample_number` would exceed the maximum value of a 32-bit unsigned integer. Parallel indexing at movie-level should only be used for the bookmark index.

5.3.2.1.4 Index validity timing

Index events are linked to either media tracks or reception hint tracks through timing and sample numbering. The index inaccuracy structure (clause 5.3.2.4) indicates how accurately the provided time and sample number relationship presents the actual occurrence of the index event.

A track-level index event becomes active at the time of the linked media sample or reception hint sample. Depending on the semantics of the index type, the track-level index may be used for parsing (and hence become active at the decoding time of the linked media sample or reception hint sample) or for playing (and hence become active at the composition time of the linked media sample or reception hint sample). If both timing accuracy and sample accuracy are equal to 'accurate', then the decoding time of a track-level index sample shall be equal to the decoding time of the respective sample of the referred reception hint track or media track indicated by the sample number in the indexing structures.

A movie-level index event becomes active at its composition time. If serial indexing is used for movie-level indexes, the composition time of an index event is derived from the decoding time to sample box of the movie-level index track or the respective syntax elements in the track fragment box. If both timing accuracy and sample accuracy are equal to 'accurate', the composition time of the media sample or reception hint sample associated with the indicated sample number shall be the closest composition time in the media or reception hint track that is greater than or equal to the composition time of the index sample. If parallel indexing is used for movie-level indexes, the composition time of an index event is identical to that of the associated media sample or reception hint sample (through the sample number mechanism of parallel indexing).

The indicated timing accuracy for the movie-level index track shall not be equal to 'unspecified'.

If the sample accuracy is unspecified for the movie-level index track, the value of `relative_sample_number` in the index elements included in the movie-level index track shall be equal to 0, linking through sample numbers is unspecified and timing shall be used to link index events to samples in media tracks or reception hint tracks. If the sample accuracy is not equal to 'unspecified', the track-level indexes are linked to media samples or reception hint samples through the sample number indicated in the indexing structures as specified subsequently.

NOTE 1: If the sample accuracy is unspecified for the movie-level index track, no zero-sized samples are required in the movie-level index base track for linking through the sample number mechanism.

NOTE 2: A parser may link an index event to a sample in a reception hint track or a media track through timing or sample numbering. A parser may choose between linking through timing or sample numbering based on the indicated timing accuracy or sample accuracy.

NOTE 3: The decoding and composition times of a movie-level index sample need not coincide with those of the associated media sample or reception hint sample. In other words, the timing accuracy of a movie-level index track is not governed by the sampling times of the associated media tracks and reception hint tracks.

5.3.2.2 Index configuration

5.3.2.2.1 Syntax

```
aligned(8) class IndexConfiguration {
    IndexCharacteristics    index_characteristics;
    unsigned int(13)        reserved = 0;
    unsigned int(1)         ic_instance_info_in_event_flag;
    unsigned int(1)         ic_grouping_index_in_event_flag;
    unsigned int(1)         ic_multi_element_event_flag;
    unsigned int(16)        ic_index_element_length;
    unsigned int(16)        entry_count;
    for (i = 1; i <= entry_count; i++) {
        IndexCharacteristics    more_index_characteristics;
    }
    Box additional_box[];
}
```

NOTE 1: `index_characteristics` is located at the start of the structure (rather than within the loop) to make the structure of the DVB sample group description box similar to the sample group description box of the ISO base media file format, i.e., starting with the index or grouping type.

NOTE 2: Syntax elements prefixed with '`ic_`' are used in other index structures.

5.3.2.2.2 Semantics

index_characteristics indicates the characteristics of indexes associated with this index configuration structure.

ic_instance_info_in_event_flag equal to 0 specifies that the instance_info syntax element is not present in the index events associated with this index configuration structure. ic_instance_info_in_event_flag equal to 1 specifies that the instance_info syntax element is present in the index events associated with this index configuration structure. ic_instance_info_in_event_flag shall be equal to 1 if more than one value of instance_info for any index_type is listed in index_characteristics and more_index_characteristics or if any_instance_info_flag is equal to 1 in index_characteristics or more_index_characteristics.

ic_grouping_index_in_event_flag equal to 0 specifies that the grouping_index syntax element is not present in the index events associated with this index configuration structure. ic_grouping_index_in_event_flag equal to 1 specifies that the grouping_index syntax element is present in the index events associated with this index configuration structure. ic_grouping_index_in_event_flag shall be equal to 0 when this index configuration structure is included in a sample entry.

NOTE 1: ic_grouping_index_in_event_flag can be equal to 1 only for the parallel indexing structure.

ic_multi_element_event_flag equal to 0 specifies that the cont_number and last_cont_number syntax elements are not present in the index elements associated with this index configuration structure. ic_multi_element_event_flag equal to 1 specifies that the cont_number and last_cont_number syntax elements are present in the index elements associated with this index configuration structure.

NOTE 2: ic_multi_element_event_flag shall be equal to 1 when any index event spans to more than one index element.

ic_index_element_length specifies the number of bytes in each index element associated with this index configuration structure.

entry_count indicates the number of subsequent index characteristics structures (called more_index_characteristics). entry_count shall be equal to 0 when the index configuration structure is included in a DVB sample group description box.

more_index_characteristics indicates the characteristics of indexes associated with this index configuration structure.

The values of syntax elements in index_characteristics and more_index_characteristics are constrained as follows. If any_instance_info_flag is equal to 1 for a value of index_type, there shall be no other index characteristics structure with the same value of index_type present in the same sample entry or DVB sample group description box. The value of instance_info shall differ in each index characteristics structure having the same value of index_type within the same sample entry or DVB sample group description box.

additional_box is unspecified and shall be ignored.

5.3.2.3 Index characteristics

5.3.2.3.1 Syntax

```
aligned(8) class IndexCharacteristics {
    unsigned int(32)    index_type;
    if (index_type == 'uuid') {
        unsigned int(8) uuid[16];
    }
    unsigned int(32)    instance_info;
    unsigned int(7)     reserved = 0;
    unsigned int(1)     any_instance_info_flag;
    unsigned int(8)     version;
    unsigned int(16)    event_payload_length;
    unsigned int(8)     static_length;
    unsigned int(8)     static_index_info[static_length];
}
```

5.3.2.3.2 Semantics

index_type indicates an index type that may be present in serial or parallel indexing structures associated with the index configuration containing this index characteristics structure. A value of index_type that is not present in a sample entry shall not be used in any sample referring to the sample entry.

uuid is an optional field shall only be present when the index_type is 'uuid'. This shall hold the UUID value, and shall be treated as a qualification of the index_type.

instance_info indicates the instance information for the index type. The semantics of instance_info are determined by index_type as specified in 5.3.5. If any_instance_info_flag is equal to 0, the value of instance_info for this index_type in the serial or parallel indexing structures associated with the index configuration containing this index characteristics structure shall be equal to one of the instance_info values listed for this index_type in the index configuration. If any_instance_info_flag is equal to 1, any value of instance_info for this index_type may be present in the serial or parallel indexing structures associated with the index configuration containing this index characteristics structure.

version shall be equal to 0 according to this Technical Specification. version greater than 0 may imply that event_payload_length can be greater than the value of event_payload_length specified for index_type in this Technical Specification. Those syntax elements occurring after the last specified syntax element for the index_type in an index event, if any, shall be ignored.

NOTE 1: version enables to extend the syntax and semantics for an index event in a backward-compatible manner. In other words, syntax elements can be appended in a new version, while the beginning of the structure remains unchanged compared to the previous version.

event_payload_length indicates the number of bytes in the index event payload associated with index_type and instance_info. event_payload_length equal to 0xFFFF indicates a varying payload length, in which case the payload length is included in each index event structure.

static_length indicates the number of bytes in the static_index_info syntax element.

static_index_info contains information that is valid for all index elements associated with these index characteristics. For index_type values specified in this Technical Specification, the syntax and semantics of static_index_info conforms to IndexInaccuracy.

In a future release of the DVB File Format, static_length may be greater than the size of the respective IndexInaccuracy structure. Those syntax elements occurring after the last specified syntax element in static_index_info, if any, shall be ignored.

NOTE 2: static_length enables to extend the syntax and semantics for static_index_info in a backward-compatible manner. In other words, syntax elements can be appended to the end of the static_index_info in a new version of the DVB File Format, while the beginning of the structure remains the same as in the previous version of the DVB File Format.

5.3.2.4 Index inaccuracy

5.3.2.4.1 General

The index inaccuracy structure is stored in the static information of the index characteristics structure. There are two types of inaccuracy indicated, sample and time inaccuracy. Index events are linked to either media tracks or reception hint tracks through sample numbers or time, as specified in clause 5.3.2.1.4. The index inaccuracy structure indicates how accurately the provided sample number and time relationships present the actual occurrence of the index event.

5.3.2.4.2 Syntax

```
aligned(8) class IndexInaccuracy {
    unsigned int(4) sample_accuracy;
    unsigned int(4) time_accuracy;
    if (sample_accuracy >= 8)
        unsigned int(24)    max_sample_inaccuracy;
    if (time_accuracy >= 8)
        unsigned int(32)    max_timing_inaccuracy;
}
```

5.3.2.4.3 Semantics

sample_accuracy indicates whether or not indexes may be inaccurate in terms of sample number of the referred track and which type of inaccuracy information is provided. The semantics of sample_inaccuracy are specified in table 7.

time_accuracy indicates whether or not indexes may be inaccurate in terms of decoding time of the referred track and which type of inaccuracy information is provided. The semantics of time_inaccuracy are specified in table 7.

`max_sample_inaccuracy` specifies the maximum inaccuracy in terms of sample numbers.

`max_timing_inaccuracy` specifies the maximum inaccuracy in terms of decoding time of the referred track.

Table 7: Interpretation of Accuracy Values

Value	Meaning
0	Accurate
1	Unspecified
2	Heuristic
3	Reserved (no maximum provided)
4-7	Application-specific (no maximum provided)
8	Maximum inaccuracy specified
9	Reserved (maximum inaccuracy provided)
10-15	Application-specific (maximum inaccuracy provided)

5.3.2.5 Index event

5.3.2.5.1 Syntax

```
abstract class IndexEvent(index_type, event_payload_length) {
    if (ic_instance_info_in_event_flag)
        unsigned int(32)    instance_info;
    if (ic_grouping_index_in_event_flag)
        unsigned int(32)    grouping_index;
    else {
        if (event_payload_length == 0xFFFF) {
            unsigned int(16)    var_payload_length;
            unsigned int(8)     index_payload[var_payload_length];
        }
        else
            unsigned int(8)     index_payload[event_payload_length];
    }
    unsigned int(8) pad[];
}
```

5.3.2.5.2 Semantics

`instance_info` indicates the instantiation information for the index type. The semantics of `instance_info` is determined by `index_type` as specified in 5.3.5.

`grouping_index` specifies the group description index within the DVB sample group description box for which the values of `index_type` and `instance_info` are the same as the values of `index_type` and `instance_info`, respectively, for this index event. The respective entry in the DVB sample group description box describes the index event. The syntax of the group description entry is specific for the value of `index_type`. `grouping_index` shall not be present when the index event is included in a sample.

`var_payload_length` contains the number of bytes in the payload.

`index_payload` contains the value of the index. The semantics of `index_payload` is determined by `index_type` as specified in clause 5.3.5.

`pad` is used adjust the size of an index event in order to make the concatenation of `index_event_data` as specified in clause 5.3.2.6 conforming to the `IndexEvent` structure. The values of data bytes contained in pad are not constrained.

NOTE: The `ic_instance_info_in_event_flag` and `ic_grouping_index_in_event_flag` flags are global and are defined in clause 5.3.2.2.

5.3.2.6 Index element payload

5.3.2.6.1 Syntax

```
class IndexElementPayload(index_type, element_payload_length) {
    if (ic_multi_element_event_flag) {
```

```
        unsigned int(8)     cont_number;
        unsigned int(8)     last_cont_number;
        this_payload_length = element_payload_length - 2;
    }
    else
        this_payload_length = element_payload_length;
    unsigned int(8) index_event_data[this_payload_length];
}
```

5.3.2.6.2 Semantics

One or more index elements contain data for one index event. An index element consists of a number of bytes given by `ic_index_element_length`. An index element has three parts, an element header, an element payload, and padding to obtain a constant index element size. The element header differs in parallel and serial indexing. The element payload is specified by the index element payload structure. Padding (`pad`) is specified in clause 5.3.2.5.

NOTE 1: The index element header in serial indexing is included in the sample format and consists of the following syntax elements:

```
unsigned int(32) index_type;
unsigned int(32) relative_sample_number;
```

The index element header in parallel indexing is included in the entry loop of the DVB sample to group box and consists of the following syntax elements:

```
unsigned int(32) sample_number;
unsigned int(32) sample_count;
```

`cont_number` shall be 0 for the first index element of an index event. `cont_number` shall be incremented by 1 per each index element of the index event in the appearance order of index elements. `cont_number` equal to 255 is reserved.

`last_cont_number` shall be equal to the greatest value of `cont_number` for this index event. `last_cont_number` equal to 255 is reserved.

NOTE 2: The `ic_multi_element_event_flag` flag and the `ic_index_element_length` field are global and are defined in clause 5.3.2.2.

`index_event_data` contains the data for an index event. An index event (`index_event`) is a byte array obtained by concatenating the byte arrays of `index_event_data` for all values of `cont_number` from 0 to `last_cont_number`, inclusive, in ascending order of `cont_number`. The index event (`index_event`) is formatted according to `IndexEvent(index_type, event_payload_length)` specified in clause 5.3.2.5, wherein `event_payload_length` is obtained from the index characteristics structure having both following properties:

- the index characteristics structure has the same value of `index_type` as given in the `index_type` input parameter for the index element payload structure; and
- the index characteristics structure has one of the following properties:
  - the value of `any_instance_info_flag` is equal to 1;
  - the value of `any_instance_info_flag` is equal to 0, `ic_instance_info_in_event_flag` is equal to 1, and the index characteristics structure has the same value of `instance_info` as the `instance_info` included in the index event structure; or
  - the value of `any_instance_info_flag` is equal to 0, `ic_instance_info_in_event_flag` is equal to 0, and there is only one index characteristics structure for the `index_type` value included in the associated index configuration structure.

5.3.3 Serial indexing

The serial indexing mechanism specified in this clause enables storage of different types of indexes in an interleaved manner.

5.3.3.1 General

Zero or more timed metadata tracks storing indexes may be present in a file. A timed metadata track storing indexes is referred to as an index track. One sample in an index track contains one or more index events, each represented by one or more index elements. Index tracks are linked to the tracks they describe by track references of type '`cdsc`'. The start

and end of track fragments of an index track should be aligned with track fragments of the linked media or reception hint track.

NOTE: If track fragments of an index track are not aligned with those of the associated media track or reception hint track, DVB parsers may be unable to process the file appropriately.

5.3.3.2 Sample description format

5.3.3.2.1 General

A DVB track-level index sample entry or a DVB movie-level index sample entry indicates the types of indexes that may be present in samples associated with this sample entry included in a track-level index track or a movie-level index track, respectively. If there are many timed metadata tracks for a reception hint track, `index_type` and `instance_info` values included in the index configuration structure can be used to locate the track containing the desired indexes.

5.3.3.2.2 Syntax

```
class DVBTrackLevelIndexSampleEntry() extends MetadataSampleEntry('ixse') {
    int(32)             sample_number_offset;
    IndexConfiguration  index_configuration;
}

class DVBMovieLevelIndexSampleEntry() extends MetadataSampleEntry('mlix') {
    int(32)             sample_number_offset;
    IndexConfiguration  index_configuration;
}
```

5.3.3.2.3 Semantics

`sample_number_offset` specifies an offset to be added to `relative_sample_number` in the associated timed metadata samples to obtain the sample number in the referred track.

`index_configuration` specifies the characteristics of the indexes that may be present in samples associated with this sample entry and the syntax of the samples associated with this sample entry.

5.3.3.3 Sample format

5.3.3.3.1 Syntax

```
aligned(8) class IndexSample {
    do {
        unsigned int(32)    index_type;
        unsigned int(32)    relative_sample_number;
        IndexElementPayload(index_type, ic_index_element_length–8) element_payload;
    } while (more_data_in_sample());
}
```

5.3.3.3.2 Semantics

A sample contains one or more index elements and the index elements of a sample may carry data for one or more index events. The number of index elements in a sample is determined by the sample size. The function `more_data_in_sample()` returns 1 if there is unread data left in the sample and it returns 0 otherwise. The sample size shall be an integer multiple of `ic_index_element_length` of the associated sample entry.

`index_type` indicates the type of the index included in this index element.

`relative_sample_number` indicates the sample in the referred track which the index element is associated with. The associated sample number in the referred track is equal to `relative_sample_number` + `sample_number_offset`, where `sample_number_offset` is obtained from the associated sample entry. If the sample accuracy is unspecified for the movie-level index track, the value of `relative_sample_number` + `sample_number_offset` shall be equal to 0, linking through sample numbers is unspecified and timing shall be used to link index events to samples in media tracks or reception hint tracks.

NOTE: `relative_sample_number` + `sample_number_offset` indicates a sample number relative to the start of the associated track. `relative_sample_number` + `sample_number_offset` is not relative to any track fragment present for the track.

`element_payload` is a structure formatted according to `IndexElementPayload` specified in clause 5.3.2.6.

5.3.3.4 Mutual relations of serial indexing structures (informative)

Figure 1 illustrates the relations of indexing structures for serial indexing. The figure illustrates the structure of a chunk of index samples. The first illustrated index sample contains two index elements, whose length is indicated in the associated sample entry containing an index configuration structure. The number of index elements for an index sample can be derived from the sample size box. An index element consists of three parts: an index element header, an index element payload, and padding to obtain a constant index element size. An index element header contains the four-character code for index type (`index_type` in the figure) and the relative sample number (`offset` in the figure), which is used to derive the associated absolute sample number for which the index event contained in the index element pertains. An index element payload may contain `cont_number` (`cont_no` in the figure) and `last_cont_number` (`last_cont` in the figure) fields if their presence is indicated in the index configuration structure. These fields are used to concatenate index events that are carried in multiple index elements. An index element also contains index event data. If an index element is the last one for an index event and would be smaller than `ic_index_element_length`, padding (`Pad bytes` in the figure) is used to make the size of the index element equal to `ic_index_element_length`. If an index event is carried in multiple index elements, the index event is a concatenation of the successive chunks of index event data and usually contains the `var_payload_length` (`payload_lnth` in the figure) to indicate the number of bytes for the index event payload (also referred to as index payload or the `index_payload` field). Otherwise, an index event consists of a single piece of index event data. When index events are used for serial indexing, they usually contain the instance information field (`instance_info` in the figure). The values of `index_type` and `instance_info` of an index element refer to the index characteristics structure (contained in the index configuration structure) with identical values of `index_type` and `instance_info`, respectively. The syntax and semantics for different index event payloads are specified in clause 5.3.5.

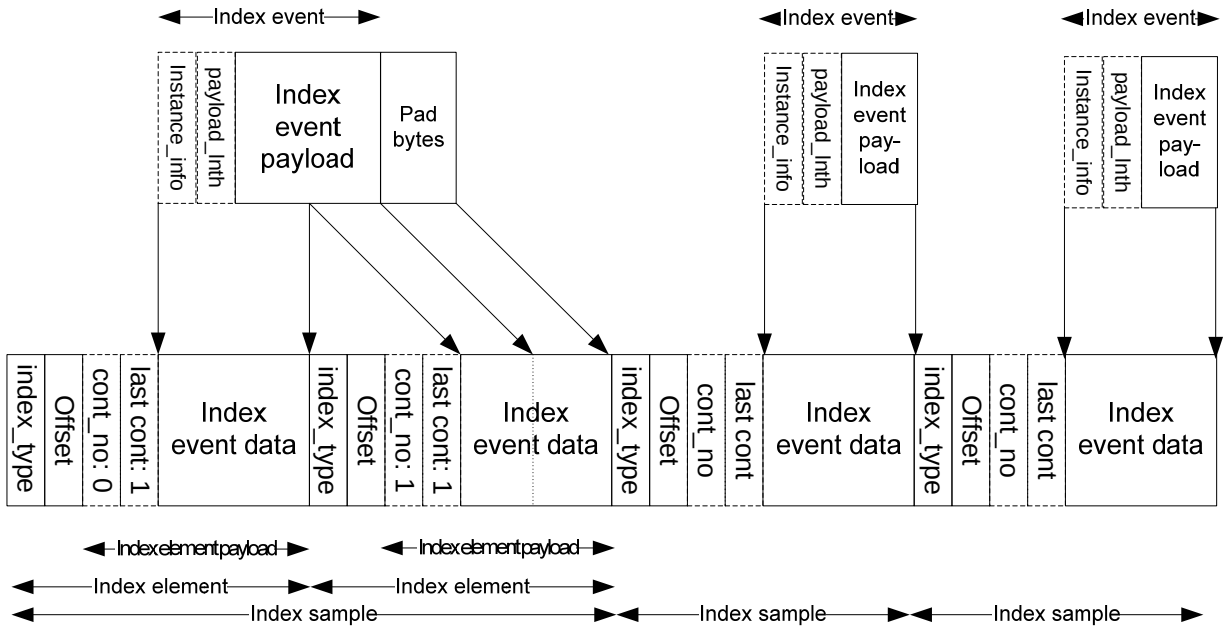

Figure 1: Classes and fields used in serial indexing

5.3.4 Parallel indexing

The parallel indexing mechanism specified in this clause enables type-wise organisation of indexes into a file.

5.3.4.1 General

The parallel indexing mechanism of the DVB File Format uses an extended version of the sample grouping mechanism of the ISO base media file format – the DVB sample to group box and the DVB sample group description box of the

DVB File Format correspond to the sample to group box and the sample group description box of the ISO base media file format, respectively.

The DVB sample to group box is used to couple samples with index events of certain index type (`index_type`) and index instance information (`instance_info`). There can be at most one DVB sample to group box having particular values of index type and instance information present for a track fragment. Each unique value pair of `index_type` and `instance_info` has its own DVB sample to group box within a track fragment. An index event included in a DVB sample to group box may contain a group description index or an index payload. A group description index points to the respective group description entry in the DVB sample group description box, and the group description entry describes the index event. The index payload contains a set of syntax elements describing the index event.

DVB sample to group boxes having particular values of index type and instance information are coupled with the DVB sample group description box having the same values of index type and instance information, respectively. There can be at most one DVB sample group description box having particular values of index type and instance information present in a file. The DVB sample group description box contains the index configuration structure for the indicated pair of index type and instance information. It may also contain sample group description entries describing the index events.

5.3.4.2 DVB sample to group box

5.3.4.2.1 General

The DVB sample to group box is used to couple samples with index events of certain index type (`index_type`) and index instantiation information (`instance_info`). The box contains a loop, where each loop entry includes an index payload for one or more consecutive samples.

DVB sample to group boxes are contained in sample table boxes '`stbl`' or track fragment boxes '`traf`'. If there is more than one DVB sample to group box with the same value of `index_type` present in the same container box, the value of `instance_info` shall differ in each of the DVB sample to group boxes.

5.3.4.2.2 Syntax

```
aligned(8) class DVBSampleToGroupBox
    extends FullBox('dstg', version = 0, flags)
{
    unsigned int(32)    index_type;
    unsigned int(32)    instance_info;
    unsigned int(32)    entry_count;
    for (i=1; i <= entry_count; i++) {
        unsigned int(32)    sample_number;
        unsigned int(32)    sample_count;
        IndexElementPayload(index_type, ic_index_element_length – 8) element_payload;
    }
}
```

5.3.4.2.3 Semantics

`index_type` indicates the semantics of `instance_info`, `static_grouping_info`, and `element_payload`. The values of `index_type` and the inferred semantics for indexing structures are specified in clause 5.3.5.

`instance_info` indicates the instance information for the grouping type.

`sample_number` indicates the first sample to which `element_payload` applies. The value of `sample_number` is relative to the first sample associated with the track fragment box containing this box or to the first sample associated with the track box containing this box, whichever is applicable. `sample_number` is equal to 1 for the mentioned first sample. Values of `sample_number` shall appear in a non-decreasing order in the DVB sample to group box.

`sample_count` indicates the number of consecutive samples, to which `element_payload` applies. `sample_count` equal to 0 indicates that the persistence of the index event is determined by its semantics.

`element_payload` is a structure formatted according to `IndexElementPayload` specified in clause 5.3.2.6. Consecutive index elements composing one index event shall have the same value of `sample_number` and the same value of `sample_count`.

Values of `sample_number` for different index events shall increase in appearance order.

When index values for a particular `index_type` are mutually exclusive, a sample shall not be included more than once in an index event of a DVB sample to group box, i.e., the sum of previous values of `sample_number` and `sample_count`

is required to be less than the present value of sample_number. When index values are not mutually exclusive, a sample can be included multiple times in a DVB sample to group box, i.e., the sum of previous values of sample_number and sample_count is not required to be less than the present value of sample_number. For example, the start and end times of consecutive content items included in an electronic service guide can overlap to some extent. For the values of index_type specified in this Technical Specification, the index values are mutually exclusive unless stated otherwise.

It is not required that each sample in a track is mapped into an index event in a DVB sample to group box.

In the following, conditions applying to sample_number + sample_count – 1 only apply where sample_count is not equal to 0.

If sample_number or (sample_number + sample_count – 1) exceeds the number of samples in the track fragment or the track in the movie box containing the DVB sample to group box, the associated samples reside in one or more subsequent track fragments and those subsequent track fragments shall not contain a DVB sample to group box with the same values of index_type and instance_info.

Unless it is explicitly recommended in the semantics of particular index types, sample_number or (sample_number + sample_count – 1) should not exceed the number of samples in the track fragment or the track in the movie box containing the DVB sample to group box.

NOTE: For the case of some index types, if sample_number or (sample_number + sample_count – 1) exceeds the number of samples in the track fragment or the track in the movie box containing the DVB sample to group box, DVB parsers may be unable to process the file appropriately.

5.3.4.3 DVB sample group description box

5.3.4.3.1 General

There shall be one DVB sample group description box having particular values of index type and instance information for the DVB sample to group boxes having the same values of index type and instance information, respectively. The DVB sample group description box contains the index configuration for the indicated values of index type and instance information.

When it is indicated in the index configuration that group description indexes are used in the respective index events, the DVB sample group description box also contains a loop of sample group description entries. Index events contain an index of the loop entry included in the DVB sample group description box. Each loop entry is formatted as a sample group entry, the syntax and semantics of which depend on the index type as specified in clause 5.3.5.

DVB sample group description boxes are contained in the sample table box. If there is more than one DVB sample group description box with the same value of index_type present in the same container box, the value of instance_info shall differ in each of these DVB sample group description boxes.

5.3.4.3.2 Syntax

```
aligned(8) class DVBSampleGroupDescriptionBox(unsigned int(32) handler_type)
    extends FullBox('dsgd', version, flags)
{
    IndexConfiguration  index_configuration;
    unsigned int(32)    default_length;
    unsigned int(32)    entry_count;
    for (i = 1 ; i <= entry_count ; i++){
        if (default_length==0)
            unsigned int(32)    description_length;
        switch (handler_type){
            case 'vide': // for video tracks
                VisualSampleGroupEntry();
                break;
            case 'soun': // for audio tracks
                AudioSampleGroupEntry();
                break;
            case 'hint': // for hint tracks
                HintSampleGroupEntry();
                break;
            case 'meta': // for timed metadata tracks
                MetadataSampleGroupEntry();
                break;
        }
    }
}
```

5.3.4.3.3 Semantics

`index_configuration` specifies the characteristics of the indexes that may be present in DVB sample to group boxes associated with this DVB sample group description box and the syntax of the loop entries in the DVB sample to group boxes associated with this DVB sample group description box.

`default_length` indicates the length of every group entry (if the length is constant), or zero (0) if it is variable.

`entry_count` indicates the number of sample group entries in the following loop.

`description_length` indicates the length of an individual group entry, in the case it varies from entry to entry and `default_length` is therefore 0.

5.3.4.4 Mutual relations of parallel indexing structures (informative)

Figure 2 illustrates the relations of indexing structures for parallel indexing. A file may contain many DVB sample to group boxes '`dstg`'. A DVB sample to group box starts with `index_type` and `instance_information` fields, the values of which are used to refer to a DVB sample group description box '`dsgd`' with identical values of `index_type` and `instance_information`, respectively. The DVB sample group description box contains the index configuration structure for the indexes contained in the respective DVB sample to group box. DVB sample to group boxes also contain an entry count indicating the number of index elements contained in the box. An index element has a fixed size in terms of bytes, which is indicated in the index configuration structure. An index element consists of three parts: an index element header, an index element payload, and padding to obtain a constant index element size. An index element header contains the sample number (`smpl_num` in the figure) and sample count (`smpl_cnt` in the figure) for which the index event contained in the index element pertains. An index element payload may contain `cont_number` (`cont_no` in the figure) and `last_cont_number` (`last_cont` in the figure) fields if their presence is indicated in the index configuration structure. These fields are used to concatenate index events that are carried in multiple index elements. An index element also contains index event data. If an index element is the last one for an index event and would be smaller than `ic_index_element_length`, padding (`pad` in the figure) is used to make the size of the index element equal to `ic_index_element_length`. If an index event is carried in multiple index elements, the index event is a concatenation of the successive chunks of index event data and usually contains the `var_payload_length` (`payload_lnth` in the figure) to indicate the number of bytes for the index event payload (also referred to as index payload or the `index_payload` field). Otherwise, an index event consists of a single piece of index event data. The syntax and semantics for different index event payloads are specified in clause 5.3.5.

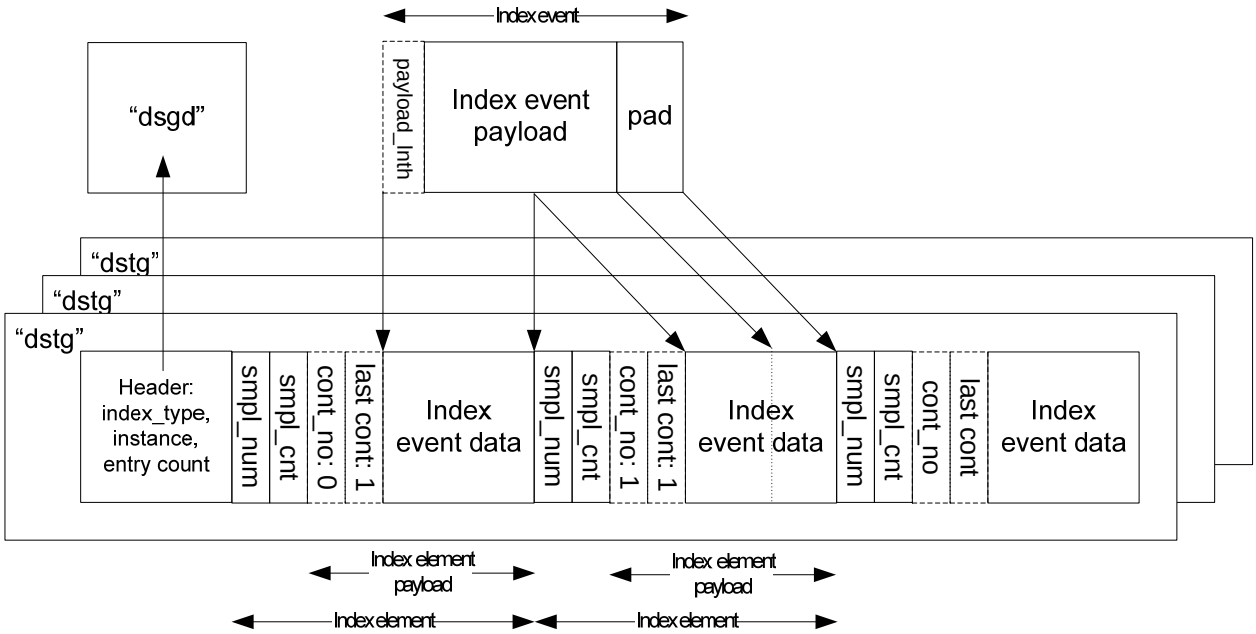

Figure 2: Classes and fields used in parallel indexing

5.3.5 Index data structures

5.3.5.1 Use of length fields

In the definitions below, no `event_payload_length` fields are given (as carried in the index characteristics structure or in the event itself as the `var_payload_length` field where the length may vary between instances of the event type). The `event_payload_length` field value shall be greater than or equal to the length determined from the structures below. The value of `event_payload_length` may be greater than the value implied from the structures below; in this case the additional data may be ignored, but shall be preserved in any file modifications performed.

NOTE: The index event payload includes data formatted as specified below, which can be followed by unspecified data. Thus, the size of the structure (including padding) does not indicate the "version" of the structure, nor vice-versa. Interpretation of the unspecified data is not in the scope of this specification. Such unspecified data may be used for implementation-specific extensions under constrained circumstances.

5.3.5.2 Use of instance information

5.3.5.2.1 Introduction

The 32-bit value of instance information (`instance_info`) is structured according to one the syntax structures specified subsequently. The syntax and semantics of `instance_info` depend on index type used. Two structures for `instance_info` are specified: MPEG2-TS instance information and instance information for timed descriptive metadata indexes. The respective syntax structures are `TS_instance_info` and `timed_metadata_instance_info`.

MPEG2-TS instance information indicates which PID an event is associated with, i.e., the PID of the packet containing the event. Where all events are associated with a single PID, then the `TS_instance_info` may occur only once in the `instance_info` field of the index characteristics structure which describes the event. Where the PID can vary, i.e., the event may occur on a range of PIDs, then the `TS_instance_info` shall be present in each `index_event`. In this case the field `ic_instance_info_in_event` shall be set to one.

Timed descriptive metadata instance information indicates the handler type of the descriptive metadata that is indexed.

Where no instance information is given for indexes associated with MPEG2-TS, the sample indicated by the index event must be used to determine the PID. However, it is recommended that instance information is provided.

5.3.5.2.2 MPEG2-TS instance information

5.3.5.2.2.1 Syntax

```
class TS_instance_info {
    unsigned int(3)     reserved = 0;
    unsigned int(13)    pid;
    unsigned int(16)    reserved = 0;
};
```

5.3.5.2.2.2 Semantics

`pid` is a 13-bit unsigned integer which contains the PID of the packet associated with the index event.

5.3.5.2.3 Timed descriptive metadata instance information

5.3.5.2.3.1 Syntax

```
aligned(8) class timed_metadata_instance_info {
    unsigned int(32)    handler_type;
}
```

5.3.5.2.3.2 Semantics

`handler_type` contains the 4CC of the format of descriptive metadata that is indexed by index events using this instance information, as specified in clause 5.1.2.2. This is used by the player to locate the appropriate metadata box to locate the correct descriptive metadata item.

5.3.5.3 Index types

Table 8 identifies the index types that are used, and the clause specifying the related index structures.

Those events whose `index_type` value starts with '`ts`' shall only occur where the indexed track is of type '`rm2t`' or '`pm2t`'. Timed descriptive metadata, SDP index, and bookmark are movie-level indexes. An index of type '`uuid`' can be either movie-level or track-level index. All other indexes only occur at the track level.

Table 8: Index types

`index_type`	**Event**	**`instance_info` structure**	**Clause**
`tspl`	Polarity Change	`TS_instance_info`	5.3.5.4.1
`tsca`	CA	`TS_instance_info`	5.3.5.4.2
`tspc`	PCR	`TS_instance_info`	5.3.5.4.3
`tstu`	Table Update	`TS_instance_info`	5.3.5.4.4
`sadi`	Synchronised Auxiliary Data	`TS_instance_info`	5.3.5.4.5
`idvi`	Video Index Event	`TS_instance_info` for '`rm2t`' and '`pm2t`' tracks. Unspecified for tracks of any other type.	5.3.5.4.6
`tsau`	AU_Information Event	`TS_instance_info`	5.3.5.4.7
`tdmi`	Timed descriptive metadata	`timed_metadata_instance_info`	5.3.5.5
`bkmr`	Bookmark	Unspecified	5.3.5.6
`sdpi`	SDP	Unspecified	5.3.5.7
`uuid`	Proprietary index event	Unspecified	5.3.5.8
`erro`	Error	`TS_instance_info` for '`rm2t`' and '`pm2t`' tracks. Unspecified for tracks of any other type.	5.3.5.9
`null`	Null index event	Unspecified	5.3.5.10

5.3.5.4 Transport stream indexes

5.3.5.4.1 Polarity change event

5.3.5.4.1.1 General

The polarity change event index contains the new polarity value which applies to the first MPEG-2 TS packet with this new polarity. A polarity change index shall only be deemed to occur when the polarity of a stream of packets on a given PID changes, and not when it changes between packets of different PIDs.

5.3.5.4.1.2 4CC

This event shall be signalled by the 4CC '`tspl`'.

5.3.5.4.1.3 Index configuration

The `version` field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.1.4 Syntax

```
aligned(8) class TSPolarityChangeEventPayload {
    unsigned int(8) polarity;
}
```

5.3.5.4.1.5 Semantics

`polarity` is a 8-bit unsigned integer which shall contain one of the values listed in table 9. This shall indicate the value of polarity that applies to the sample (TS packet) indexed by this event, which shall be the first sample (TS packet) of this new value.

Table 9: Interpretation of polarity values

Value	Meaning
0	Clear
1	Odd polarity
2	Even polarity
3	Reserved
4	Unspecified polarity
5 to 127	Reserved for future specifications
128 to 255	Reserved for Private use

5.3.5.4.2 CA event

5.3.5.4.2.1 General

The CA event index contains a new CA message and indicates the MPEG-2 TS packet where this new CA message starts. A new CA event index shall only be deemed to occur when a previously unseen CA message arrives, and not when previously seen CA messages are interleaved.

5.3.5.4.2.2 4CC

This event shall be signalled by the 4CC '`tsca`'.

5.3.5.4.2.3 Index configuration

The `version` field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.2.4 Syntax

```
aligned(8) class TSCAEventPayload {
    unsigned int(8)     polarity;
    unsigned int(16)    CA_event_data_length;
    unsigned int(8)     CA_event_data[CA_event_data_length];
}
```

5.3.5.4.2.5 Semantics

`polarity` indicates the polarity to which this event applies, according to table 9. If no polarity applies, the value 4 (unspecified polarity) shall be used.

`CA_event_data_length` specifies the number of bytes in the following CA section or message data. A `CA_event_data_length` of zero indicates that the `CA_event_data` is not present in the `TSCAEventPayload` class, and must be retrieved from the indexed sample.

`CA_event_data` contains the actual CA section or message data, if present.

5.3.5.4.3 PCR event

5.3.5.4.3.1 General

The PCR event index contains the PCR value carried in the adaptation field of the indexed MPEG-2 TS packet. Where PCR events are supported, a PCR event need not be generated for every single PCR in the stream. However, if the `discontinuity_indicator` flag in the MPEG-2 TS packet for which the PCR event is listed, it is strongly recommended that the PCR event is generated.

5.3.5.4.3.2 4CC

This event shall be signalled by the 4CC 'tspc'.

5.3.5.4.3.3 Index configuration

The version field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.3.4 Syntax

```
aligned(8) class TSPCREventPayload {
    unsigned int(1)     PCR_discontinuity_flag;
    unsigned int(1)     PCR_value_valid;
    unsigned int(4)     reserved = 0;
    unsigned int(42)    PCR_value;
}
```

5.3.5.4.3.5 Semantics

PCR_discontinuity_flag shall have the same value as the discontinuity_indicator flag in the TS packet containing the indexed PCR.

PCR_value_valid shall be set to 1 if the value carried in the field PCR_value is valid. It is recommended that this value is set to 1, and the value is carried.

PCR_value shall have the PCR value as reconstructed from the fields program_clock_reference_base and program_clock_reference_extension, as defined in ISO/IEC 13818-1 [27].

5.3.5.4.4 Table update event

5.3.5.4.4.1 General

The table update event index occurs when an updated version of a table occurs, and shall index the MPEG-2 TS packet that contains the first byte of the new table. Optionally, various fields, including the entire section data, may be piggybacked in the event.

NOTE: It is not a requirement that all tables are indexed; it is at the discretion of the implementation which tables are indexed.

5.3.5.4.4.2 4CC

This event shall be signalled by the 4CC 'tstu'.

5.3.5.4.4.3 Index configuration

The version field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.4.4 Syntax

```
aligned(8) class TSTableUpdateEventPayload {
    unsigned int(1)     table_id_valid;
    unsigned int(1)     table_id_extension_valid;
    unsigned int(1)     section_no_valid;
    unsigned int(1)     table_data_present;
    unsigned int(4)     reserved = 0;
    unsigned int(8)     table_id;
    unsigned int(16)    table_id_extension;
    unsigned int(8)     section_no;
    unsigned int(16)    table_data_length;
    unsigned int(8)     table_data[table_data_length];
}
```

5.3.5.4.4.5 Semantics

table_id_valid indicates that the table_id field below is valid if this field has the value 1.

table_id_extension_valid indicates that the table_id_extension field below is valid if this field has the value 1.

`section_no_valid` indicates that the `section_no` field below is valid if this field has the value 1.

`table_data_present` is a flag which indicates if the data for the table update (i.e., the table data itself) is present in the subsequent bytes).

`table_id` shall take the value of the `table_id` field of the indexed table update, if valid.

`table_id_extension` shall take the value of the `table_id_extension` (which may take other names, e.g., `program_number`, for some tables) field of the indexed table update, if valid.

`section_no` shall take the value of the `section_number` field of the indexed table update, if valid.

`table_data_length` contains the length of the table data. If the table data is not present, this shall take the value 0.

`table_data` contains the table data, if present.

5.3.5.4.5 Synchronised Auxiliary Data event

5.3.5.4.5.1 General

The Synchronised Auxiliary Data (SAD) event provides a means to index the presence of SAD descriptors in the recorded stream, and store them into the index. SAD is defined in TS 102 823 [13].

5.3.5.4.5.2 4CC

This event shall be signalled by the 4CC '`sadi`'.

5.3.5.4.5.3 Index configuration

The `version` field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.5.4 Syntax

```
aligned(8) class TSSADEventPayload {
    unsigned int(8)     SAD_descriptor[];
}
```

5.3.5.4.5.5 Semantics

This index is used to indicate the location of SAD descriptors in the target track, and provide a copy of the descriptor.

`SAD_descriptor` contains the descriptor which starts in the indicated packet, in accordance with clause 5 of TS 102 823 [13].

NOTE: The length of the `SAD_descriptor` is carried in the 2nd byte of the descriptor, as defined by TS 102 823 [13], and so is not replicated in the `TSSADEventPayload`.

5.3.5.4.6 Video index event

5.3.5.4.6.1 General

The video index event is used to signal the location of the start of video pictures, and potentially indicate the properties of these pictures. It is not required that every single video picture is indexed. For instance, some applications may wish to only index a subset of frames that meet certain properties, though this may require indexing other frames to indicate the end of a frame that is of interest.

5.3.5.4.6.2 4CC

This event shall be signalled by the 4CC '`idvi`'.

5.3.5.4.6.3 Index configuration

The `version` field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.6.4 Syntax

```
aligned(8) class DVBVideoIndexEventPayload {
    unsigned int(2) reserved = 0;
    unsigned int(2) sample_depends_on;
    unsigned int(2) sample_is_depended_on;
    unsigned int(2) sequence_depends_on;
    unsigned int(8) dependency_level;
}
```

5.3.5.4.6.5 Semantics

`sequence_depends_on` signals the dependencies of the sequence that this sample is part of, as defined in table 10.

Table 10: Meaning of `sequence_depends_on` values

Value	Meaning
0	The dependency of the sequence starting from this sample is unknown
1	This or a subsequent sample depends on earlier samples
2	This or any subsequent sample does not depend on earlier samples of the same elementary stream
3	Reserved

`sample_depends_on` signals the dependencies of the sample, as defined in table 11.

Table 11: Meaning of `sample_depends_on` values

Value	Meaning
0	The dependency of this sample is unknown
1	This sample does depend on others (not an I picture)
2	This sample does not depend on others (I picture)
3	Reserved

`sample_is_dependend_on` indicates if this sample is used by other samples, as defined in table 12.

Table 12: Meaning of `sample_is_dependend_on` values

Value	Meaning
0	The dependency of other samples on this sample is unknown
1	Other samples depend on this one (not disposable)
2	No other sample depends on this one (disposable)
3	Reserved

`dependency_level` indicates the dependency level of the sample, as defined in table 13. No sample marked with a dependency level of M shall depend on a sample of dependency level greater than M.

Table 13: Meaning of `dependency_level` values

Value	Meaning
0x00	The dependency level of this sample is unknown
0x01 to 0xFE	This sample does not depend on any sample having a greater value of dependency level
0xFF	Reserved

NOTE: "sample" means the video access unit contained in, or starting in, the associated reception hint sample.

5.3.5.4.7 AU_information event

5.3.5.4.7.1 General

The AU_information event signals the presence of an AU_information structure in the adaptation layer of a transport packet, as defined in Annex D of TS 101 154 [3]. It also allows for the information contained to be optionally carried in the event payload.

5.3.5.4.7.2 4CC

This event shall be signalled by the 4CC 'tsau'.

5.3.5.4.7.3 Index configuration

The version field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.4.7.4 Syntax

```
aligned(8) class TSAUInformationEventPayload {
    unsigned int(8) AU_Information_descriptor_length;
    unsigned int(8) AU_information[AU_Information_descriptor_length];
}
```

5.3.5.4.7.5 Semantics

AU_Information_descriptor_length signals the length of the AU_Information field that follows. This field may take the value 0 which indicates that no descriptor is present, and the relevant TS packet must be consulted to determine the values.

NOTE: This field is not the same value as the descriptor length field in the AU_information structure. It is actually 2 more.

AU_Information carries the AU_information descriptor present in the adaptation layer of the packet, if present.

5.3.5.5 Timed descriptive metadata

5.3.5.5.2 Timed descriptive metadata index

5.3.5.5.2.1 General

This clause specifies indexes related to descriptive metadata fragment specified in clause 5.1. A timed descriptive metadata index event provides a definitive set of metadata items of the specified handler type that are valid at the declared time.

5.3.5.5.2.2 4CC

This event shall be signalled by the 4CC 'tdmi'.

5.3.5.5.2.3 Index configuration

The version field of the index characteristics class associated with events of this format shall have the value 1.

The value of ic_grouping_index_in_event_flag shall be equal to 0.

5.3.5.5.2.4 Syntax

```
aligned (8) class TimedDescriptiveMetadataIndexEventPayload ('tdmi') {
    unsigned int(8) item_count;
    for(i = 0; i < item_count; i++) {
        unsigned int(16)    item_ID;
    }
}
```

5.3.5.5.2.5 Semantics

`item_count` contains the number of items in the following loop.

`item_ID` contains the identifier of the item declared by the item location box in the appropriate metadata box. This is used to locate the actual descriptive metadata fragments.

5.3.5.6 Bookmark index

5.3.5.6.1 Introduction (informative)

Two types of bookmarks can be present in a DVB file: pre-authored and user-defined. In the case a DVB file is transferred to a DVB player device (instead of streaming the content of the file to DVB receivers), pre-authored bookmarks can be inserted into the file by a content provider. Pre-authored bookmarks can be used to provide access to various content scenes and they are often available in several languages. User-defined bookmarks are inserted to a file by a writer or player usually as a response to a user action. A specific user-defined bookmark is reserved for indicating the latest playback position.

DVB players can list the provided bookmark descriptions in a user interface and offer users for the possibility to access the bookmarked positions.

NOTE 1: A player is not required to implement any bookmark functionality. Whilst a file creator must insert a '`dsdg`' and '`dstg`' boxes as detailed below, it is not required to support any further funtionality.

NOTE 2: If a file is exchanged, appropriate privacy measures may be required, such as removal or reseting of certain bookmark fields.

5.3.5.6.2 General

The bookmark index shall not be used with serial indexing.

A conforming file shall contain a DVB sample group description box with index type '`bkmr`' in the movie-level index base track. The DVB sample group description box with index type '`bkmr`' shall contain at least one sample group description entry. The first entry in the DVB sample group description box with index type '`bkmr`' is reserved for the latest playback position of the file. A conforming file shall not contain any DVB sample group description box with index type '`bkmr`' in any track that is not the movie-level index base track.

NOTE 1: Unlike other file format structures, bookmarks are usually inserted by file players. The ability of file players to modify files is often quite limited. Therefore, when a DVB file is originally generated, it is recommended to create a free space box immediately following the DVB sample group description box with index type '`bkmr`'. It is up to the file writers to decide the size of the free space box, but it is recommended to be 1 kbytes, which is large enough for several bookmarks but does not increase the file size dramatically.

A conforming file shall contain one DVB sample to group box with index type '`bkmr`' in the sample table box of the movie-level index base track. A conforming file shall not contain any DVB sample to group boxes with index type '`bkmr`' in the track fragment box of the movie-level index base track unless the total number of samples starting from the track or track fragment containing the previous DVB sample to group box with index type '`bkmr`' is greater than the maximum value of a 32-bit unsigned integer. A conforming file shall not contain any DVB sample to group box with index type '`bkmr`' in any track that is not the movie-level index base track.

NOTE 2: The number of DVB sample to group boxes of index type '`bkmr`' is constrained to make the operation of a file player easier. Due to this constraint, all the bookmarked positions are usually stored in one table. Consequently, players can modify a single table when adding bookmarks rather than dealing with several tables associated with different movie fragments. Moreover, the bookmarked positions (in terms of sample numbers) are stored in a single table making random access to a bookmark straightforward.

NOTE 3: When a DVB file is originally generated, it is recommended to create a free space box immediately following the DVB sample to group box with index type '`bkmr`'. It is up to the file writers to decide the size of the free space box. However, box size is recommended to be an integer multiple of `ic_index_element_length` where the integer multiplier is selected to yield a box size close to 768 bytes, which is large enough for several bookmarks but does not increase the file size dramatically.

If a DVB player chooses to implement the support for storing the last playback position, the player should store the latest playback position of the file by creating an entry in the DVB sample to group box of index type 'bkmr', where sample_number indicates the latest playback position and grouping_index is equal to 1. The total number of entries where grouping_index is equal to 1 shall be exactly zero or one when considering all DVB sample to group boxes of index type 'bkmr'. Consequently, when writing a new entry with grouping_index equal to 1, DVB players are obliged to remove the previous entry with grouping_index equal to 1.

NOTE 4: A bookmark is associated with a description included in the DVB sample group description box and bookmarked sample numbers in DVB sample to group boxes.

NOTE 5: A DVB player can insert a bookmark entry with the following procedures. Let the new bookmark entry concern sample number equal to sam_num in the following procedures.

When the bookmark description is not present in the file yet, a player can insert it with the following procedure into the DVB sample group description box with index type 'bkmr'.

1) If the DVB sample group description box is not followed by a free space box whose size covers at least the size of the bookmark description entry, the file should be rearranged to make room for a new description entry in the DVB sample group description box by means not specified in this procedure.

2) A new description entry is added as the last item in the description entry loop in the DVB sample group description box.

3) The box header of the DVB sample group description box is modified to contain the added group description entry.

4) If there is remaining free space following the DVB sample group description box, the header for the free space box is newly created into the file.

The bookmarked sample is then marked into the DVB sample to group box having index type 'bkmr' with the following procedure.

1) Traverse the DVB sample to group box until the previous entry, if any, has sample_number less than or equal to sam_num and the current entry, if any, has sample_number greater than sam_num.

2) If the DVB sample to group box is not followed by a free space box of at least ic_index_element_length bytes of size (usually 12 bytes), the file should be rearranged to make room for a new entry in the DVB sample to group box by means not specified in this procedure.

3) If there is no entry having sample_number greater than sam_num, the new entry is appended to the end of the table of the DVB sample to group box.

4) If there are one or more values of sample_number present that are greater than sam_num, these entries are moved forwards in the file by the amount of ic_index_element_length. The new bookmark entry is created at the position of the current entry.

5) The box header of the DVB sample to group box is modified to contain the additional ic_index_element_length bytes of data.

6) If there is remaining free space following the DVB sample to group box, the header for the free space box is newly created into the file.

NOTE 6: Starting the playback from a bookmarked position can be implemented similarly to any other random access play operation, e.g., based on other indexes.

5.3.5.6.3 Syntax for sample group description entry

```
class DVBBookmarkEntry {
    unsigned int(7)     reserved = 0;
    unsigned int(1)     read_only_flag;
    unsigned int(8)     reference_count;
    unsigned int(16)    num_languages;
    for (i = 0; i <= num_languages; i++) {
        const bit(1)    pad = 0;
        unsigned int(5) language[3]; // ISO-639-2/T language code
        string          bookmark;
    }
```

}

5.3.5.6.4 Semantics for sample group description entry

`read_only_flag` indicates recommendations how the associated sample group description entry and index events mapped to it should be processed when a file is modified. `read_only_flag` equal to 0 specifies that any modifications are allowed concerning this sample group description entry and index events mapped to it. `read_only_flag` equal to 1 specifies that the sample group description entry should not be modified, the index events that are mapped to this sample group description entry should not be removed, and no index event mapped to this sample group description entry should be added.

NOTE 1: When a DVB file is provided for playback using a file delivery mechanism, the content provider may include pre-defined bookmarks that cannot be changed by end-users by setting the value of `read_only_flag` equal to 1. The value of `read_only_flag` for user-defined bookmarks should be equal to 0.

`reference_count` shall be equal to the number of index events mapped to this sample group description entry as long as it is less than or equal to 255. `reference_count` equal to 255 indicates that the number of index event mapped to this sample group description entry is greater than or equal to 255.

NOTE 2: When displaying the set of bookmarks in the file, a DVB file player can display only the bookmarks with `reference_count` greater than 0 for the end-user.

`num_languages` specifies the number of languages according to which a bookmark is given. All the bookmark strings (`bookmark`) in the following loop in the same sample group description entry specify translations of the same bookmark in different languages. The value of `num_languages` shall not be equal to 0 except for the first sample group description entry.

`language` declares the language code for the following bookmark. See ISO 639-2 [24] for the set of three character codes. Each character is packed as the difference between its ASCII value and `0x60`. The code is confined to being three lower-case letters, so these values are strictly positive.

`bookmark` is a null-terminated string in either UTF-8 or UTF-16 characters, giving the bookmark text. If UTF-16 is used, the string shall start with the BYTE ORDER MARK (`0xFEFF`).

NOTE 3: Regardless of the provided text in the `bookmark` field for the first entry, the first entry always indicates the latest playback position of the file. It is allowed to omit `language` and `bookmark` for the first entry by setting `num_languages` for the first entry equal to 0.

5.3.5.6.5 Index configuration

The value of `ic_instance_info_in_event_flag` shall be equal to 0. The value of `ic_grouping_index_in_event_flag` shall be equal to 1. The value of `ic_multi_element_event_flag` shall be equal to 0.

5.3.5.7 SDP index

5.3.5.7.1 General

Both initial SDP descriptions and SDP updates, which can be provided by mechanisms such as those, for example, specified in TS 102 591 [11], are stored with SDP indexes specified in this clause. There shall be one movie-level index track containing SDP indexes in any DVB file that contains RTP reception hint tracks. SDP indexes shall not be provided with the parallel indexing mechanism.

Each track fragment shall contain the SDP descriptions that are valid for some samples of the referred track fragment.

NOTE 1: Consequently, SDP descriptions have to be repeated when changing track fragments if they are still applied in the latter track fragment. DVB players can find valid SDP descriptions from the indexing for the track fragment from which the playback is started. In other words, DVB players need not scan any other track fragments than the one where playback is started to locate applied SDP descriptions.

NOTE 2: SDP indexes should not be redundantly repeated except for the those updates that are inherited from the previous track fragment.

When timing_accuracy for SDP indexes is equal to 0 ('accurate'), the DT for the sample including the SDP index shall match the NTP timestamp value in the 't=' line of SDP.

5.3.5.7.2 4CC

The index_type shall be equal to 'sdpi'.

5.3.5.7.3 Index configuration

The version field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.7.4 Syntax

```
aligned(8) class SDPPayload {
    unsigned int(16)    media_specific_part_count;
    for (i = 1; i <= media_specific_part_count; i++) {
        unsigned int(16)    track_reference_index;
    }
    string  sdp_text;
}
```

5.3.5.7.5 Semantics

media_specific_part_count indicates the number of media-level sections in the provided SDP description. Each media-level section starts with an 'm=' line and continues to the next media-level section or the end of the whole session description.

track_reference_index is a 1-based index to the track reference type box of type 'sdpi' in the track box of the index track containing the SDP indexes. The i-th track_reference_index indicates the track which the i-th media-level section of the session description refers to. track_reference_index equal to 0 indicates that the i-th media level section corresponds to no track.

NOTE: If an RTP reception hint track is deleted from a file, the respective media level SDP section in SDP index events can be mapped to track_reference_index equal to 0 to indicate that the track is no longer present in the file.

sdp_text is a null-terminated string containing an SDP description including all media-specific parts of SDP. sdp_text is correctly formatted as a series of lines, each terminated by <crlf>, as required by SDP. sdp_text is valid starting from the time indicated in the 't=' line of the SDP.

5.3.5.8 Proprietary indexes

5.3.5.8.1 General

This provides a framework within which proprietary indexes can be stored. As the index mechanism does not use boxes, this provides the framework to store the uuid.

NOTE: A proprietary index may use a specified 4CC, assuming that this has been suitably registered.

5.3.5.8.2 4CC

This event shall be signalled by the 4CC 'uuid'.

5.3.5.8.3 Index configuration

The version field of the index characteristics class associated with events of this format shall have the value 1. The value of any_instance_info_flag shall be equal to 1, i.e., no instance information is defined within this specification for proprietary indexes.

5.3.5.8.4 Syntax

```
aligned(8) class uuidEventPayload {
    unsigned int(8)     uuid[16];
```

```
    unsigned int(8)     data[];
}
```

5.3.5.8.5 Semantics

The `uuid` field carries the UUID bytes that identify the event.

NOTE: The UUID bytes duplicate the information in the index characteristics, and are used to identify the index and instance information relevant to this event.

The `data` field is not defined by this specification, and the size is inferred from the length fields of the indexing structures, specifically the field `event_payload_length` in the index event (defined in 5.3.2.5), or, when `event_payload_length` is equal to `0xFFFF`, `var_payload_length` in the index event.

5.3.5.9 Error occurrence event

5.3.5.9.1 General

The error occurrence event is used to signal locations in the file where it is know that data is lost. Examples of this include when a transmission signal failed, or where a storage device suffered failure when content was moved or copied. The error shall indicate the first good packet after an error has occurred.

5.3.5.9.2 4CC

This event shall be signalled by the 4CC '`erro`'.

5.3.5.9.3 Index configuration

The `version` field of the index characteristics class associated with events of this format shall have the value 1.

5.3.5.9.4 Syntax

```
aligned(8) class ErrorEvent {
}
```

5.3.5.9.5 Semantics

As the class has no members, no semantics are defined.

5.3.5.10 Null index

5.3.5.10.1 General

The null index provides a generic mechanism for a range of operations such as where it is desirable to delete an index, but not desirable to alter the index data, or where it is desirable to insert space for an index.

5.3.5.10.2 4CC

This event shall be signalled by the 4CC '`null`'.

5.3.5.10.3 Index configuration

The values of the index configurations for this event are undefined, however the `index_length` should be variable to allow for any length of index to be supported.

5.3.5.10.4 Syntax

```
aligned(8) class NullEventPayload {
    unsigned int(8) data[];
}
```

5.3.5.10.5 Semantics

The semantics of `data` are undefined. This field may be of any length.

5.4 Content protection

5.4.1 Storage of CPCM Protected Content

5.4.1.1 Introduction

This clause specifies the use of the mechanism defined in clause 5.2.1.3 to signal CPCM protection of MPEG-2 TS reception hint tracks, and carry the content licence associated with the protection.

5.4.1.2 Usage of the protected scheme information box

5.4.1.2.1 Presence of sub-boxes

The ISO base file format defines some of the sub-boxes of the '`sinf`' box as optional.

In the scope of this specification, the following boxes shall be present:

- the original format box '`frma`';
- the scheme type box '`schm`'; and
- the scheme information box '`schi`'.

The usage of these boxes is defined below.

In the scope of this specification, the following boxes shall not be present:

- the IPMP info box '`imif`'.

5.4.1.2.2 CPCM scheme type

The scheme type box '`schm`' identifies the protection scheme used to protect the track and carries information about the type and version of the scheme. For CPCM, the `scheme_type` field shall take the value '`cpcm`' to signal the storage of CPCM protected content. A `scheme_version` of 1 is used to signal the CPCM encapsulation in the file format as defined in this specification.

NOTE: The CPCM version is stored inside the CPCM structures; the value of `scheme_version` signals the version of the encapsulation of CPCM, and not CPCM itself.

5.4.1.2.3 CPCM scheme information

The ISO base file format defines the scheme information box as a container box for scheme specific information. For CPCM, this box contains the CPCM Content Licence.

5.4.1.2.4 CPCM licence box

5.4.1.2.4.1 Definition

The CPCM licence box '`clic`' is a sub-box of the scheme box '`schi`'. It stores a CPCM content licence in an opaque container.

5.4.1.2.4.2 Syntax

```
class CPCMLicenceBox extends FullBox('clic', version, flags) {
    if (version == 0) {
        unsigned int(8) cpcm_licence_data[];
    } else {
        unsigned int(8) reserved[];
    }
}
```

5.4.1.2.4.3 Semantics

cpcm_licence_data[] contains binary data of the actual CPCM content licence, as detailed in TS 102 825-4 [14].

5.4.1.2.5 CPCM auxiliary data box

5.4.1.2.5.1 Definition

The CPCM auxiliary data box 'caux' is a sub-box of the scheme box 'schi'. It either stores CPCM auxiliary data in an opaque container, or contains the identifier of the metadata item that contains the auxiliary data.

It is recommended that auxiliary data larger than 5 kilobytes is stored as a metadata item, and referred to via the cpcm_auxiliary_data_metadata_item_id field.

NOTE: This box is part of the 'moov' box, and so the size of the auxiliary data contributes to the overall size of the 'moov' box. Size limits on the 'moov' box may make it difficult to contain large auxiliary data within the 'caux' box.

5.4.1.2.5.2 Syntax

```
class CPCMAuxiliaryData extends FullBox('caux', version, flags) {
    if (version == 0) {
        if (flags & 1) {
            unsigned int(8) cpcm_auxiliary_data[];
        } else {
            unsigned int(32) cpcm_auxiliary_data_metadata_item_id;
        }
    } else {
        unsigned int(8) reserved[];
    }
}
```

5.4.1.2.5.3 Semantics

The least significant bit of the flags is used to signal if the auxiliary data is stored in the 'caux' box or in a metadata box.

cpcm_auxiliary_data[] contains binary data of the CPCM auxiliary data, as detailed TS 102 825-4 [14], if it is present in this box.

Where the auxiliary data is not stored in the 'caux' box, cpcm_auxiliary_data_metadata_item_id carries the item_id of the metadata item that contains the auxiliary data. This will be located in a metadata container box, as detailed below.

5.4.1.2.6 CPCM auxiliary data metadata handler box

The handler type for CPCM auxiliary data held in a metadata box shall be 'cpad'.

5.4.1.2.7 CPCM auxiliary data metadata primary item box

The primary item box shall be present in the metadata box with handler type 'cpad'.

5.4.1.2.8 CPCM auxiliary data metadata item information box

The Item information box need not be present, may be ignored.

5.4.1.2.9 CPCM auxiliary data metadata item location box

The metadata box with handler type 'cpad' shall contain an item location box. The item_ID shall be unique within the file. Any item_id(s) listed in 'caux' box(es) shall have an entry in an item location box associated with a handler type of 'cpad'.

5.4.1.2.10 Data information box

The data information box shall be present only if the CPCM auxiliary data is stored externally to the file containing the metadata box with handler type 'cpad'.

NOTE: Certain brand constraints may prevent the auxiliary data from being stored externally.

5.4.1.2.11 Location of CPCM auxiliary data metadata

It is recommended that the location of the metadata container box containing the auxiliary data is placed in the file in advance of the content that it protects.

5.4.1.3 Supporting multiple CPCM content items

If a file contains more than one CPCM protected content item, each new CPCM content item shall correspond to a new protected MPEG-2 TS reception hint track sample entry.

5.4.1.4 Supporting CPCM Revocation Lists

CPCM Revocation Lists shall be stored in the SRM mechanism as detailed in clause 5.4.6.

5.4.2 Key message reception tracks

5.4.2.1 Introduction (informative)

This clause specifies the reception track format for key messages. The key track is based on a timed metadata track. It stores key messages that are transmitted in parallel to the audiovisual content. It can be used for any IP based broadcast/multicast system, like DVB-H or OMA BCAST. It can be linked either to a RTP reception hint track or to an elementary stream media track.

IP-based broadcast/multicast systems like DVB-H are using RTP streams for transmission of audiovisual content. For protected services the packetized audiovisual content is encrypted. Key messages are streamed in parallel to the audiovisual content to transport the keys (and other related information) necessary to decrypt the media content.

During recording of protected services every key stream is stored into a key message reception track. Key message packets are stored as samples. Key messages may be sent repeatedly, it is up to the recording device to recognize duplicated messages and not store these again.

A track reference is used to connect the key track to the corresponding reception hint track or to an elementary stream track.

5.4.2.2 Key message reception track format

The key message reception track is based on a timed metadata track. It is used to store key messages that are transmitted in parallel to the audiovisual content.

A track reference of type '`cdsc`' shall be used to connect each key message reception track to the associated media track, either an RTP reception hint track or an elementary stream track.

5.4.2.3 Sample description format

5.4.2.3.1 Description

The entry format in the sample description for the key messages reception tracks is '`keym`'. The `KeyMessageReceptionSampleEntry` is derived from the `MetadataSampleEntry`.

The format of the key sample is signalled by `key_sample_type` and `key_sample_version`. Additional boxes may be available at the end of this sample entry containing additional configuration information for the specific key sample type and version.

5.4.2.3.2 Syntax

```
class KeyMessageReceptionSampleEntry() extends MetadataSampleEntry('keym') {
    unsigned int(8) key_sample_type;
    unsigned int(8) key_sample_version;
    if (key_sample_type == 0xFF) {
        unsigned int(8) uuid[16];
```

```
    }
    box additionaldata[];
}
```

5.4.2.3.3 Semantics

`key_sample_type` is the key sample type identifier; see table 14 for specified identifier values.

Table 14: Semantics of the key sample type identifier

Key sample type	Protection system
1	DVB-H SPP OSF ECM key message type
2	DVB-H SPP 18C KSM key message type
3-254	Reserved for future use
255	Proprietary key message type specified with UUID

`key_sample_version` identifies the key sample entry version for the type of key samples signalled in `key_sample_type`; in the current version of the specification `sample_version` is `0x01`.

`uuid` is a UUID to uniquely signal a proprietary key message type.

`additionaldata` is a set of boxes that may include boxes with further configuration data for specific key sample types.

5.4.2.4 Sample format

5.4.2.4.1 Description

Each sample in the key message reception track represents one key message without additional headers. For instance, if the key messages are directly encapsulated in UDP packets without additional headers, the sample contains the plain UDP packet payload.

Sample times of the key messages are usually the reception times of the received packets.

NOTE: The size of the sample is known from the sample size table or movie fragment equivalent.

5.4.2.4.2 Syntax

```
aligned(8) class receivedKeySample {
    unsigned int(8) keyMessage[keyMessageLength];
}
```

5.4.2.4.3 Semantics

`keyMessage` is the plain key message that is e.g., the UDP packet payload.

`keyMessageLength` is the length of the key message.

NOTE: The key message length is implicitly known from the sample size table, as the sample contains no additional fields besides the key message.

5.4.3 Key stream storage of protected DVB-H IPDC services

5.4.3.1 Introduction (informative)

The IP layer of DVB-H is using RTP streams for transmission of the audiovisual content. DVB-H Service Purchase and Protection [9] specifies protection of DVB-H services and encryption of the packetized audiovisual content.

Two alternatives are available for the key management: the Open Security Framework (OSF) and 18Crypt (18C). Both are using key messages that are streamed in parallel to the audiovisual content to transport the keys (and other related information) necessary to decrypt the media content.

The Key Stream Messages (KSM) are encapsulated into UDP packets. In OSF, the KSM is also called Entitlement Control Message (ECM). One (or more) ECMs are encapsulated in one UDP packet. In 18Crypt, each KSM is encapsulated into a separate UDP packet. The KSM UDP packets are streamed on a separate port from the audiovisual content.

During recording, the KSM stream (respectively the ECM stream) is stored into a key message reception track as specified in clause 5.4.2 above.

NOTE: Both OSF and 18C support Simulcrypt, allowing multiple KMSs to control a single protected media stream. In this case several ECM streams are used in parallel, one for every KMS. During recording, every ECM stream is stored into a separate key message reception track.

5.4.3.2 18Crypt (18C)

5.4.3.2.1 Description

A key message reception track as specified in clause 5.4.2 is used to store the Key Stream Messages (KSM). As specified in TS 102 474 [9], each KSM is sent as one UDP packet. The key messages are received on a dedicated UDP port.

Signalling information related to the 18C protection scheme is part of the SDP information. The SDP parameters are specified in clause 5.2 of TS 102 474 [9] and clause 10.1 of TS 102 472 [8]. These parameters are stored in the DVB-H IPDC 18C parameters box '`d18c`' as part of the `additionaldata` of the key track sample entry (as specified in clause 5.4.2 above). The complete SDP information is saved with the SDP indexes as specified in clause 5.3.5.7. The parameters in the DVB-H IPDC 18C parameters box shall contain the same values as the equivalent parameters in the stored SDP information.

To identify the rights object needed for a particular 18Crypt stream, a `programme_CID` and `service_CID` need to be calculated, as detailed in clause B.3.2.1.3 of TS 102 474 [9]. The `serviceBaseCID` and the `OperatorID` that are needed for this calculation shall be stored as detailed below.

The sample entry type of the corresponding RTP reception hint track is not modified.

It is possible that there may be multiple key tracks associated with the same RTP reception hint track.

5.4.3.2.2 Syntax

```
class DvbhIpdc18Cparameters() extends Box('d18c') {
    unsigned int(32)    IPDCKMSId;
    string              IPDCOperatorId;
    string              IPDCAccessRights;
    string              serviceBaseCID;
}
```

5.4.3.2.3 Semantics

`IPDCKMSId` identifies the KMS.

`IPDCOperatorId` identifies the operator. `IPDCOperatorID` is mapped to `socID`, as defined in IEC 62455 [25]

`IPDCAccessRights` is an optional description or URL of the access rights associated with the content.

`serviceBaseCID` is the base ID of the service as defined in IEC 62455 [25], used as part of the CID of all service and program keys related to the service.

5.4.3.3 Open Security Framework (OSF)

5.4.3.3.1 Description

A key message reception track as specified in clause 5.4.2 is used to store the Entitlement Control Messages (ECM). The ECMs are received on a dedicated UDP port. As specified in TS 102 474 [9], one UDP packet contains one or more ECMs without any additional header.

Signalling information related to the OSF protection scheme is part of the SDP information, the SDP parameters are specified in TS 102 474 [9]. The equivalent OSF parameters are stored in the DVB-H IPDC OSF parameters box '`dosf`' as part of the `additionaldata` of the key track sample entry (as specified in clause 5.4.2 above). The complete SDP information is saved with the SDP indexes as specified in clause 5.3.5.7. The parameters in the DVB-H IPDC OSF parameters box shall contain the same values as the equivalent parameters in the stored SDP information.

In case Simulcrypt is used, a separate ECM stream is sent for each KMS. During recording every ECM stream is stored into a separate key message reception track. The `CA_system_id` field and the `OperatorId` field are used to establish the relationship between key track and the corresponding KMS.

The sample entry type of the corresponding RTP reception hint track is not modified.

5.4.3.3.2 Syntax

```
class DvbhIpdcOSFparameters() extends Box('dosf') {
    unsigned int(16)    CA_system_id;
    unsigned int(16)    OperatorId;
}
```

5.4.3.3.3 Semantics

`CA_system_id` identifies the KMS applicable for the associated ECM, as defined in ETR 162 [2].

`OperatorId` identifies the operator using the associated ECM. Allocation of the values of this field is under the control of the KMS identified by `CA_system_id`, allowing for differentiation between operators using the same KMS.

5.4.4 Key tracks and ISMACryp elementary stream media tracks

Key tracks as specified in clause 5.4.2 may also be used in combination with audiovisual elementary stream tracks that are encrypted using ISMACryp [23].

NOTE: ISMACryp is designed in a way that allows depacketization of audiovisual data from RTP payloads to elementary stream access units without decryption. Thus, the RTP reception track can be transformed to an elementary stream track.

The key messages in the key track that corresponds to the RTP reception track remain unmodified. The `key_indicator` field serves as the link between access unit and the corresponding key message. To simplify playback, it is recommended that the timing of the key track is aligned to the elementary stream track, such that the key message sample that contains the key with a specific `key_indicator` is aligned to the access unit that first uses the key with that same `key_indicator`.

5.4.5 Proprietary protection schemes

The ISO base file format offers a generic solution to signal that a media track is protected and to store protection scheme related information in the sample description (see clause 8.12 of ISO/IEC 14496-12 [30]). In the case of proprietary protected media tracks, this mechanism shall be used.

In the case of proprietary protected MPEG-2 Transport Stream reception hint tracks, the mechanism outlined in section 5.2.1.3 shall be used to mark the track as protected.

In the case of proprietary protected RTP reception hint tracks, the mechanism outlined in section 5.2.1.5 shall be used to mark the track as protected.

These methods can be used to signal any proprietary protection scheme. A scheme type identifier has to be specified for every protection scheme and preferably be registered with the MPEG-4 Registration Authority (see Annex D of ISO/IEC 14496-12 [30]). If the scheme type is unknown to the playback device, the device shall stop further parsing of the sample description and shall not playback this track.

5.4.6 System Renewability Messages

5.4.6.1 Introduction (informative)

System Renewability Messages (SRMs) are a generic way of conveying messages that may be required by systems outside the scope of this file format. These systems may form part of the consumption chain of the content contained in the file, and the passage of the content through these systems may require the presence of this information. The format of the SRMs is outside the scope of this document, and defined by the system specified by the identifier associated with each SRM. One example of SRM messages are the CPCM revocation lists TS 102 825 [14].

It is recommended that, where a recording is made from a broadcast system, such SRMs as are present at the time of recording are included in the file. A file that is authored by a content provider may choose to include such SRMs as they deem needed.

5.4.6.2 System Renewability Message container box

5.4.6.2.1 Description

The System Renewability Messages container box '`srmc`' may be used to store a list of System Renewability Messages (SRM). If present in the file, the SRM container box shall occur at the top level of the file hierarchy.

5.4.6.2.2 Syntax

```
aligned(8) class SRMContainerBox extends FullBox('srmc', 0, flags) {
    unsigned int(32)    entry_count;
    for (i=1; i<=entry_count; i++)
        SRMBox  srm;
}
```

5.4.6.2.3 Semantics

`entry_count` is the number of SRMs stored in the '`srmc`' box.

`srm` is the system renewability message box as defined in clause 5.4.6.3.

5.4.6.3 System Renewability Message box

5.4.6.3.1 Description

The System Renewability Message box '`srmb`' may be used to store a System Renewability Message (SRM). The format of the SRM and the usage of the SRM are defined by the relevant KMS.

5.4.6.3.2 Syntax

```
aligned(8) class SRMBox extends FullBox('srmb', 0, flags) {
    unsigned int(16)    CP_system_id;
    unsigned int(8)     SystemRenewabilityMessage[];
}
```

5.4.6.3.3 Semantics

`CP_system_id` identifies the Copy Protection (CP) system that this system renewability message applies to, and therefore specifies who defines the format of the message. The values are managed by DVB as per TR 101 162 (v1.2.4) [2].

NOTE: Only certain Copy Protection systems may make use of SRMs and have CP_system_id for this use.

`SystemRenewabilityMessage` is the binary data of the system renewability message.

5.4.7 Storage of Key Management Messages

5.4.7.1 Overview

Key Management Messages (KMM) or Rights Objects (RO) (defined in TS 102 474 [9]), in OSF also known as Entitlement Messages (EMM), contain content/service access rights and enable authorized users to access the Key Stream Messages. They are either infrequently sent in-band or are acquired out-of-band using an interactive channel.

One of the following methods should be used to store the KMM:

- The KMM should be stored as a metadata object, compliant with the framework detailed below.
- Protection mechanisms for which the storage of rights objects in an ISO base file format compatible file is already defined should use that storage mechanism. In the case of OMA DRM, this is described in OMA-DRM-DCF [34] and consists of storing the `OMADRMKMSBox` (clause 7.1.5 of OMA-DRM-DCF [34]) in the

ProtectionSchemeInfoBox as well as a MutableDRMInformation box (clause 5.2.4 of OMA-DRM-DCF [34]) at file level if one or more right objects are stored in the file.

The metadata object shall be stored in a metadata box in a metadata container box at file level, as defined below. The format of the messages stored as Key Management Messages is defined by the relevant Key Management System. The Key Management System for a given message is defined in the item information box.

This clause, and its sub-clauses, applies only to Key Management Messages stored within the framework defined in this specification.

5.4.7.2 Handler box

The handler type for key management messages shall be 'skmm'. There shall be only one metadata box with handler type 'skmm' present in a file.

NOTE: This box may contain multiple key messages, and messages for more than one system.

5.4.7.3 Primary item box

The primary item box shall be present in the metadata box with handler type 'skmm'. If more than one item exists in the metadata, then the choice of primary item is made by the creator of the file.

NOTE: Typically the protection mechanism will identify which is the "primary" item, e.g., which one should be presented to the protection system first.

5.4.7.4 Item information box

The item information box shall be present in the metadata box with handler type 'skmm', and shall contain one item information entry for each key management message present (and therefore for each item_ID given in the item location box). The item information entry shall have a version of 1, and shall contain at least one item information extension of the type SKMMItemInfoExtension.

In the item information entry:

- the field item_name may contain a null string; and
- the field content_type may contain null string, if none is defined for the key mechanism.

5.4.7.4.1 SKMM item information extension

5.4.7.4.1.1 Overview

The SKMM item information extension is used to signal the Key Management System to which the messages apply.

5.4.7.4.1.2 Syntax

```
class SKMMItemInfoExtension() extends ItemInfoExtension('skid') {
    unsigned int(8) key_management_message_type;
    unsigned int(8) key_management_message_version;
    if (key_sample_type == 0xFF) {
        unsigned int(8) uuid[16];
    }
    box additionaldata[];
}
```

5.4.7.4.1.3 Semantics

key_management_message_type indicates to which key management system this message applies. This field shall take values from table 14 above.

key_management_message_version identifies the key management message version or the referenced message; in the current version of the specification this is 0x01.

uuid is a UUID and may be used to uniquely signal a proprietary key management system for which no specific value is defined in table 14.

`additionaldata` is a set of boxes that may include boxes with further configuration data for a specific key management system.

5.4.7.5 Item location box

The metadata box with handler type '`skmm`' shall contain an item location box. For each metadata item present, there shall be an entry in the item location box that maps the Key Management Message to an `item_ID`. Each `item_ID` shall be unique within the file.

NOTE: This means that a given `item_ID` cannot refer to both a key management message and any other item of metadata, such as descriptive metadata.

5.4.7.6 Data information box

The data information box shall be present only if the Key Management Messages are stored externally to the file containing the metadata box with handler type '`skmm`'.

6 Proprietary extensions

It is reasonable to expect that some implementations of the file format will wish to provide proprietary extensions. This section provides definitions that minimise interoperability problems. Where appropriate, extension mechanisms have already been defined in the clauses above.

6.1 General issues

Any extensions shall follow the guidelines in Annex C of the ISO base file format [30].

It is recommended that proprietary extensions do not contain standard defined and published boxes.

NOTE: Encapsulating standard boxes within a proprietary box may "hide" them from a player, and as such reduce the interoperability (or functionality obtained from this) of the file format.

6.2 Use of UUID

It is strongly recommended that any proprietary extensions use the 4CC '`uuid`', as defined in clause 4.2 of ISO/IEC 14496-12 [30]. This mechanism utilises a 16-byte UUID as defined in ISO/IEC 9834-8 [26] to identify the box type, and this shall be allocated by the body responsible for the proprietary extension.

NOTE: The nature of UUIDs essentially deals with the issues of a code-point clash which might generate an interoperability issue.

A box of type '`uuid`' may occur at any location in the file, and may occur any number of times.

A box of type '`uuid`' may be ignored and skipped.

6.3 Use of 4CC

No 4CC shall be present in the file that has not been registered with the MPEG-4 Registration Authority, as explained in Annex D of ISO/IEC 14496-12 [30].

NOTE: This means that proprietary extensions within a DVB File that do not make use of the '`uuid`' mechanism have to be registered. This is to ensure that there are no clashes with other 4CC definitions.

A player shall cleanly ignore and skip any unknown 4CC types, as per ISO/IEC 14496-12 [30] specification.

6.4 Use of DVB brands in the presence of proprietary extensions

It is strongly recommended that the appropriate DVB (and compatible) brands are set in the appropriate parts of the file type box.

Annex A:
MIME types for DVB files

A.1 General

This annex registers the MIME types for files created according to this specification with the suffix as defined in clause 4.1.1 and a major brand as defined in clause 4.2.

A.2 Files with audio but no visual content

The type "audio/vnd.dvb.file" may be used for files that contain no other tracks than audio tracks.

Type name: audio

Subtype name: vnd.dvb.file

Required parameters: none

Optional parameters:

codecs: is a single value or a comma-separated list that identifies the codec(s) needed for rendering the content contained (in tracks) of a file. The codecs parameter is defined in RFC 4281 [19]. The ISO file format name space and ISO syntax in clauses 3.2 and 3.3 of RFC 4281 [19], respectively, shall be used.

protected-content: is value '1' if some or all of the media data in the file is protected, otherwise '0'.

brands: is a field which contains a textual representation of the file type box of the DVB File. It consists of dot-separated elements, enclosed in double quotes. The first element is the major brand as a printable four-character code, followed by the second element, which is the minor version of the major brand. The following elements are the compatible brands of the file as printable four-character codes.

Example: `brands="dvt1.0.iso3"`

Encoding considerations: files are binary and should be transmitted in a suitable encoding without CR/LF conversion, 7-bit stripping etc.; base64 (RFC 4648 [21]) is a suitable encoding.

Security considerations: DVB files may contain audio, video, displayable text data images, ESG metadata, etc. Clearly it is possible to author malicious files which attempt to call for e.g., an excessively large picture size, high-sampling rate audio, etc. However, clients can and usually do protect themselves against this kind of attack. It should be noted that the file may contain fields that encompass information partly intended to protect media against unauthorized use or distribution. In this case, the intention is that alteration or removal of the data in these fields would be treated as an offense under national agreements based on World Intellectual Property Organization (WIPO) treaties.

Interoperability considerations: DVB has defined this specification. Interoperability considerations will be in the corresponding Implementation Guidelines document.

Published specification:

Applications which use this media type: Multi-media

Additional information: The type "audio/vnd.dvb.file" may be used for files containing audio but no visual presentation. Files served under this type must not contain any visual material. (Note that timed text is visually presented and is considered to be visual material).

Magic number(s): None. However, the file type box must occur first in the file, and must have a DVB brand as the major brand.

File extension(s): 'dvb'

Macintosh File Type Code(s): none

Person & email address to contact for further information:

Peter MacAvock
Digital Video Broadcasting (DVB)
macavock@dvb.org

Intended usage: COMMON

Restrictions on usage: Note that this MIME type is used only for files; separate types are used for real-time transfer, such as for the RTP payload formats.

Authors:

Members of the Digital Video Broadcasting Project - Technical Module File Format (DVB TM-FF).

Change controller: DVB TM / DVB TM-FF

A.3 Any files

The type "video/vnd.dvb.file" is valid for all files, including files that contain audio tracks only. It is valid to serve an audio-only file as "video/vnd.dvb.file".

Type name: video

Subtype name: vnd.dvb.file

Required parameters: none

Optional parameters:

codecs: is a single value or a comma-separated list that identifies the codec(s) needed for rendering the content contained (in tracks) of a file. The codecs parameter is defined in RFC 4281 [19]. The ISO file format name space and ISO syntax in clauses 3.2 and 3.3 of RFC 4281 [19], respectively, shall be used.

protected-content: is value '1' if some or all of the media data in the file is protected, otherwise '0'.

brands: is a field which contains a textual representation of the file type box of the DVB File. It consists of dot-separated elements, enclosed in double quotes. The first element is the major brand as a printable four-character code, followed by the second element, which is the minor version of the major brand in decimal representation. The following elements are the compatible brands of the file as printable four-character codes.

Example: `brands="dvt1.0.iso3"`

Encoding considerations: files are binary and should be transmitted in a suitable encoding without CR/LF conversion, 7-bit stripping etc.; base64 (RFC 4648 [21]) is a suitable encoding.

Security considerations: DVB files may contain audio, video, displayable text data images, ESG metadata, etc. Clearly it is possible to author malicious files which attempt to call for e.g., an excessively large picture size, high-sampling rate audio, etc. However, clients can and usually do protect themselves against this kind of attack. It should be noted that the file may contain fields that encompass information partly intended to protect media against unauthorized use or distribution. In this case, the intention is that alteration or removal of the data in these fields would be treated as an offense under national agreements based on World Intellectual Property Organization (WIPO) treaties.

Interoperability considerations: DVB has defined this specification. Interoperability considerations will be in the corresponding Implementation Guidelines document.

Published specification:

Applications which use this media type: Multi-media

Additional information:

Magic number(s): None. However, the file type box must occur first in the file, and must have a DVB brand as the major brand.

File extension(s): 'dvb'

Macintosh File Type Code(s): none

Person & email address to contact for further information:

Peter MacAvock
Digital Video Broadcasting (DVB)
macavock@dvb.org

Intended usage: COMMON

Restrictions on usage: Note that this MIME type is used only for files; separate types are used for real-time transfer, such as for the RTP payload formats.

Authors:

Members of the Digital Video Broadcasting Project - Technical Module File Format (DVB TM-FF).

Change controller: DVB TM / DVB TM-FF

A.4 ABNF Syntax for optional parameters

The BNF uses the rules specified in RFC 4234 [18].

```
brands              :=  DQUOTE major-brand [minor-version [compatible-brands]] DQUOTE

major-brand         :=  FOURCC  ; each brand is a FOURCC with printable characters only

minor-version       :=  DOT 1*DIGIT  ; decimal representation of the minor version,
                        unsigned 32-bit integer range (which is 0 to 4294967295)

compatible-brands   :=  *(DOT compatible-brand)  ; the list of compatible brands

compatible-brand    :=  FOURCC  ; each brand is a FOURCC with printable characters only

FOURCC              :=  4PRINTABLE-CHAR  ; exactly four printable characters

PRINTABLE-CHAR      :=  %x20-7E  ; one printable character from the US-ASCII charset

DOT                 :=  %x2E  ; "."

content-protection  :=  BIT
```

Annex B (informative): Recording and parsing procedure for RTP streams

B.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 transmission errors, but only players capable of parsing RTP reception hint track 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 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.

The operation of a recording unit for each of the basic file structure above is described in clauses B.2, B.3, and B.4, respectively.

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.

The file structures and the operation of a recording unit are described here without reference to movie fragments. However, movie fragments can be used in files containing recorded RTP stream. For example, when streams are stored to and played from the same file simultaneously, movie fragments are necessary. Furthermore, movie fragments can be used to control the consumption of the operating memory used for maintaining the data structures for the movie box.

B.2 Creation of a file with RTP reception hint tracks only

RTP reception hint tracks enable detection of packet losses. Consequently, parsers capable of processing RTP reception hint tracks may handle packet losses more gracefully compared to the playback of media tracks generated from received streams.

A recording operation creating one reception hint sample, i.e., one `receivedRTPsample` structure, per each RTP packet is described next. The `receivedRTPsample` structure is set to contain one `receivedRTPpacket` containing one packet constructor of type 2. The pre-computed packet payload is copied to the `extradata` section of the sample. The track reference of the 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.

If media frames are not both fragmented and interleaved in an RTP stream, 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 recording operation for storing all packets sharing the same RTP timestamps as one pre-computed reception hint sample is very similar to the one described above. The RTP payloads (without the RTP header extension) sharing the same RTP timestamp are stored sequentially to `extradata` of the `receivedRTPsample` structure.

Depending on the capabilities of the implementation the streams may be recorded and be either synchronous or contain an associated RTCP reception hint track.

B.3 Creation of a file with media tracks only

Media samples are created from the received RTP packets as instructed by the relevant RTP payload specification and RTP itself [22]. 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 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.

Synchronous media streams can be achieved by parsing the incoming RTCP stream in addition to the RTP stream. This procedure is out of scope for this document and evident from IETF RFC 3550 [22].

B.4 Creation of a file with RTP reception hint tracks and media tracks

Media samples and tracks are created from the received RTP packets as explained in clause B.3. RTP reception hint tracks are created as explained in clause B.2, but the creation of `receivedRTPpacket` is conditional 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 `receivedRTPpacket` structure is created as explained in clause B.2.

Associated RTCP reception hint tracks can be used to recover the timing relations of several RTP reception hint tracks so that the media streams are in sync. This is not necessary if the RTP reception hint tracks are marked to be in sync already by the timestamp synchrony box '`tssy`'.

B.5 Parsing procedure

File players should prefer an RTP reception hint track over any other tracks in the same alternate group, indicated by the same value of `alternate_group` in the track header box. File players may re-create the packet stream that was received based on the reception hint tracks and process the re-created packet stream as if it was newly received. The reaction to packet losses depends on the particular media decoder implementation and may also depend on user preferences. Hence, no guidelines on handling of transmission errors are given here. However, at minimum, file players should play the media track in the same alternate group as the RTP reception hint track.

Annex C (informative)
Example mapping of the mandatory basic description metadata to ETSI EN 300 468, ETSI TS 102 323, ETSI TS 102 539 and ETSI TS 102 471 metadata

This clause proposes an example mapping between the information elements defined by the mandatory basic description and DVB SI [1], TV-Anytime over transport streams [5], IPTV BCG [10] and IPDC ESG [7] metadata elements respectively. Other mappings may of course be established which may be more advantageous in specific situations. As stated in clause 5.1.4.1, the data types of the mandatory basic description are heavily based on the TV-Anytime [12] definitions which are also re-used in TV-Anytime over transport streams (TVA-TS) [5] and IPTV BCG [10]. A mapping to or from metadata based on those definitions should therefore expected to be simple. Hence the more detailed mapping examples throughout the rest of this clause focus on mapping to/from IPDC ESG [7] metadata. Absence of a proposed mapping for a particular element in this clause does not imply that such mapping could not be established or were particularly difficult to establish.

- `FileContentItemInformation.ContentItemInformation.BasicDescription` element
  - In DVB-SI, the language information is located in the `iso639_language_descriptor` in the PMT and most of the other information is located in the EIT.
  - In TVA-TS, the elements map directly to the elements of the same name in the `TVAMain.ProgramDescription.ProgramInformationTable.ProgramInformation.BasicDescription` element.
  - In IPTV BCG, the elements map directly to the elements of the same name in the `TVAMain.ProgramDescription.ProgramInformationTable.ProgramInformation.BasicDescription` element.
  - In IPDC ESG, most of the elements map directly to the elements of the same or similar name in the `ESGMain.ESG.ContentTable.Content` element.
- `FileContentItemInformation.ContentItemInformation.AVAttributes` element
  - In DVB-SI, the information for the `FileFormat`, `FileSize`, `System` and `Bitrate` elements needs to be determined from the characteristics of the transport stream during reception. Information for the `AudioAttributes`, `VideoAttributes` and `CaptioningAttributes` elements can be found in the PMT and in the respective elementary streams during reception.
  - In TVA-TS, the elements map directly to the elements of the same name in the `TVAMain.ProgramDescription.ProgramInformationTable.ProgramInformation.AVAttributes` element.
  - In IPTV BCG, the elements map directly to the elements of the same name in the `TVAMain.ProgramDescription.ProgramInformationTable.ProgramInformation.AVAttributes` element.
  - In IPDC ESG, the most important elements map to elements in the `ESGMain.ESG.AcquisitionTable.Acquisition` element.
- `FileContentItemInformation.BroadcastServicename` element
  - In DVB-SI, this information is conveyed in the `service_descriptor` and/or `multilingual_service_descriptor` in the SDT.
  - In TVA-TS, the elements map directly to the `TVAMain.ProgramDescription.ServiceInformationTable.ServiceInformation.Name` element.
  - In IPTV BCG, the elements map directly to the `TVAMain.ProgramDescription.ServiceInformationTable.ServiceInformation.Name` element.
  - In IPDC ESG, this information is conveyed in the `ESGMain.ESG.ServiceTable.Service.ServiceName` element.

- `FileContentItemInformation.ContentItemInformation.ScheduleEvent` element

  - In DVB-SI, this information is conveyed in the EIT.

  - In TVA-TS, the elements map directly to the elements of the same name in the `TVAMain.ProgramDescription.ProgramLocationTable.Schedule` or `TVAMain.ProgramDescription.ProgramLocationTable.BroadcastEvent` elements.

  - In IPTV BCG, the elements map directly to the elements of the same name in the `TVAMain.ProgramDescription.ProgramLocationTable.Schedule` or `TVAMain.ProgramDescription.ProgramLocationTable.BroadcastEvent` elements.

  - In IPDC ESG, most of the elements map directly to the elements of the same or similar name in the `ESGMain.ESG.ScheduleEventTable.ScheduleEvent` element.

As a further example, the listing below shows an XSLT for mapping IPDC ESG to the Mandatory Basic Description.

Listing C.1: Example IPDC ESG to mandatory basic description XSLT

```
<?xml version='1.0' encoding='UTF-8'?>

<!-- ########################################################################
     DVB IPDC to DVB File Format Mandatory Description Translation Stylesheet
     (c) 2008 Micronas GmbH
     All rights reserved.
     ######################################################################## -->

<?xe.source dvb_ipdc_esg_v1.2.1.xsd#ESGMain?>
<?xe.target FileContentItemDescription.xml#FileContentItemInformation?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:dvbfile="urn:dvb:metadata:schema:fileContentItemInformation:2007"
    xmlns:dvb="urn:dvb:metadata:schema:contentItemInformation:2007"
    xmlns:esg="urn:dvb:ipdc:esg:2005"
    xmlns:tva-2005="urn:tva:metadata:2005"
    xmlns:tva-2007="urn:tva:metadata:2007"
    xmlns:mpeg7-2001="urn:mpeg:mpeg7:schema:2001"
    xmlns:mpeg7-2005="urn:tva:mpeg7:2005"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">
    <xsl:template match="/">
        <dvbfile:FileContentItemInformation>
            <dvbfile:ContentItemInformation>
                <dvb:BasicDescription>
                    <dvb:Title>
                        <xsl:attribute name="lang"
                            namespace="http://www.w3.org/XML/1998/namespace">
                        <xsl:value-of
                select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Title/@xml:lang" />
                        </xsl:attribute>
                        <xsl:attribute name="type">
                            <xsl:value-of
                select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Title/@type" />
                        </xsl:attribute>
                        <xsl:value-of
                            select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Title" />
                    </dvb:Title>
                    <dvb:MediaTitle>
                        <mpeg7-2005:TitleImage>
                            <mpeg7-2005:MediaUri>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleImage/mpeg7-
2001:MediaUri" />
                            </mpeg7-2005:MediaUri>
                            <mpeg7-2005:StreamID>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleImage/mpeg7-
2001:StreamID" />
                            </mpeg7-2005:StreamID>
                            <mpeg7-2005:MediaTimePoint>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleImage/mpeg7-
2001:MediaTimePoint" />
                            </mpeg7-2005:MediaTimePoint>
                        </mpeg7-2005:TitleImage>
                        <mpeg7-2005:TitleVideo>
                            <mpeg7-2005:MediaUri>
```

```
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleVideo/mpeg7-
2001:MediaUri" />
                            </mpeg7-2005:MediaUri>
                            <mpeg7-2005:StreamID>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleVideo/mpeg7-
2001:StreamID" />
                            </mpeg7-2005:StreamID>
                            <mpeg7-2005:MediaTime>
                                <mpeg7-2005:MediaTimePoint>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleVideo/mpeg7-
2001:MediaTime/mpeg7-2001:MediaTimePoint" />
                                </mpeg7-2005:MediaTimePoint>
                                <mpeg7-2005:MediaDuration>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleVideo/mpeg7-
2001:MediaTime/mpeg7-2001:MediaDuration" />
                                </mpeg7-2005:MediaDuration>
                            </mpeg7-2005:MediaTime>
                        </mpeg7-2005:TitleVideo>
                        <mpeg7-2005:TitleAudio>
                            <mpeg7-2005:MediaUri>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleAudio/mpeg7-
2001:MediaUri" />
                            </mpeg7-2005:MediaUri>
                            <mpeg7-2005:StreamID>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleAudio/mpeg7-
2001:StreamID" />
                            </mpeg7-2005:StreamID>
                            <mpeg7-2005:MediaTime>
                                <mpeg7-2005:MediaTimePoint>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleAudio/mpeg7-
2001:MediaTime/mpeg7-2001:MediaTimePoint" />
                                </mpeg7-2005:MediaTimePoint>
                                <mpeg7-2005:MediaDuration>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:MediaTitle/mpeg7-2001:TitleAudio/mpeg7-
2001:MediaTime/mpeg7-2001:MediaDuration" />
                                </mpeg7-2005:MediaDuration>
                            </mpeg7-2005:MediaTime>
                        </mpeg7-2005:TitleAudio>
                    </dvb:MediaTitle>
                    <dvb:Synopsis>
                        <xsl:attribute name="lang"
                            namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Synopsis/@xml:lang" />
                        </xsl:attribute>
                        <xsl:value-of
                        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Synopsis" />
                    </dvb:Synopsis>
                    <dvb:Keyword>
                        <xsl:attribute name="lang"
                            namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
                select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Keyword/@xml:lang" />
                        </xsl:attribute>
                        <xsl:attribute name="type"><xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Keyword/@type" />
                        </xsl:attribute>
                        <xsl:value-of
                            select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Keyword" />
                    </dvb:Keyword>
                    <dvb:Genre>
                        <xsl:attribute name="href"><xsl:value-of
                        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/@href" />
                        </xsl:attribute>
                        <xsl:attribute name="type"><xsl:value-of
                        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/@type" />
                        </xsl:attribute>
                        <tva-2007:Name>
                            <xsl:attribute name="lang"
                                namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/tva-2005:Name/@xml:lang" />
                            </xsl:attribute>
                            <xsl:attribute name="preferred"><xsl:value-of
    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/tva-2005:Name/@preferred" />
```

```
                            </xsl:attribute>
                            <xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/tva-2005:Name" />
                        </tva-2007:Name>
                        <tva-2007:Definition>
                            <xsl:attribute name="lang"
                                namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/tva-2005:Definition/@xml:lang" />
                            </xsl:attribute>
                            <xsl:value-of
            select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Genre/tva-2005:Definition" />
                        </tva-2007:Definition>
                    </dvb:Genre>
                    <dvb:ParentalGuidance>
                        <mpeg7-2005:ParentalRating>
                            <xsl:attribute name="href"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-
2001:ParentalRating/@href" />
                            </xsl:attribute>
                            <mpeg7-2005:Name>
                                <xsl:attribute name="lang"
                                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-
2001:ParentalRating/mpeg7-2001:Name/@xml:lang" />
                                </xsl:attribute>
                                <xsl:attribute name="preferred"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-
2001:ParentalRating/mpeg7-2001:Name/@preferred" />
                                </xsl:attribute>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-
2001:ParentalRating/mpeg7-2001:Name" />
                            </mpeg7-2005:Name>
                            <mpeg7-2005:Definition>
                                <xsl:attribute name="lang"
                                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-
2001:ParentalRating/mpeg7-2001:Definition/@xml:lang" />
                                </xsl:attribute>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-
2001:ParentalRating/mpeg7-2001:Definition" />
                            </mpeg7-2005:Definition>
                        </mpeg7-2005:ParentalRating>
                        <mpeg7-2005:Region>
                            <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:ParentalGuidance/mpeg7-2001:Region" />
                        </mpeg7-2005:Region>
                    </dvb:ParentalGuidance>
                    <dvb:Language>
                        <xsl:attribute name="type"><xsl:value-of
                        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Language/@type" />
                        </xsl:attribute>
                        <xsl:attribute name="supplemental"><xsl:value-of
                select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Language/@supplemental" />
                        </xsl:attribute>
                        <xsl:value-of
                        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Language" />
                    </dvb:Language>
                    <dvb:CaptionLanguage>
                        <xsl:attribute name="closed"><xsl:value-of
                select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CaptionLanguage/@closed" />
                        </xsl:attribute>
                        <xsl:attribute name="supplemental"><xsl:value-of
        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CaptionLanguage/@supplemental" />
                        </xsl:attribute>
                        <xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CaptionLanguage" />
                    </dvb:CaptionLanguage>
                    <dvb:SignLanguage>
                        <xsl:attribute name="primary"><xsl:value-of
                select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:SignLanguage/@primary" />
                        </xsl:attribute>
                        <xsl:attribute name="translation"><xsl:value-of
            select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:SignLanguage/@translation" />
                        </xsl:attribute>
                        <xsl:attribute name="type"><xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:SignLanguage/@type" />
                        </xsl:attribute>
                        <xsl:value-of
```

```
                    select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:SignLanguage" />
                    </dvb:SignLanguage>
                    <dvb:CreditsList>
                        <tva-2007:CreditsItem>
                            <xsl:attribute name="role"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/@role"
/>
                            </xsl:attribute>
                            <tva-2007:PersonName>
                                <xsl:attribute name="dateFrom"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/@dateFrom" />
                                </xsl:attribute>
                                <xsl:attribute name="dateTo"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/@dateTo" />
                                </xsl:attribute>
                                <xsl:attribute name="lang"
                                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/@xml:lang" />
                                </xsl:attribute>
                                <xsl:attribute name="type"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/@type" />
                                </xsl:attribute>
                                <mpeg7-2005:GivenName>
                                    <xsl:attribute name="abbrev"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/mpeg7-2001:GivenName/@abbrev" />
                                    </xsl:attribute>
                                    <xsl:attribute name="initial"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/mpeg7-2001:GivenName/@initial" />
                                    </xsl:attribute>
                                    <xsl:attribute name="lang"
namespace="http://www.w3.org/XML/1998/namespace">
                                        <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/mpeg7-2001:GivenName/@xml:lang" />
                                    </xsl:attribute>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:PersonName/mpeg7-2001:GivenName" />
                                </mpeg7-2005:GivenName>
                            </tva-2007:PersonName>
                            <tva-2007:Character>
                                <xsl:attribute name="dateFrom"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/@dateFrom" />
                                </xsl:attribute>
                                <xsl:attribute name="dateTo"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/@dateTo" />
                                </xsl:attribute>
                                <xsl:attribute name="type"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/@type" />
                                </xsl:attribute>
                                <xsl:attribute name="lang"
                                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/@xml:lang" />
                                </xsl:attribute>
                                <mpeg7-2005:GivenName>
                                    <xsl:attribute name="abbrev"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/mpeg7-2001:GivenName/@abbrev" />
                                    </xsl:attribute>
                                    <xsl:attribute name="initial"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/mpeg7-2001:GivenName/@initial" />
                                    </xsl:attribute>
                                    <xsl:attribute name="lang"
                                        namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-
of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/mpeg7-2001:GivenName/@xml:lang" />
                                    </xsl:attribute>
                                    <xsl:value-of
```

```
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:CreditsList/tva-2005:CreditsItem/tva-
2005:Character/mpeg7-2001:GivenName" />
                                </mpeg7-2005:GivenName>
                            </tva-2007:Character>
                        </tva-2007:CreditsItem>
                    </dvb:CreditsList>
                    <dvb:RelatedMaterial>
                        <tva-2007:HowRelated>
                            <xsl:attribute name="href"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:HowRelated/@href"
/>
                            </xsl:attribute>
                            <tva-2007:Name>
                                <xsl:attribute name="lang"
                                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:HowRelated/tva-
2005:Name/@xml:lang" />
                                </xsl:attribute>
                                <xsl:attribute name="preferred"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:HowRelated/tva-
2005:Name/@preferred" />
                                </xsl:attribute>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:HowRelated/tva-
2005:Name" />
                            </tva-2007:Name>
                            <tva-2007:Definition>
                                <xsl:attribute name="lang"
                                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:HowRelated/tva-
2005:Definition/@xml:lang" />
                                </xsl:attribute>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:HowRelated/tva-
2005:Definition" />
                            </tva-2007:Definition>
                        </tva-2007:HowRelated>
                        <tva-2007:MediaLocator>
                            <mpeg7-2005:MediaUri>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:MediaLocator/mpeg7-
2001:MediaUri" />
                            </mpeg7-2005:MediaUri>
                            <mpeg7-2005:StreamID>
                                <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:MediaLocator/mpeg7-
2001:StreamID" />
                            </mpeg7-2005:StreamID>
                        </tva-2007:MediaLocator>
                        <tva-2007:PromotionalText>
                            <xsl:attribute name="lang"
                                namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalText/@xm
l:lang" />
                            </xsl:attribute>
                            <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalText" />
                        </tva-2007:PromotionalText>
                        <tva-2007:PromotionalMedia>
                            <mpeg7-2005:TitleImage>
                                <mpeg7-2005:MediaUri>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleImage/mpeg7-2001:MediaUri" />
                                </mpeg7-2005:MediaUri>
                                <mpeg7-2005:StreamID>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleImage/mpeg7-2001:StreamID" />
                                </mpeg7-2005:StreamID>
                                <mpeg7-2005:MediaTimePoint>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleImage/mpeg7-2001:MediaTimePoint" />
                                </mpeg7-2005:MediaTimePoint>
                            </mpeg7-2005:TitleImage>
                            <mpeg7-2005:TitleVideo>
                                <mpeg7-2005:MediaUri>
                                    <xsl:value-of
```

```
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleVideo/mpeg7-2001:MediaUri" />
                                </mpeg7-2005:MediaUri>
                                <mpeg7-2005:StreamID>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleVideo/mpeg7-2001:StreamID" />
                                </mpeg7-2005:StreamID>
                                <mpeg7-2005:MediaTime>
                                    <mpeg7-2005:MediaTimePoint>
                                        <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleVideo/mpeg7-2001:MediaTime/mpeg7-2001:MediaTimePoint" />
                                    </mpeg7-2005:MediaTimePoint>
                                    <mpeg7-2005:MediaDuration>
                                        <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleVideo/mpeg7-2001:MediaTime/mpeg7-2001:MediaDuration" />
                                    </mpeg7-2005:MediaDuration>
                                </mpeg7-2005:MediaTime>
                            </mpeg7-2005:TitleVideo>
                            <mpeg7-2005:TitleAudio>
                                <mpeg7-2005:MediaUri>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleAudio/mpeg7-2001:MediaUri" />
                                </mpeg7-2005:MediaUri>
                                <mpeg7-2005:StreamID>
                                    <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleAudio/mpeg7-2001:StreamID" />
                                </mpeg7-2005:StreamID>
                                <mpeg7-2005:MediaTime>
                                    <mpeg7-2005:MediaTimePoint>
                                        <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleAudio/mpeg7-2001:MediaTime/mpeg7-2001:MediaTimePoint" />
                                    </mpeg7-2005:MediaTimePoint>
                                    <mpeg7-2005:MediaDuration>
                                        <xsl:value-of
select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:RelatedMaterial/esg:PromotionalMedia/mp
eg7-2001:TitleAudio/mpeg7-2001:MediaTime/mpeg7-2001:MediaDuration" />
                                    </mpeg7-2005:MediaDuration>
                                </mpeg7-2005:MediaTime>
                            </mpeg7-2005:TitleAudio>
                        </tva-2007:PromotionalMedia>
                    </dvb:RelatedMaterial>
                    <dvb:Duration>
                        <xsl:value-of
                        select="esg:ESGMain/esg:ESG/esg:ContentTable/esg:Content/esg:Duration" />
                    </dvb:Duration>
                </dvb:BasicDescription>
                <dvb:AVAttributes>
                    <tva-2007:BitRate>
                        <xsl:attribute name="variable"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:AcquisitionTable/esg:Acquisition/esg:ComponentDescription/esg:Compon
entCharacteristic/esg:Bandwidth/@variable" />
                        </xsl:attribute>
                        <xsl:attribute name="minimum"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:AcquisitionTable/esg:Acquisition/esg:ComponentDescription/esg:Compon
entCharacteristic/esg:Bandwidth/@minimum" />
                        </xsl:attribute>
                        <xsl:attribute name="maximum"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:AcquisitionTable/esg:Acquisition/esg:ComponentDescription/esg:Compon
entCharacteristic/esg:Bandwidth/@maximum" />
                        </xsl:attribute>
                        <xsl:attribute name="average"><xsl:value-of
select="esg:ESGMain/esg:ESG/esg:AcquisitionTable/esg:Acquisition/esg:ComponentDescription/esg:Compon
entCharacteristic/esg:Bandwidth/@average" />
                        </xsl:attribute>
                    </tva-2007:BitRate>
                </dvb:AVAttributes>
            </dvbfile:ContentItemInformation>
            <dvbfile:BroadcastServiceName>
                <xsl:attribute name="lang"
                    namespace="http://www.w3.org/XML/1998/namespace"><xsl:value-of
            select="esg:ESGMain/esg:ESG/esg:ServiceTable/esg:Service/esg:ServiceName/@xml:lang" />
                </xsl:attribute>
                <xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ServiceTable/esg:Service/esg:ServiceName" />
```

```
            </dvbfile:BroadcastServiceName>
            <dvbfile:ScheduleEvent>
                <tva-2007:ProgramURL>
                    <xsl:value-of
        select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/esg:ContentLocation" />
                </tva-2007:ProgramURL><!--Do not use InstanceDesctription. The information is
conveyed in the ContentItemInformation instead.-->
                <tva-2007:PublishedStartTime>
                    <xsl:value-of
    select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/esg:PublishedStartTime" />
                </tva-2007:PublishedStartTime>
                <tva-2007:PublishedEndTime>
                    <xsl:value-of
    select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/esg:PublishedEndTime" />
                </tva-2007:PublishedEndTime>
                <tva-2007:Live>
                    <xsl:attribute name="value"><xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/@live" />
                    </xsl:attribute>
                </tva-2007:Live>
                <tva-2007:Repeat>
                    <xsl:attribute name="value"><xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/@repeat" />
                    </xsl:attribute>
                </tva-2007:Repeat>
                <tva-2007:Free>
                    <xsl:attribute name="value"><xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/@freeToAir" />
                        <xsl:value-of
                    select="esg:ESGMain/esg:ESG/esg:ScheduleEventTable/esg:ScheduleEvent/@clearToAir" />
                    </xsl:attribute>
                </tva-2007:Free>
            </dvbfile:ScheduleEvent>
        </dvbfile:FileContentItemInformation>

    </xsl:template>
</xsl:stylesheet>
```