INTERNATIONAL STANDARD

ISO/IEC 13818-1

Third edition
2007-10-15

Information technology — Generic coding of moving pictures and associated audio information: Systems

Technologies de l'information — Codage générique des images animées et des informations sonores associées: Systèmes

Reference number
ISO/IEC 13818-1:2007(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

CONTENTS

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 13818-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information,* in collaboration with ITU-T. The identical text is published as ITU-T Rec. H.222.0 (05/2006).

This third edition cancels and replaces the second edition (ISO/IEC 13818-1:2000), which has been technically revised. It also incorporates the Amendments ISO/IEC 13818-1:2000/Amd.1:2003, ISO/IEC 13818-1:2000/Amd.2:2004, ISO/IEC 13818-1:2000/Amd.3:2004, ISO/IEC 13818-1:2000/Amd.4:2005 and ISO/IEC 13818-1:2000/Amd.5:2005, and the Technical Corrigenda ISO/IEC 13818-1:2000/Cor.1:2002, ISO/IEC 13818-1:2000/Cor.2:2002, ISO/IEC 13818-1:2000/Cor.3:2005, ISO/IEC 13818-1:2000/Cor.4:2007.

ISO/IEC 13818 consists of the following parts, under the general title *Information technology — Generic coding of moving pictures and associated audio information*:

— *Part 1: Systems*

— *Part 2: Video*

— *Part 3: Audio*

— *Part 4: Conformance testing*

— *Part 5: Software simulation* [Technical Report]

— *Part 6: Extensions for DSM-CC*

— *Part 7: Advanced Audio Coding (AAC)*

— *Part 9: Extension for real time interface for systems decoders*

— *Part 10: Conformance extensions for Digital Storage Media Command and Control (DSM-CC)*

— *Part 11: IPMP on MPEG-2 systems*

Introduction

The systems part of this Recommendation | International Standard addresses the combining of one or more elementary streams of video and audio, as well as other data, into single or multiple streams which are suitable for storage or transmission. Systems coding follows the syntactical and semantic rules imposed by this Specification and provides information to enable synchronized decoding of decoder buffers over a wide range of retrieval or receipt conditions.

System coding shall be specified in two forms: the **Transport Stream** and the **Program Stream**. Each is optimized for a different set of applications. Both the Transport Stream and Program Stream defined in this Recommendation | International Standard provide coding syntax which is necessary and sufficient to synchronize the decoding and presentation of the video and audio information, while ensuring that data buffers in the decoders do not overflow or underflow. Information is coded in the syntax using time stamps concerning the decoding and presentation of coded audio and visual data and time stamps concerning the delivery of the data stream itself. Both stream definitions are packet-oriented multiplexes.

The basic multiplexing approach for single video and audio elementary streams is illustrated in Figure Intro. 1. The video and audio data is encoded as described in ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3. The resulting compressed elementary streams are packetized to produce **PES packets**. Information needed to use PES packets independently of either Transport Streams or Program Streams may be added when PES packets are formed. This information is not needed and need not be added when PES packets are further combined with system level information to form **Transport Streams** or **Program Streams**. This systems standard covers those processes to the right of the vertical dashed line.

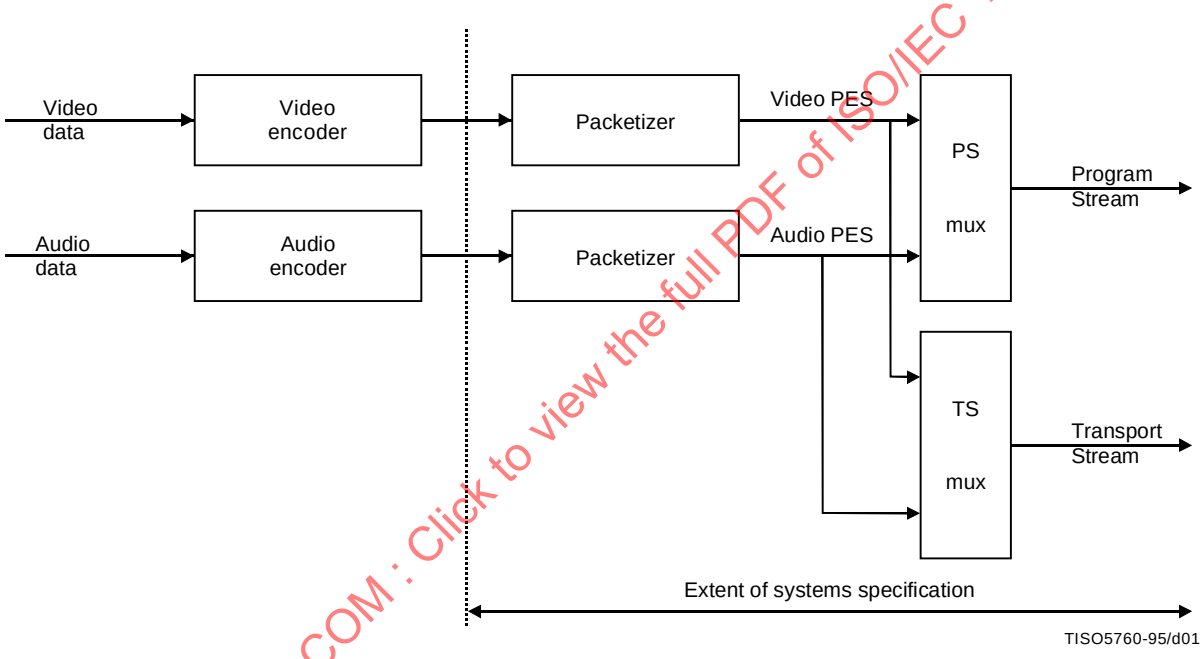

Figure Intro. 1 – Simplified overview of the scope of this Recommendation | International Standard

The **Program Stream** is analogous and similar to ISO/IEC 11172 Systems layer. It results from combining one or more streams of PES packets, which have a common time base, into a single stream.

For applications that require the elementary streams which comprise a single program to be in separate streams which are not multiplexed, the elementary streams can also be encoded as separate Program Streams, one per elementary stream, with a common time base. In this case the values encoded in the SCR fields of the various streams shall be consistent.

Like the single Program Stream, all elementary streams can be decoded with synchronization.

The Program Stream is designed for use in relatively error-free environments and is suitable for applications which may involve software processing of system information such as interactive multi-media applications. Program Stream packets may be of variable and relatively great length.

The **Transport Stream** combines one or more programs with one or more independent time bases into a single stream. PES packets made up of elementary streams that form a program share a common timebase. The Transport Stream is designed for use in environments where errors are likely, such as storage or transmission in lossy or noisy media. Transport Stream packets are 188 bytes in length.

Program and Transport Streams are designed for different applications and their definitions do not strictly follow a layered model. It is possible and reasonable to convert from one to the other; however, one is not a subset or superset of the other. In particular, extracting the contents of a program from a Transport Stream and creating a valid Program Stream is possible and is accomplished through the common interchange format of PES packets, but not all of the fields needed in a Program Stream are contained within the Transport Stream; some must be derived. The Transport Stream may be used to span a range of layers in a layered model, and is designed for efficiency and ease of implementation in high bandwidth applications.

The scope of syntactical and semantic rules set forth in the systems specification differ: the syntactical rules apply to systems layer coding only, and do not extend to the compression layer coding of the video and audio specifications; by contrast, the semantic rules apply to the combined stream in its entirety.

The systems specification does not specify the architecture or implementation of encoders or decoders, nor those of multiplexors or demultiplexors. However, bit stream properties do impose functional and performance requirements on encoders, decoders, multiplexors and demultiplexors. For instance, encoders must meet minimum clock tolerance requirements. Notwithstanding this and other requirements, a considerable degree of freedom exists in the design and implementation of encoders, decoders, multiplexors, and demultiplexors.

Intro. 1 Transport Stream

The Transport Stream is a stream definition which is tailored for communicating or storing one or more programs of coded data according to ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3 and other data in environments in which significant errors may occur. Such errors may be manifested as bit value errors or loss of packets.

Transport Streams may be either fixed or variable rate. In either case the constituent elementary streams may either be fixed or variable rate. The syntax and semantic constraints on the stream are identical in each of these cases. The Transport Stream rate is defined by the values and locations of Program Clock Reference (PCR) fields, which in general are separate PCR fields for each program.

There are some difficulties with constructing and delivering a Transport Stream containing multiple programs with independent time bases such that the overall bit rate is variable. Refer to 2.4.2.2.

The Transport Stream may be constructed by any method that results in a valid stream. It is possible to construct Transport Streams containing one or more programs from elementary coded data streams, from Program Streams, or from other Transport Streams which may themselves contain one or more programs.

The Transport Stream is designed in such a way that several operations on a Transport Stream are possible with minimum effort. Among these are:

1) Retrieve the coded data from one program within the Transport Stream, decode it and present the decoded results as shown in Figure Intro. 2.
2) Extract the Transport Stream packets from one program within the Transport Stream and produce as output a different Transport Stream with only that one program as shown in Figure Intro. 3.
3) Extract the Transport Stream packets of one or more programs from one or more Transport Streams and produce as output a different Transport Stream (not illustrated).
4) Extract the contents of one program from the Transport Stream and produce as output a Program Stream containing that one program as shown in Figure Intro. 4.
5) Take a Program Stream, convert it into a Transport Stream to carry it over a lossy environment, and then recover a valid, and in certain cases, identical Program Stream.

Figure Intro. 2 and Figure Intro. 3 illustrate prototypical demultiplexing and decoding systems which take as input a Transport Stream. Figure Intro. 2 illustrates the first case, where a Transport Stream is directly demultiplexed and decoded. Transport Streams are constructed in two layers:

- a system layer; and
- a compression layer.

The input stream to the Transport Stream decoder has a system layer wrapped about a compression layer. Input streams to the Video and Audio decoders have only the compression layer.

Operations performed by the prototypical decoder which accepts Transport Streams either apply to the entire Transport Stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The Transport Stream system layer is divided into two sub-layers, one for multiplex-wide operations (the Transport Stream packet layer), and one for stream-specific operations (the PES packet layer).

A prototypical decoder for Transport Streams, including audio and video, is also depicted in Figure Intro. 2 to illustrate the function of a decoder. The architecture is not unique – some system decoder functions, such as decoder timing

control, might equally well be distributed among elementary stream decoders and the channel-specific decoder – but this figure is useful for discussion. Likewise, indication of errors detected by the channel-specific decoder to the individual audio and video decoders may be performed in various ways and such communication paths are not shown in the diagram. The prototypical decoder design does not imply any normative requirement for the design of a Transport Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

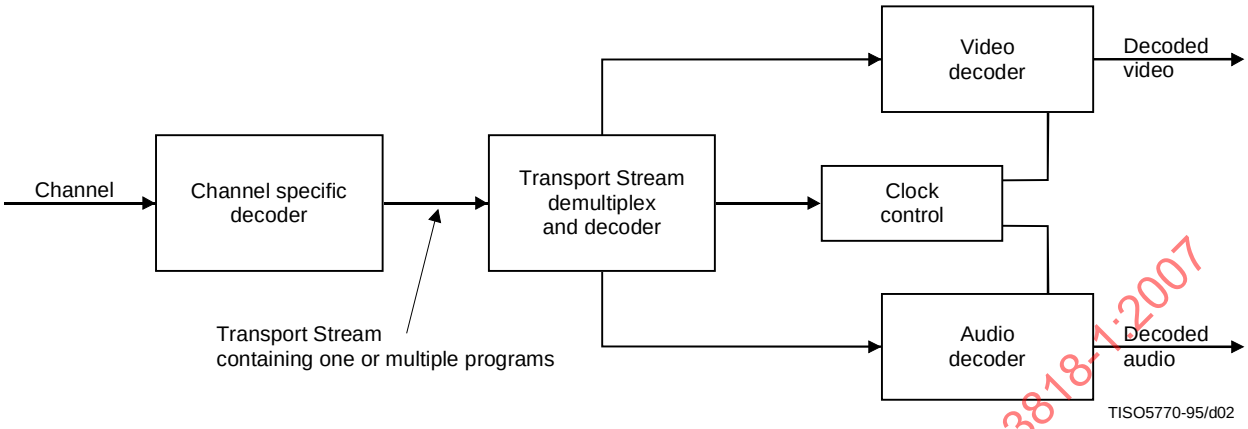

Figure Intro. 2 – Prototypical transport demultiplexing and decoding example

Figure Intro. 3 illustrates the second case, where a Transport Stream containing multiple programs is converted into a Transport Stream containing a single program. In this case the re-multiplexing operation may necessitate the correction of Program Clock Reference (PCR) values to account for changes in the PCR locations in the bit stream.

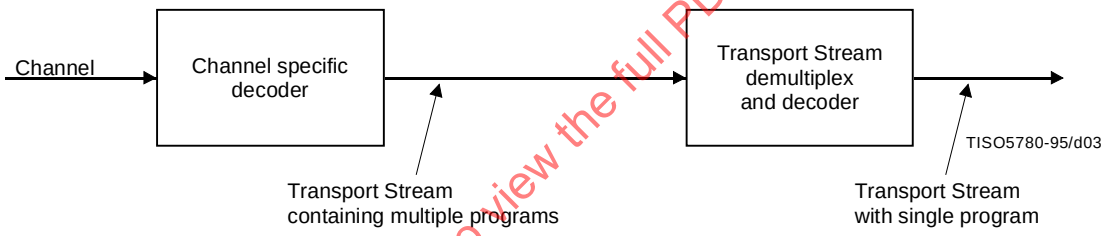

Figure Intro. 3 – Prototypical transport multiplexing example

Figure Intro. 4 illustrates a case in which a multi-program Transport Stream is first demultiplexed and then converted into a Program Stream.

Figures Intro. 3 and Intro. 4 indicate that it is possible and reasonable to convert between different types and configurations of Transport Streams. There are specific fields defined in the **Transport Stream** and **Program Stream** syntax which facilitate the conversions illustrated. There is no requirement that specific implementations of demultiplexors or decoders include all of these functions.

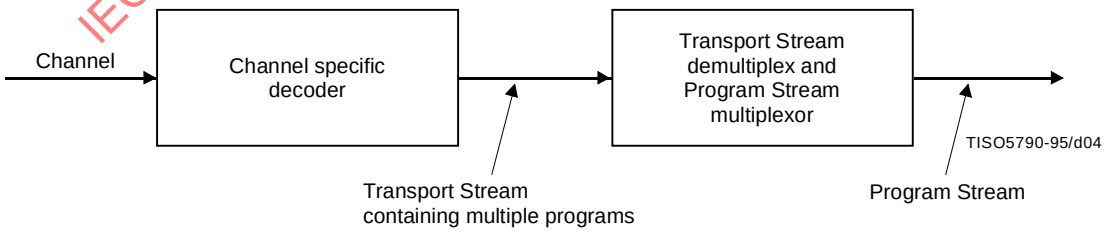

Figure Intro. 4 – Prototypical Transport Stream to Program Stream conversion

Intro. 2 Program Stream

The Program Stream is a stream definition which is tailored for communicating or storing one program of coded data and other data in environments where errors are very unlikely, and where processing of system coding, e.g., by software, is a major consideration.

Program Streams may be either fixed or variable rate. In either case, the constituent elementary streams may be either fixed or variable rate. The syntax and semantics constraints on the stream are identical in each case. The Program Stream rate is defined by the values and locations of the System Clock Reference (SCR) and mux_rate fields.

A prototypical audio/video Program Stream decoder system is depicted in Figure Intro. 5. The architecture is not unique – system decoder functions including decoder timing control might as equally well be distributed among elementary stream decoders and the channel-specific decoder – but this figure is useful for discussion. The prototypical decoder design does not imply any normative requirement for the design of an Program Stream decoder. Indeed non-audio/video data is also allowed, but not shown.

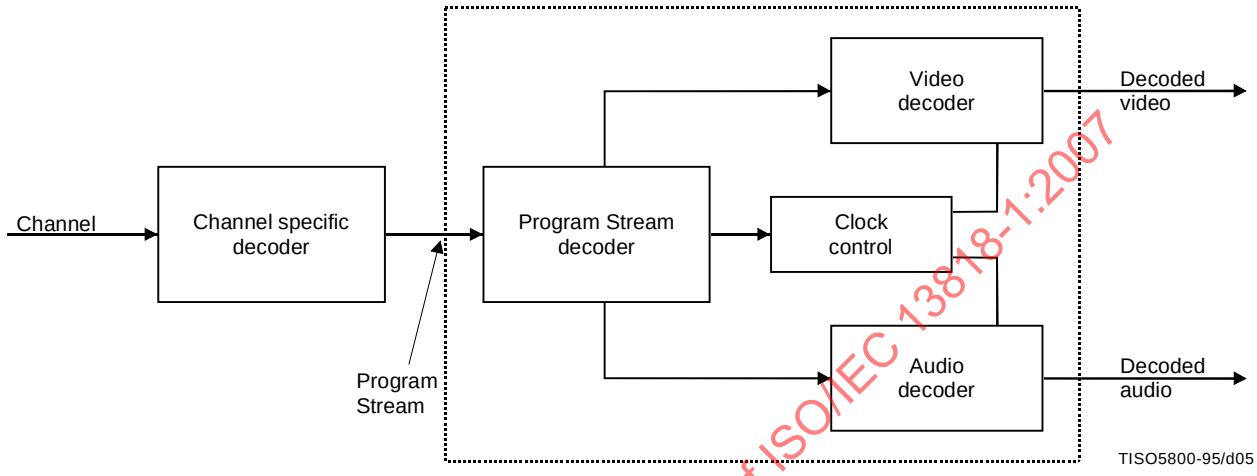

Figure Intro. 5 – Prototypical decoder for Program Streams

The prototypical decoder for Program Streams shown in Figure Intro. 5 is composed of System, Video and Audio decoders conforming to Parts 1, 2 and 3, respectively, of ISO/IEC 13818. In this decoder, the multiplexed coded representation of one or more audio and/or video streams is assumed to be stored or communicated on some channel in some channel-specific format. The channel-specific format is not governed by this Recommendation | International Standard, nor is the channel-specific decoding part of the prototypical decoder.

The prototypical decoder accepts as input a Program Stream and relies on a Program Stream Decoder to extract timing information from the stream. The Program Stream Decoder demultiplexes the stream, and the elementary streams so produced serve as inputs to Video and Audio decoders, whose outputs are decoded video and audio signals. Included in the design, but not shown in the figure, is the flow of timing information among the Program Stream decoder, the Video and Audio decoders, and the channel-specific decoder. The Video and Audio decoders are synchronized with each other and with the channel using this timing information.

Program Streams are constructed in two layers: a system layer and a compression layer. The input stream to the Program Stream Decoder has a system layer wrapped about a compression layer. Input streams to the Video and Audio decoders have only the compression layer.

Operations performed by the prototypical decoder either apply to the entire Program Stream ("multiplex-wide operations"), or to individual elementary streams ("stream-specific operations"). The Program Stream system layer is divided into two sub-layers, one for multiplex-wide operations (the pack layer), and one for stream-specific operations (the PES packet layer).

Intro. 3 Conversion between Transport Stream and Program Stream

It may be possible and reasonable to convert between **Transport Streams** and **Program Streams** by means of PES packets. This results from the specification of **Transport Stream** and **Program Stream** as embodied in 2.4.1 and 2.5.1 of the normative requirements of this Recommendation | International Standard. PES packets may, with some constraints, be mapped directly from the payload of one multiplexed bit stream into the payload of another multiplexed bit stream. It is possible to identify the correct order of PES packets in a program to assist with this if the program_packet_sequence_counter is present in all PES packets.

Certain other information necessary for conversion, e.g., the relationship between elementary streams, is available in tables and headers in both streams. Such data, if available, shall be correct in any stream before and after conversion.

Intro. 4 Packetized Elementary Stream

Transport Streams and **Program Streams** are each logically constructed from PES packets, as indicated in the syntax definitions in 2.4.3.6. PES packets shall be used to convert between Transport Streams and Program Streams; in some cases the PES packets need not be modified when performing such conversions. PES packets may be much larger than the size of a Transport Stream packet.

A continuous sequence of PES packets of one elementary stream with one stream ID may be used to construct a PES Stream. When PES packets are used to form a PES stream, they shall include Elementary Stream Clock Reference (ESCR) fields and Elementary Stream Rate (ES_Rate) fields, with constraints as defined in 2.4.3.8. The PES stream data shall be contiguous bytes from the elementary stream in their original order. PES streams do not contain some necessary system information which is contained in Program Streams and Transport Streams. Examples include the information in the Pack Header, System Header, Program Stream Map, Program Stream Directory, Program Map Table, and elements of the Transport Stream packet syntax.

The PES Stream is a logical construct that may be useful within implementations of this Recommendation | International Standard; however, it is not defined as a stream for interchange and interoperability. Applications requiring streams containing only one elementary stream can use Program Streams or Transport Streams which each contain only one elementary stream. These streams contain all of the necessary system information. Multiple Program Streams or Transport Streams, each containing a single elementary stream, can be constructed with a common time base and therefore carry a complete program, i.e., with audio and video.

Intro. 5 Timing model

Systems, Video and Audio all have a timing model in which the end-to-end delay from the signal input to an encoder to the signal output from a decoder is a constant. This delay is the sum of encoding, encoder buffering, multiplexing, communication or storage, demultiplexing, decoder buffering, decoding, and presentation delays. As part of this timing model all video pictures and audio samples are presented exactly once, unless specifically coded to the contrary, and the inter-picture interval and audio sample rate are the same at the decoder as at the encoder. The system stream coding contains timing information which can be used to implement systems which embody constant end-to-end delay. It is possible to implement decoders which do not follow this model exactly; however, in such cases it is the decoder's responsibility to perform in an acceptable manner. The timing is embodied in the normative specifications of this Recommendation | International Standard, which must be adhered to by all valid bit streams, regardless of the means of creating them.

All timing is defined in terms of a common system clock, referred to as a System Time Clock. In the Program Stream this clock may have an exactly specified ratio to the video or audio sample clocks, or it may have an operating frequency which differs slightly from the exact ratio while still providing precise end-to-end timing and clock recovery.

In the Transport Stream the system clock frequency is constrained to have the exactly specified ratio to the audio and video sample clocks at all times; the effect of this constraint is to simplify sample rate recovery in decoders.

Intro. 6 Conditional access

Encryption and scrambling for conditional access to programs encoded in the Program and Transport Streams is supported by the system data stream definitions. Conditional access mechanisms are not specified here. The stream definitions are designed so that implementation of practical conditional access systems is reasonable, and there are some syntactical elements specified which provide specific support for such systems.

Intro. 7 Multiplex-wide operations

Multiplex-wide operations include the coordination of data retrieval of the channel, the adjustment of clocks, and the management of buffers. The tasks are intimately related. If the rate of data delivery of the channel is controllable, then data delivery may be adjusted so that decoder buffers neither overflow nor underflow; but if the data rate is not controllable, then elementary stream decoders must slave their timing to the data received from the channel to avoid overflow or underflow.

Program Streams are composed of packs whose headers facilitate the above tasks. Pack headers specify intended times at which each byte is to enter the Program Stream Decoder from the channel, and this target arrival schedule serves as a reference for clock correction and buffer management. The schedule need not be followed exactly by decoders, but they must compensate for deviations about it.

Similarly, Transport Streams are composed of Transport Stream packets with headers containing information which specifies the times at which each byte is intended to enter a Transport Stream Decoder from the channel. This schedule provides exactly the same function as that which is specified in the Program Stream.

An additional multiplex-wide operation is a decoder's ability to establish what resources are required to decode a Transport Stream or Program Stream. The first pack of each Program Stream conveys parameters to assist decoders in

this task. Included, for example, are the stream's maximum data rate and the highest number of simultaneous video channels. The Transport Stream likewise contains globally useful information.

The Transport Stream and Program Stream each contain information which identifies the pertinent characteristics of, and relationships between, the elementary streams which constitute each program. Such information may include the language spoken in audio channels, as well as the relationship between video streams when multi-layer video coding is implemented.

Intro. 8 Individual stream operations (PES Packet Layer)

The principal stream-specific operations are:

1) demultiplexing; and
2) synchronizing playback of multiple elementary streams.

Intro. 8.1 Demultiplexing

On encoding, Program Streams are formed by multiplexing elementary streams, and Transport Streams are formed by multiplexing elementary streams, Program Streams, or the contents of other Transport Streams. Elementary streams may include private, reserved, and padding streams in addition to audio and video streams. The streams are temporally subdivided into packets, and the packets are serialized. A PES packet contains coded bytes from one and only one elementary stream.

In the Program Stream both fixed and variable packet lengths are allowed subject to constraints as specified in 2.5.1 and 2.5.2. For Transport Streams the packet length is 188 bytes. Both fixed and variable PES packet lengths are allowed, and will be relatively long in most applications.

On decoding, demultiplexing is required to reconstitute elementary streams from the multiplexed Program Stream or Transport Stream. Stream_id codes in Program Stream packet headers, and Packet ID codes in the Transport Stream make this possible.

Intro. 8.2 Synchronization

Synchronization among multiple elementary streams is accomplished with Presentation Time Stamps (PTS) in the Program Stream and Transport streams. Time stamps are generally in units of 90 kHz, but the System Clock Reference (SCR), the Program Clock Reference (PCR) and the optional Elementary Stream Clock Reference (ESCR) have extensions with a resolution of 27 MHz. Decoding of N-elementary streams is synchronized by adjusting the decoding of streams to a common master time base rather than by adjusting the decoding of one stream to match that of another. The master time base may be one of the N-decoders' clocks, the data source's clock, or it may be some external clock.

Each program in a Transport Stream, which may contain multiple programs, may have its own time base. The time bases of different programs within a Transport Stream may be different.

Because PTSs apply to the decoding of individual elementary streams, they reside in the PES packet layer of both the Transport Streams and Program Streams. End-to-end synchronization occurs when encoders save time stamps at capture time, when the time stamps propagate with associated coded data to decoders, and when decoders use those time stamps to schedule presentations.

Synchronization of a decoding system with a channel is achieved through the use of the SCR in the Program Stream and by its analogue, the PCR, in the Transport Stream. The SCR and PCR are time stamps encoding the timing of the bit stream itself, and are derived from the same time base used for the audio and video PTS values from the same program. Since each program may have its own time base, there are separate PCR fields for each program in a Transport Stream containing multiple programs. In some cases it may be possible for programs to share PCR fields. Refer to 2.4.4, Program Specific Information (PSI), for the method of identifying which PCR is associated with a program. A program shall have one and only one PCR time base associated with it.

Intro. 8.3 Relation to compression layer

The PES packet layer is independent of the compression layer in some senses, but not in all. It is independent in the sense that PES packet payloads need not start at compression layer start codes, as defined in Parts 2 and 3 of ISO/IEC 13818. For example, video start codes may occur anywhere within the payload of a PES packet, and start codes may be split by a PES packet header. However, time stamps encoded in PES packet headers apply to presentation times of compression layer constructs (namely, presentation units). In addition, when the elementary stream data conforms to ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 13818-3, the PES_packet_data_bytes shall be byte aligned to the bytes of this Recommendation | International Standard.

Intro. 9 System reference decoder

Part 1 of ISO/IEC 13818 employs a "System Target Decoder" (STD), one for Transport Streams (refer to 2.4.2) referred to as "Transport System Target Decoder" (T-STD) and one for Program Streams (refer to 2.5.2) referred to as "Program System Target Decoder" (P-STD), to provide a formalism for timing and buffering relationships. Because the STD is parameterized in terms of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 fields (for example, buffer sizes) each elementary stream leads to its own parameterization of the STD. Encoders shall produce bit streams that meet the appropriate STD's constraints. Physical decoders may assume that a stream plays properly on its STD. The physical decoder must compensate for ways in which its design differs from that of the STD.

Intro. 10 Applications

The streams defined in this Recommendation | International Standard are intended to be as useful as possible to a wide variety of applications. Application developers should select the most appropriate stream.

Modern data communications networks may be capable of supporting ITU-T Rec. H.222.0 | ISO/IEC 13818-1 video and ISO/IEC 13818 audio. A real-time transport protocol is required. The Program Stream may be suitable for transmission on such networks.

The Program Stream is also suitable for multimedia applications on CD-ROM. Software processing of the Program Stream may be appropriate.

The Transport Stream may be more suitable for error-prone environments, such as those used for distributing compressed bit-streams over long-distance networks and in broadcast systems.

Many applications require storage and retrieval of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams on various Digital Storage Media (DSM). A Digital Storage Media Command and Control (DSM-CC) protocol is specified in Annex B and Part 6 of ISO/IEC 13818 in order to facilitate the control of such media.

INTERNATIONAL STANDARD
ITU-T RECOMMENDATION

Information technology – Generic coding of moving pictures and associated audio information: Systems

SECTION 1 – GENERAL

1.1 Scope

This Recommendation | International Standard specifies the system layer of the coding. It was developed principally to support the combination of the video and audio coding methods defined in Parts 2 and 3 of ISO/IEC 13818. The system layer supports six basic functions:

1) the synchronization of multiple compressed streams on decoding;
2) the interleaving of multiple compressed streams into a single stream;
3) the initialization of buffering for decoding start up;
4) continuous buffer management;
5) time identification;
6) multiplexing and signalling of various components in a system stream.

An ITU-T Rec. H.222.0 | ISO/IEC 13818-1 multiplexed bit stream is either a **Transport Stream** or a **Program Stream**. Both streams are constructed from **PES packets** and packets containing other necessary information. Both stream types support multiplexing of video and audio compressed streams from one program with a common time base. The **Transport Stream** additionally supports the multiplexing of video and audio compressed streams from multiple programs with independent time bases. For almost error-free environments the **Program Stream** is generally more appropriate, supporting software processing of program information. The **Transport Stream** is more suitable for use in environments where errors are likely.

An ITU-T Rec. H.222.0 | ISO/IEC 13818-1 multiplexed bit stream, whether a Transport Stream or a Program Stream, is constructed in two layers: the outermost layer is the system layer, and the innermost is the compression layer. The system layer provides the functions necessary for using one or more compressed data streams in a system. The video and audio parts of this Specification define the compression coding layer for audio and video data. Coding of other types of data is not defined by this Specification, but is supported by the system layer provided that the other types of data adhere to the constraints defined in 2.7.

1.2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations.

1.2.1 Identical Recommendations | International Standards

– ITU-T Recommendation H.262 (2000) | ISO/IEC 13818-2:2000, *Information technology – Generic coding of moving pictures and associated audio information: Video*.

1.2.2 Paired Recommendations | International Standards equivalent in technical content

– ITU-T Recommendation H.264 (2005), *Advanced video coding for generic audiovisual services*.

ISO/IEC 14496-10:2005, *Information technology – Coding of audio-visual objects – Part 10: Advanced Video Coding*.

– ITU-T Recommendation T.171 (1996), *Protocols for interactive audiovisual services: coded representation of multimedia and hypermedia objects*.

ISO/IEC 13522-1:1997, *Information technology – Coding of multimedia and hypermedia information – Part 1: MHEG object representation – Base notation (ASN.1)*.

1.2.3 Additional references

– ISO 639-2:1998, *Codes for the representation of names of languages – Part 2: Alpha-3 code*.

– ISO/IEC 8859-1:1998, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*.

– ISO 15706:2002, *Information and documentation – International Standard Audiovisual Number (ISAN)*.

– ISO 15706-2:2007, *Information and documentation – International Standard Audiovisual Number (ISAN) – Part 2: Version identifier*.

– ISO/IEC 11172-1:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 1: Systems*.

– ISO/IEC 11172-2:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: Video*.

– ISO/IEC 11172-3:1993, *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*.

– ISO/IEC 13818-3:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 3: Audio*.

– ISO/IEC 13818-6:1998, *Information technology – Generic coding of moving pictures and associated audio information – Part 6: Extensions for DSM-CC*.

– ISO/IEC 13818-7:2006, *Information technology – Generic coding of moving pictures and associated audio information – Part 7: Advanced Audio Coding (AAC)*.

– ISO/IEC 13818-11:2004, *Information technology – Generic coding of moving pictures and associated audio information – Part 11: IPMP on MPEG-2 systems*.

– ISO/IEC 14496-1:2004, *Information technology – Coding of audio-visual objects – Part 1: Systems*.

– ISO/IEC 14496-2:2004, *Information technology – Coding of audio-visual objects – Part 2: Visual*.

– ISO/IEC 14496-3:2005, *Information technology – Coding of audio-visual objects – Part 3: Audio*.

– Recommendation ITU-R BT.601-6 (2007), *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16.9 aspect ratios*.

– Recommendation ITU-R BT.470-7 (2005), *Conventional analogue television systems*.

– Recommendation ITU-R BR.648, *Digital recording of audio signals*.

– ITU-T Recommendation J.17 (1988), *Pre-emphasis used on sound-programme circuits*.

– IEC Publication 60908:1999, *Audio recording – Compact disc digital audio system*.

SECTION 2 – TECHNICAL ELEMENTS

2.1 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply. If specific to a Part, this is parenthetically noted.

2.1.1 access unit (system): A coded representation of a presentation unit. In the case of audio, an access unit is the coded representation of an audio frame.

In the case of video, an access unit includes all the coded data for a picture, and any stuffing that follows it, up to but not including the start of the next access unit. If a picture is not preceded by a group_start_code or a sequence_header_code, the access unit begins with the picture start code. If a picture is preceded by a group_start_code and/or a sequence_header_code, the access unit begins with the first byte of the first of these start codes. If it is the last picture preceding a sequence_end_code in the bitstream, all bytes between the last byte of the coded picture and the sequence_end_code (including the sequence_end_code) belong to the access unit.

For the definition of an access unit for ITU-T Rec. H.264 | ISO/IEC 14496-10 video, see the AVC access unit definition in 2.1.3.

2.1.2 AVC 24-hour picture (system): An AVC access unit with a presentation time that is more than 24 hours in the future. For the purpose of this definition, AVC access unit n has a presentation time that is more than 24 hours in the future if the difference between the initial arrival time $t_{ai}(n)$ and the DPB output time $t_{o,dpb}(n)$ is more than 24 hours.

2.1.3 AVC access unit (system): An access unit as defined for byte streams in ITU-T Rec. H.264 | ISO/IEC 14496-10 with the constraints specified in 2.14.1.

2.1.4 AVC Slice (system): A byte_stream_nal_unit as defined in ITU-T Rec. H.264 | ISO/IEC 14496-10 with nal_unit_type values of 1 or 5, or a byte_stream_nal_unit data structure with nal_unit_type value of 2 and any associated byte_stream_nal_unit data structures with nal_unit_type equal to 3 and/or 4.

2.1.5 AVC still picture (system): An AVC still picture consists of an AVC access unit containing an IDR picture, preceded by SPS and PPS NAL units that carry sufficient information to correctly decode the IDR picture. Preceding an AVC still picture, there shall be another AVC still picture or an End of Sequence NAL unit terminating a preceding coded video sequence unless the AVC still picture is the very first access unit in the video stream.

2.1.6 AVC video sequence (system): Coded video sequence as defined in 3.30 of ITU-T Rec. H.264 | ISO/IEC 14496-10.

2.1.7 AVC video stream (system): An ITU-T Rec. H.264 | ISO/IEC 14496-10 stream. An AVC video stream consists of one or more AVC video sequences.

2.1.8 bitrate: The rate at which the compressed bit stream is delivered from the channel to the input of a decoder.

2.1.9 byte aligned: A bit in a coded bit stream is byte-aligned if its position is a multiple of 8-bits from the first bit in the stream.

2.1.10 channel: A digital medium that stores or transports an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream.

2.1.11 coded B-frame: A B-frame picture or a pair of B-field pictures.

2.1.12 coded frame: A coded frame is a coded I-frame, coded B-frame or a coded P-frame.

2.1.13 coded I-frame: An I-frame picture or a pair of field pictures where the first field picture is an I-picture and the second field picture is either an I-picture or a P-picture.

2.1.14 coded P-frame: A P-frame picture or a pair of P-field pictures.

2.1.15 coded representation: A data element as represented in its encoded form.

2.1.16 compression: Reduction in the number of bits used to represent an item of data.

2.1.17 constant bitrate: Operation where the bitrate is constant from start to finish of the compressed bit stream.

2.1.18 constrained system parameter stream; CSPS (system): A Program Stream for which the constraints defined in 2.7.9 apply.

2.1.19 Cyclic Redundancy Check (CRC): The CRC to verify the correctness of data.

2.1.20 data element: An item of data as represented before encoding and after decoding.

2.1.21 decoded stream: The decoded reconstruction of a compressed bit stream.

2.1.22 decoder: An embodiment of a decoding process.

2.1.23 decoding (process): The process defined in this Recommendation | International Standard that reads an input-coded bit stream and outputs decoded pictures or audio samples.

2.1.24 decoding time-stamp; DTS (system): A field that may be present in a PES packet header that indicates the time that an access unit is decoded in the system target decoder.

2.1.25 digital storage media (DSM): A digital storage or transmission device or system.

2.1.26 DSM-CC: Digital storage media command and control.

2.1.27 entitlement control message (ECM): Entitlement Control Messages are private conditional access information which specify control words and possibly other, typically stream-specific, scrambling and/or control parameters.

2.1.28 entitlement management message (EMM): Entitlement Management Messages are private conditional access information which specify the authorization levels or the services of specific decoders. They may be addressed to single decoders or groups of decoders.

2.1.29 editing: The process by which one or more compressed bit streams are manipulated to produce a new compressed bit stream. Edited bit streams meet the same requirements as streams which are not edited.

2.1.30 elementary stream; ES (system): A generic term for one of the coded video, coded audio or other coded bit streams in PES packets. One elementary stream is carried in a sequence of PES packets with one and only one stream_id.

2.1.31 Elementary Stream Clock Reference; ESCR (system): A time stamp in the PES Stream from which decoders of PES streams may derive timing.

2.1.32 encoder: An embodiment of an encoding process.

2.1.33 encoding (process): A process, not specified in this Recommendation | International Standard, that reads a stream of input pictures or audio samples and produces a coded bit stream conforming to this Recommendation.

2.1.34 entropy coding: Variable length lossless coding of the digital representation of a signal to reduce redundancy.

2.1.35 event: An event is defined as a collection of elementary streams with a common time base, an associated start time, and an associated end time.

2.1.36 fast forward playback (video): The process of displaying a sequence, or parts of a sequence, of pictures in display-order faster than real-time.

2.1.37 forbidden: The term "forbidden", when used in the clauses of this Recommendation | International Standard defining the coded bit stream, indicates that the value specified shall never be used.

2.1.38 metadata: Information to describe audiovisual content and data essence in a format defined by ISO or any other authority.

2.1.39 metadata access unit: A global structure within metadata that defines the fraction of metadata that is intended to be decoded at a specific instant in time. The internal structure of a metadata Access Unit is defined by the format of the metadata.

2.1.40 metadata application format: Identifies the format of the application that uses the metadata; signals application specific information for transport of metadata.

2.1.41 metadata decoder configuration information: Data needed by a receiver to decode a specific metadata service. Depending on the format of the metadata, decoder configuration information may or may not be needed.

2.1.42 metadata format: Identifies the coding format of metadata.

2.1.43 metadata service: Coherent set of metadata of the same format delivered to a receiver for a specific purpose.

2.1.44 metadata service id: Identifier of a specific metadata service; used for some transport methods of the metadata.

2.1.45 metadata stream: The concatenation or collection of metadata Access Units from one or more metadata services.

2.1.46 (multiplexed) stream (system): A bit stream composed of 0 or more elementary streams combined in a manner that conforms to this Recommendation | International Standard.

2.1.47 layer (video and systems): One of the levels in the data hierarchy of the video and system specifications defined in Parts 1 and 2 of this Recommendation | International Standard.

2.1.48 pack (system): A pack consists of a pack header followed by zero or more packets. It is a layer in the system coding syntax described in 2.5.3.3.

2.1.49 packet data (system): Contiguous bytes of data from an elementary stream present in a packet.

2.1.50 packet identifier; PID (system): A unique integer value used to identify elementary streams of a program in a single or multi-program Transport Stream as described in 2.4.3.

2.1.51 padding (audio): A method to adjust the average length of an audio frame in time to the duration of the corresponding PCM samples, by conditionally adding a slot to the audio frame.

2.1.52 payload: Payload refers to the bytes which follow the header bytes in a packet. For example, the payload of some Transport Stream packets includes a PES_packet_header and its PES_packet_data_bytes, or pointer_field and

PSI sections, or private data; but a PES_packet_payload consists of only PES_packet_data_bytes. The Transport Stream packet header and adaptation fields are not payload.

2.1.53 PES (system): An abbreviation for Packetized Elementary Stream.

2.1.54 PES packet (system): The data structure used to carry elementary stream data. A PES packet consists of a PES packet header followed by a number of contiguous bytes from an elementary data stream. It is a layer in the system coding syntax described in 2.4.3.6.

2.1.55 PES packet header (system): The leading fields in a PES packet up to and not including the PES_packet_data_byte fields, where the stream is not a padding stream. In the case of a padding stream the PES packet header is similarly defined as the leading fields in a PES packet up to and not including padding_byte fields.

2.1.56 PES Stream (system): A PES Stream consists of PES packets, all of whose payloads consist of data from a single elementary stream, and all of which have the same stream_id. Specific semantic constraints apply. Refer to Intro. 4.

2.1.57 presentation time-stamp; PTS (system): A field that may be present in a PES packet header that indicates the time that a presentation unit is presented in the system target decoder.

2.1.58 presentation unit; PU (system): A decoded Audio Access Unit or a decoded picture.

2.1.59 program (system): A program is a collection of program elements. Program elements may be elementary streams. Program elements need not have any defined time base; those that do, have a common time base and are intended for synchronized presentation.

2.1.60 Program Clock Reference; PCR (system): A time stamp in the Transport Stream from which decoder timing is derived.

2.1.61 program element (system): A generic term for one of the elementary streams or other data streams that may be included in a program.

2.1.62 Program Specific Information; PSI (system): PSI consists of normative data which is necessary for the demultiplexing of Transport Streams and the successful regeneration of programs and is described in 2.4.4. An example of privately defined PSI data is the non-mandatory network information table.

2.1.63 random access: The process of beginning to read and decode the coded bit stream at an arbitrary point.

2.1.64 reserved: The term "reserved", when used in the clauses defining the coded bit stream, indicates that the value may be used in the future for ISO defined extensions. Unless otherwise specified within this Recommendation | International Standard, all reserved bits shall be set to '1'.

2.1.65 scrambling (system): The alteration of the characteristics of a video, audio or coded data stream in order to prevent unauthorized reception of the information in a clear form. This alteration is a specified process under the control of a conditional access system.

2.1.66 source stream: A single non-multiplexed stream of samples before compression coding.

2.1.67 splicing (system): The concatenation, performed on the system level, of two different elementary streams. The resulting system stream conforms totally to this Recommendation | International Standard. The splice may result in discontinuities in timebase, continuity counter, PSI, and decoding.

2.1.68 start codes (system): 32-bit codes embedded in the coded bit stream. They are used for several purposes including identifying some of the layers in the coding syntax. Start codes consist of a 24-bit prefix (0x000001) and an 8-bit stream_id as shown in Table 2-22.

2.1.69 STD input buffer (system): A first-in first-out buffer at the input of a system target decoder for storage of compressed data from elementary streams before decoding.

2.1.70 still picture: A still picture consists of a video sequence, coded as defined in ITU-T Rec. H.262 | ISO/IEC 13818-2, ISO/IEC 11172-2 or ISO/IEC 14496-2, that contains exactly one coded picture which is intra-coded. This picture has an associated PTS and in case of coding according to ISO/IEC 11172-2, ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 14496-2, the presentation time of succeeding pictures, if any, is later than that of the still picture by at least two picture periods.

2.1.71 system header (system): The system header is a data structure defined in 2.5.3.5 that carries information summarizing the system characteristics of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Stream.

2.1.72 System Clock Reference; SCR (system): A time stamp in the Program Stream from which decoder timing is derived.

2.1.73 system target decoder; STD (system): A hypothetical reference model of a decoding process used to define the semantics of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 multiplexed bit stream.

2.1.74 time-stamp (system): A term that indicates the time of a specific action such as the arrival of a byte or the presentation of a Presentation Unit.

2.1.75 transport stream packet header (system): The leading fields in a Transport Stream packet, up to and including the continuity_counter field.

2.1.76 variable bitrate: An attribute of Transport Streams or Program Streams wherein the rate of arrival of bytes at the input to a decoder varies with time.

2.2 Symbols and abbreviations

The mathematical operators used to describe this Recommendation | International Standard are similar to those used in the C-programming language. However, integer division with truncation and rounding are specifically defined. The bitwise operators are defined assuming two's-complement representation of integers. Numbering and counting loops generally begin from 0.

2.2.1 Arithmetic operators

+	Addition
–	Subtraction (as a binary operator) or negation (as a unary operator)
++	Increment
– –	Decrement
* or ×	Multiplication
^	Power
/	Integer division with truncation of the result toward 0. For example, 7/4 and –7/–4 are truncated to 1 and –7/4 and 7/–4 are truncated to –1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from 0 unless otherwise specified. For example 3//2 is rounded to 2, and –3//2 is rounded to –2.
DIV	Integer division with truncation of the result towards – ∞.
%	Modulus operator. Defined only for positive numbers.
Sign( )	$\text{Sign}(x) = \begin{cases} 1 & x > 0 \\ 0 & x == 0 \\ -1 & x < 0 \end{cases}$
NINT( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from 0.
sin	Sine
cos	Cosine
exp	Exponential
√	Square root
$\log_{10}$	Logarithm to base ten
$\log_e$	Logarithm to base e

2.2.2 Logical operators

||	Logical OR
&&	Logical AND
!	Logical NOT

2.2.3 Relational operators

>	Greater than
≥	Greater than or equal to
<	Less than
≤	Less than or equal to

= =	Equal to
!=	Not equal to
max [,...,]	The maximum value in the argument list
min [,...,]	The minimum value in the argument list

2.2.4 Bitwise operators

&	AND
|	OR
>>	Shift right with sign extension
<<	Shift left with 0 fill

2.2.5 Assignment

=	Assignment operator

2.2.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in this Recommendation | International Standard. Bit strings are written as a string of 1s and 0s within single quote marks, e.g., '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.
ch	Channel
gr	Granule of 3 * 32 sub-band samples in audio Layer II, 18 * 32 sub-band samples in audio Layer III.
main_data	The main_data portion of the bit stream contains the scale factors, Huffman encoded data, and ancillary information.
main_data_beg	This gives the location in the bit stream of the beginning of the main_data for the frame. The location is equal to the ending location of the previous frame's main_data plus 1 bit. It is calculated from the main_data_end value of the previous frame.
part2_length	This value contains the number of main_data bits used for scale factors.
rpchof	Remainder polynomial coefficients, highest order first
sb	Sub-band
scfsi	Scalefactor selector information
switch_point_l	Number of scalefactor band (long block scalefactor band) from which point on window switching is used
switch_point_s	Number of scalefactor band (short block scalefactor band) from which point on window switching is used
tcimsbf	Two's complement integer, msb (sign) bit first
uimsbf	Unsigned integer, most significant bit first
vlclbf	Variable length code, left bit first, where "left" refers to the order in which the variable length codes are written
window	Number of actual time slot in case of block_type = = 2, $0 \leq$ window ≤ 2.

The byte order of multi-byte words is most significant byte first.

2.2.7 Constants

π	3.14159265359
e	2.71828182845

2.3 Method of describing bit stream syntax

The bit streams retrieved by the decoder are described in 2.4.1 and 2.5.1. Each data item in the bit stream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bit stream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and definition of the state variables used in their

decoding are described in the clauses containing the semantic description of the syntax. The following constructs are used to express the conditions when data elements are present, and are in normal type.

Note this syntax uses the "C"-code convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true:

while (condition) {
data_element
. . .
}
	If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

do {
data_element
. . .
}
while (condition)
	The data element always occurs at least once. The data element is repeated until the condition is not true.

if (condition) {
data_element
. . .
}
	If the condition is true, then the first group of data elements occurs next in the data stream.

else {
data_element
. . .
}
	If the condition is not true, then the second group of data elements occurs next in the data stream.

for (i = 0; i < n; i++) {
data_element
. . .
}
	The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to 1 for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} are omitted when only one data element follows:

data_element []	data_element [] is an array of data. The number of data elements is indicated by the context.
data_element [n]	data_element [n] is the n+1th element of an array of data.
data_element [m][n]	data_element [m][n] is the m+1,n+1th element of a two-dimensional array of data.
data_element [l][m][n]	data_element [l][m][n] is the l+1,m+1,n+1th element of a three-dimensional array of data.
data_element [m..n]	is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that either Figure 2-1 or Figure 2-2 implements a satisfactory decoding procedure. In particular, they define a correct and error-free input bitstream. Actual decoders must include a means to look for start codes and sync bytes (Transport Stream) in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardized.

2.4 Transport Stream bitstream requirements

2.4.1 Transport Stream coding structure and parameters

The ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Transport Stream coding layer allows one or more programs to be combined into a single stream. Data from each elementary stream are multiplexed together with information that allows synchronized presentation of the elementary streams within a program.

A Transport Stream consists of one or more programs. Audio and video elementary streams consist of access units.

Elementary Stream data is carried in PES packets. A PES packet consists of a PES packet header followed by packet data. PES packets are inserted into Transport Stream packets. The first byte of each PES packet header is located at the first available payload location of a Transport Stream packet.

The PES packet header begins with a 32-bit start-code that also identifies the stream or stream type to which the packet data belongs. The PES packet header may contain decoding and presentation time stamps (DTS and PTS). The PES packet header also contains other optional fields. The PES packet data field contains a variable number of contiguous bytes from one elementary stream.

Transport Stream packets begin with a 4-byte prefix, which contains a 13-bit Packet ID (PID), defined in Table 2-2. The PID identifies, via the Program Specific Information (PSI) tables, the contents of the data contained in the Transport Stream packet. Transport Stream packets of one PID value carry data of one and only one elementary stream.

The PSI tables are carried in the Transport Stream. There are Six PSI tables:

- Program Association Table;
- Program Map Table;
- Conditional Access Table;
- Network Information Table;
- Transport Stream Description Table;
- IPMP Control Information Table.

These tables contain the necessary and sufficient information to demultiplex and present programs. The Program Map Table, in Table 2-33 specifies, among other information, which PIDs, and therefore which elementary streams are associated to form each program. This table also indicates the PID of the Transport Stream packets which carry the PCR for each program. The Conditional Access Table shall be present if scrambling is employed. The Network Information Table is optional and its contents are not specified by this Recommendation | International Standard. The IPMP Control Information Table shall be present if IPMP as described in ISO/IEC 13818-11 is used by any of the components in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream.

Transport Stream packets may be null packets. Null packets are intended for padding of Transport Streams. They may be inserted or deleted by re-multiplexing processes and, therefore, the delivery of the payload of null packets to the decoder cannot be assumed.

This Recommendation | International Standard does not specify the coded data which may be used as part of conditional access systems. This Specification does, however, provide mechanisms for program service providers to transport and identify this data for decoder processing, and to reference correctly data which are specified by this Specification. This type of support is provided both through Transport Stream packet structures and in the conditional access table (refer to Table 2-32 of the PSI).

2.4.2 Transport Stream system target decoder

The semantics of the Transport Stream specified in 2.4.3 and the constraints on these semantics specified in 2.7 require exact definitions of byte arrival and decoding events and the times at which these occur. The definitions needed are set out in this Recommendation | International Standard using a hypothetical decoder known as the Transport Stream System Target Decoder (T-STD). Informative Annex D contains further explanation of the T-STD.

The T-STD is a conceptual model used to define these terms precisely and to model the decoding process during the construction or verification of Transport Streams. The T-STD is defined only for this purpose. There are three types of decoders in the T-STD: video, audio, and systems. Figure 2-1 illustrates an example. Neither the architecture of the T-STD nor the timing described precludes uninterrupted, synchronized play-back of Transport Streams from a variety of decoders with different architectures or timing schedules.

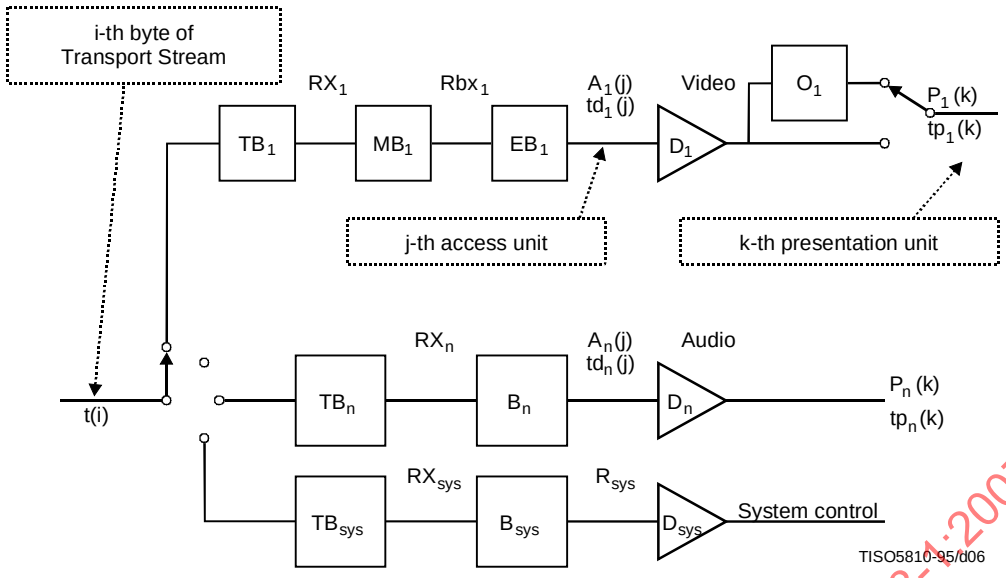

Figure 2-1 – Transport Stream system target decoder notation

The following notation is used to describe the Transport Stream system target decoder and is partially illustrated in Figure 2-1 above.

i, i′, i″	are indices to bytes in the Transport Stream. The first byte has index 0.
j	is an index to access units in the elementary streams.
k, k′, k″	are indices to presentation units in the elementary streams.
n	is an index to the elementary streams.
p	is an index to Transport Stream packets in the Transport Stream.
t(i)	indicates the time in seconds at which the i-th byte of the Transport Stream enters the system target decoder. The value t(0) is an arbitrary constant.
PCR(i)	is the time encoded in the PCR field measured in units of the period of the 27-MHz system clock where i is the byte index of the final byte of the program_clock_reference_base field.
$A_n(j)$	is the j-th access unit in elementary stream n. $A_n(j)$ is indexed in decoding order.
$td_n(j)$	is the decoding time, measured in seconds, in the system target decoder of the j-th access unit in elementary stream n.
$P_n(k)$	is the k-th presentation unit in elementary stream n. $P_n(k)$ results from decoding $A_n(j)$. $P_n(k)$ is indexed in presentation order.
$tp_n(k)$	is the presentation time, measured in seconds, in the system target decoder of the k-th presentation unit in elementary stream n.
t	is time measured in seconds.
$F_n(t)$	is the fullness, measured in bytes, of the system target decoder input buffer for elementary stream n at time t.
B_n	is the main buffer for elementary stream n. It is present only for audio elementary streams.
BS_n	is the size of buffer, B_n, measured in bytes.
B_{sys}	is the main buffer in the system target decoder for system information for the program that is in the process of being decoded.
BS_{sys}	is the size of B_{sys}, measured in bytes.
MB_n	is the multiplexing buffer, for elementary stream n. It is present only for video elementary streams.
MBS_n	is the size of MB_n, measured in bytes.
EB_n	is the elementary stream buffer for elementary stream n. It is present only for video elementary streams.
EBS_n	is the size of the elementary stream buffer EB_n, measured in bytes.

TB_{sys} is the transport buffer for system information for the program that is in the process of being decoded.

TBS_{sys} is the size of TB_{sys}, measured in bytes.

TB_n is the transport buffer for elementary stream n.

TBS_n is the size of TB_n, measured in bytes.

D_{sys} is the decoder for system information in Program Stream n.

D_n is the decoder for elementary stream n.

O_n is the re-order buffer for video elementary stream n.

R_{sys} is the rate at which data are removed from B_{sys}.

Rx_n is the rate at which data are removed from TB_n.

Rbx_n is the rate at which PES packet payload data are removed from MB_n when the leak method is used. Defined only for video elementary streams.

$Rbx_n(j)$ is the rate at which PES packet payload data are removed from MB_n when the vbv_delay method is used. Defined only for video elementary streams.

Rx_{sys} is the rate at which data are removed from TB_{sys}.

R_{es} is the video elementary stream rate coded in a sequence header.

2.4.2.1 System clock frequency

Timing information referenced in the T-STD is carried by several data fields defined in this Specification. Refer to 2.4.3.4 and 2.4.3.6. In PCR fields this information is coded as the sampled value of a program's system clock. The PCR fields are carried in the adaptation field of the Transport Stream packets with a PID value equal to the PCR_PID defined in the TS_program_map_section of the program being decoded.

Practical decoders may reconstruct this clock from these values and their respective arrival times. The following are minimum constraints which apply to the program's system clock frequency as represented by the values of the PCR fields when they are received by a decoder.

The value of the system clock frequency is measured in Hz and shall meet the following constraints:

$$27\,000\,000 - 810 \leq \text{system\_clock\_frequency} \leq 27\,000\,000 + 810$$

$$\text{rate of change of system\_clock\_frequency with time} \leq 75 \times 10^{-3}\ \text{Hz/s}$$

NOTE – Sources of coded data should follow a tighter tolerance in order to facilitate compliant operation of consumer recorders and playback equipment.

A program's system_clock_frequency may be more accurate than required. Such improved accuracy may be transmitted to the decoder via the System clock descriptor described in 2.6.20.

Bit rates defined in this Specification are measured in terms of system_clock_frequency. For example, a bit rate of 27 000 000 bits per second in the T-STD would indicate that one byte of data is transferred every eight (8) cycles of the system clock.

The notation "system_clock_frequency" is used in several places in this Specification to refer to the frequency of a clock meeting these requirements. For notational convenience, equations in which PCR, PTS, or DTS appear, lead to values of time which are accurate to some integral multiple of $(300 \times 2^{33}/\text{system\_clock\_frequency})$ seconds. This is due to the encoding of PCR timing information as 33 bits of 1/300 of the system clock frequency plus 9 bits for the remainder, and encoding as 33 bits of the system clock frequency divided by 300 for PTS and DTS.

2.4.2.2 Input to the Transport Stream system target decoder

Input to the Transport Stream System Target Decoder (T-STD) is a Transport Stream. A Transport Stream may contain multiple programs with independent time bases. However, the T-STD decodes only one program at a time. In the T-STD model all timing indications refer to the time base of that program.

Data from the Transport Stream enters the T-STD at a piecewise constant rate. The time t(i) at which the i-th byte enters the T-STD is defined by decoding the program clock reference (PCR) fields in the input stream, encoded in the Transport Stream packet adaptation field of the program to be decoded and by counting the bytes in the complete Transport Stream between successive PCRs of that program. The PCR field (see equation 2-1) is encoded in two parts: one, in units of the period of 1/300 times the system clock frequency, called program_clock_reference_base

(see equation 2-2), and one in units of the system clock frequency called program_clock_reference_extension (see equation 2-3). The values encoded in these are computed by PCR_base(i) (see equation 2-2) and PCR_ext(i) (see equation 2-3) respectively. The value encoded in the PCR field indicates the time t(i), where i is the index of the byte containing the last bit of the program_clock_reference_base field.

Specifically:

$$PCR(i) = PCR\_base(i) \times 300 + PCR\_ext(i) \tag{2-1}$$

where:

$$PCR\_base(i) = ((system\_clock\_frequency \times t(i))\ DIV\ 300)\ \%\ 2^{33} \tag{2-2}$$

$$PCR\_ext(i) = ((system\_clock\_frequency \times t(i))\ DIV\ 1)\ \%\ 300 \tag{2-3}$$

For all other bytes the input arrival time, t(i) shown in equation 2-4 below, is computed from PCR(i″) and the transport rate at which data arrive, where the transport rate is determined as the number of bytes in the Transport Stream between the bytes containing the last bit of two successive program_clock_reference_base fields of the same program divided by the difference between the time values encoded in these same two PCR fields.

$$t(i) = \frac{PCR(i'')}{system\_clock\_frequency} + \frac{i - i''}{transport\_rate(i)} \tag{2-4}$$

where:

i is the index of any byte in the Transport Stream for i″ < i < i′.

i″ is the index of the byte containing the last bit of the most recent program_clock_reference_base field applicable to the program being decoded.

PCR(i″) is the time encoded in the program clock reference base and extension fields in units of the system clock.

The transport rate is given by:

$$transport\_rate(i) = \frac{((i - i'') \times system\_clock\_frequency)}{PCR(i') - PCR(i'')} \tag{2-5}$$

where:

i′ is the index of the byte containing the last bit of the immediately following program_clock_reference_base field applicable to the program being decoded.

NOTE – i″ < i ≤ i′.

In the case of a timebase discontinuity, indicated by the discontinuity_indicator in the transport packet adaptation field, the definition given in equation 2-4 and equation 2-5 for the time of arrival of bytes at the input to the T-STD is not applicable between the last PCR of the old timebase and the first PCR of the new timebase. In this case the time of arrival of these bytes is determined according to equation 2-4 with the modification that the transport rate used is that applicable between the last and next to last PCR of the old timebase.

A tolerance is specified for the PCR values. The PCR tolerance is defined as the maximum inaccuracy allowed in received PCRs. This inaccuracy may be due to imprecision in the PCR values or to PCR modification during re-multiplexing. It does not include errors in packet arrival time due to network jitter or other causes. The PCR tolerance is ± 500 ns.

In the T-STD model, the inaccuracy will be reflected as an inaccuracy in the calculated transport rate using equation 2-5.

Transport Streams with multiple programs and variable rate

Transport Streams may contain multiple programs which have independent time bases. Separate sets of PCRs, as indicated by the respective PCR_PID values, are required for each such independent program, and therefore the PCRs

cannot be co-located. The Transport Stream rate is piecewise constant for the program entering the T-STD. Therefore, if the Transport Stream rate is variable it can only vary at the PCRs of the program under consideration. Since the PCRs, and therefore the points in the transport Stream where the rate varies, are not co-located, the rate at which the Transport Stream enters the T-STD would have to differ depending on which program is entering the T-STD. Therefore, it is not possible to construct a consistent T-STD delivery schedule for an entire Transport Stream when that Transport Stream contains multiple programs with independent time bases and the rate of the Transport Stream is variable. It is straightforward, however, to construct constant bit rate Transport Streams with multiple variable rate programs.

2.4.2.3 Buffering

Complete Transport Stream packets containing system information, for the program selected for decoding, enter the system transport buffer, TB_{sys}, at the Transport Stream rate. These include Transport Stream packets whose PID values are 0, 1, 2 or 3, and all Transport Stream packets identified via the Program Association Table (see Table 2-30) as having the program_map_PID value for the selected program. Network Information Table (NIT) data as specified by the NIT PID is not transferred to TB_{sys}.

NOTE 1 – Size of IPMP Control Information table could be large, and the repetition rate of this table should be adjusted to meet the buffer requirement.

All bytes that enter the buffer TB_n are removed at the rate Rx_n specified below. Bytes which are part of the PES packet or its contents are delivered to the main buffer B_n for audio elementary streams and system data, and to the multiplexing buffer MB_n for video elementary streams. Other bytes are not, and may be used to control the system. Duplicate Transport Stream packets are not delivered to B_n, MB_n, or B_{sys}.

The buffer TB_n is emptied as follows:

– When there is no data in TB_n, Rx_n is equal to zero.

– Otherwise for video:

$$Rx_n = 1,\ 2 \times R_{max}[profile, level]$$

where:

R_{max}[profile, level] is specified according to the profile and level which can be found in Table 8-13 of ITU-T Rec. H.262 | ISO/IEC 13818-2. This Table specifies the upper bound of the rate of each elementary video stream within a specific profile and level.

Rx_n is equal to 1, 2 × R_{max} for ISO/IEC 11172-2 constrained parameter video streams, where R_{max} refers to the maximum bitrate for a Constrained Parameters bitstream in ISO/IEC 11172-2.

For ISO/IEC 13818-7 ADTS audio:

Number of Channels	Rx_n [bit/s]
1-2	2 000 000
3-8	5 529 600
9-12	8 294 400
13-48	33 177 600

Channels: The number of full-bandwidth audio output channels plus the number of independently switched coupling channel elements within the same elementary audio stream. For example, in the typical case that there are no independently switched coupling channel elements, mono is 1 channel, stereo is 2 channels and 5.1 channel surround is 5 channels (the LFE channel is not counted).

For other audio,

$$Rx_n = 2 \times 10^6 \text{ bits per second}$$

For systems data:

$$Rx_n = 1 \times 10^6 \text{ bits per second}$$

Rx_n is measured with respect to the system clock frequency.

Complete Transport Stream packets containing system information, for the program selected for decoding, enter the system transport buffer, TB_{sys}, at the Transport Stream rate. These include Transport Stream packets whose PID values are 0, 1, 2 and 3 (if present), and all Transport Stream packets identified via the Program Association Table (see Table 2-30) as having the program_map_PID value for the selected program. Network Information Table (NIT) data as specified by the NIT PID is not transferred to TB_{sys}.

Bytes are removed from TB_{sys} at the rate Rx_{sys} and delivered to B_{sys}. Each byte is transferred instantaneously.

Duplicate Transport Stream packets are not delivered to B_{sys}.

Transport packets which do not enter any TB_n or TB_{sys} are discarded.

The transport buffer size is fixed at 512 bytes.

The elementary stream buffer sizes EBS_1 through EBS_n are defined for video as equal to the vbv_buffer_size as it is carried in the sequence header. Refer to Summary of Constrained Parameters in ISO/IEC 11172-2 and Table 8-14 of ITU-T Rec. H.262 | ISO/IEC 13818-2.

The multiplexing buffer size MBS_1 through MBS_n are defined for video as follows:

For Low and Main level:

$$MBS_n = BS_{mux} + BS_{oh} + VBV_{\max}[profile, level] - vbv\_buffer\_size$$

where BS_{oh}, PES packet overhead buffering is defined as:

$$BS_{oh} = (1/750)\,\text{seconds} \times R_{\max}[profile, level]$$

and BS_{mux}, additional multiplex buffering is defined as:

$$BS_{mux} = 0.004\ \text{seconds} \times R_{\max}[profile, level]$$

and where VBV_{max}[profile, level] is defined in Table 8-14 of ITU-T Rec. H.262 | ISO/IEC 13818-2 and R_{max}[profile, level] is defined in Table 8-13 of ITU-T Rec. H.262 | ISO/IEC 13818-2, and vbv buffer size is carried in the sequence header described in 6.2.2 of ITU-T Rec. H.262 | ISO/IEC 13818-2.

For High 1440 and High level:

$$MBS_n = BS_{mux} + BS_{oh}$$

where BS_{oh} is defined as:

$$BS_{oh} = (1/750)\ \text{seconds} \times R_{\max}[profile, level]$$

and BS_{mux} is defined as:

$$BS_{mux} = 0.004\,\text{seconds} \times R_{\max}[profile, level]$$

and where R_{max}[profile, level] is defined in Table 8-13 of ITU-T Rec. H.262 | ISO/IEC 13818-2.

For Constrained Parameters ISO/IEC 11172-2 bitstreams:

$$MBS_n = BS_{mux} + BS_{oh} + vbv\_\max - vbv\_buffer\_size$$

where BS_{oh} is defined as:

$$BS_{oh} = (1/750)\,\text{seconds} \times R_{\max}$$

and BS_{mux} is defined as:

$$BS_{mux} = 0.004\,\text{seconds} \times R_{\max}$$

and where R_{max} and vbv_max refer to the maximum bitrate and the maximum vbv_buffer_size for a Constrained Parameters bitstream in ISO/IEC 11172-2 respectively.

A portion BS_{mux} = 4 ms × R_{max}[profile, level] of the MBS_n is allocated for buffering to allow multiplexing. The remainder is available for BS_{oh} and may also be available for initial multiplexing.

NOTE 2 – Buffer occupancy by PES packet overhead is directly bounded in PES streams by the PES-STD which is defined in 2.5.2.4. It is possible, but not necessary, to utilize PES streams to construct Transport Streams.

Buffer BS_n

The main buffer sizes BS_1 through BS_n are defined as follows.

Audio

For ISO/IEC 13818-7 ADTS audio:

Number of Channels	BS_n [bytes]
1-2	3 584
3-8	8 976
9-12	12 804
13-48	51 216

Channels: The number of full-bandwidth audio output channels plus the number of independently switched coupling channel elements within the same elementary audio stream. For example, in the typical case that there are no independently switched coupling channel elements, mono is 1 channel, stereo is 2 channels and 5.1 channel surround is 5 channels (the LFE channel is not counted).

For other audio:

$$BS_n = BS_{mux} + BS_{dec} + BS_{oh} = 3584\ bytes$$

The size of the access unit decoding buffer BS_{dec}, and the PES packet overhead buffer BS_{oh} are constrained by:

$$BS_{dec} + BS_{oh} \le 2848\ bytes$$

A portion (736 bytes) of the 3584 byte buffer is allocated for buffering to allow multiplexing. The rest, 2848 bytes, are shared for access unit buffering BS_{dec}, BS_{oh} and additional multiplexing.

Systems

The main buffer B_{sys} for system data is of size BS_{sys} = 1536 bytes.

Video

For video elementary streams, data is transferred from MB_n to EB_n using one of two methods: the leak method or the VBV delay method.

Leak method

The leak method transfers data from MB_n to EB_n using a leak rate R_{bx}. The leak method is used whenever any of the following is true:

- the STD descriptor (refer to 2.6.32) for the elementary stream is not present in the Transport Stream;
- the STD descriptor is present and the leak_valid flag has a value of '1';
- the STD descriptor is present, the leak_valid has a value of '0', and the vbv_delay fields coded in the video stream have the value 0xFFFF; or
- trick mode status is true (refer to 2.4.3.7).

For Low and Main level:

$$Rbx_n = R_{\max}[profile, level]$$

For High-1440 and High level:

$$Rbx_n = Min\{1.05 \times R_{es}, R_{\max}[profile, level]\}$$

For Constrained Parameters bitstream in ISO/IEC 11172-2:

$$Rbx_n = 1,\ 2 \times R_{\max}$$

where R_{max} is the maximum bit rate for a Constrained Parameters bitstream in ISO/IEC 11172-2.

If there is PES packet payload data in MB_n, and buffer EB_n is not full, the PES packet payload is transferred from MB_n to EB_n at a rate equal to Rbx_n. If EB_n is full, data are not removed from MB_n. When a byte of data is transferred from MB_n to EB_n, all PES packet header bytes that are in MB_n and immediately precede that byte, are instantaneously removed and discarded. When there is no PES packet payload data present in MB_n, no data is removed from MB_n. All data that enters MB_n leaves it. All PES packet payload data bytes enter EB_n instantaneously upon leaving MB_n.

Vbv_delay method

The vbv_delay method specifies precisely the time at which each byte of coded video data is transferred from MB_n to EB_n, using the vbv_delay values coded in the video elementary stream. The vbv_delay method is used whenever the STD descriptor (refer to 2.6.32) for this elementary stream is present in the Transport Stream, the leak_valid flag in the descriptor has the value '0', and vbv_delay fields coded in the video stream are not equal to 0xFFFF. If any vbv_delay values in a video sequence are not equal to 0xFFFF, none of the vbv_delay fields in that sequence shall be equal to 0xFFFF (refer to ISO/IEC 11172-2 and ITU-T Rec. H.262 | ISO/IEC 13818-2).

When the vbv_delay method is used, the final byte of the video picture start code for picture j is transferred from MB_n to the EB_n at the time $td_n(j) - vbv\_delay(j)$, where $td_n(j)$ is the decoding time of picture j, as defined above, and vbv_delay(j) is the delay time, in seconds, indicated by the vbv_delay field of picture j. The transfer of bytes between the final bytes of successive picture start codes (including the final byte of the second start code), into the buffer EB_n, is at a piecewise constant rate, $R_{bx}(j)$, which is specified for each picture j. Specifically, the rate, $R_{bx}(j)$, of transfer into this buffer is given by:

$$R_{bx}(j) = NB(j)/(vbv\_delay(j) - vbv\_delay(j+1) + td_n(j+1) - td_n(j)) \qquad (2\text{-}6)$$

where NB(j) is the number of bytes between the final bytes of the picture start codes (including the final byte of the second start code) of pictures j and j + 1, excluding PES packet header bytes.

NOTE 3 – vbv_delay(j + 1) and $td_n(j + 1)$ may have values that differ from those normally expected for periodic video display if the low_delay flag in the video sequence extension is set to '1'. It may not be possible to determine the correct values by examination of the bit stream.

The $R_{bx}(j)$ derived from equation 2-6 shall be less than or equal to R_{max}[profile, level] for elementary streams of stream type 0x02 (refer to Table 2-34), where R_{max}[profile, level] is defined in ITU-T Rec. H.262 | ISO/IEC 13818-2, and shall be less than or equal to the maximum bit rate allowed for constrained parameter video elementary streams of stream type 0x01, refer to ISO/IEC 11172-2.

When a byte of data is transferred from MB_n to EB_n, all PES packet header bytes that are in MB_n and immediately precede that byte are instantaneously removed and discarded. All data that enters MB_n leaves it. All PES packet payload data bytes enter EB_n instantaneously upon leaving MB_n.

Removal of access units

For each elementary stream buffer EB_n and main buffer B_n all data for the access unit that has been in the buffer longest, $A_n(j)$, and any stuffing bytes that immediately precede it that are present in the buffer at the time $td_n(j)$ are removed instantaneously at time $td_n(j)$. The decoding time $td_n(j)$ is specified in the DTS or PTS fields (refer to 2.4.3.6). Decoding times $td_n(j + 1)$, $td_n(j + 2)$, ... of access units without encoded DTS or PTS fields which directly follow access unit j may be derived from information in the elementary stream. Refer to Annex C of ITU-T Rec. H.262 | ISO/IEC 13818-2, ISO/IEC 13818-3, or ISO/IEC 11172. Also refer to 2.7.5. In the case of audio, all PES packet headers that are stored immediately before the access unit or that are embedded within the data of the access unit are removed

simultaneously with the removal of the access unit. As the access unit is removed it is instantaneously decoded to a presentation unit.

System data

In the case of system data, data is removed from the main buffer B_{sys} at a rate of R_{sys} whenever there is at least 1 byte available in buffer B_{sys}.

$$R_{sys} = \max(80\,000 \text{ bits/s}, \; transport\_rate(i) \times 8 \; bits/byte / 500) \qquad (2\text{-}7)$$

NOTE 4 – The intention of increasing R_{sys} in the case of high transport rates is to allow an increased data rate for the Program Specific Information.

Low delay

When the low_delay flag in the video sequence extension is set to '1' (see 6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2) the EB_n buffer may underflow. In this case, when the T-STD elementary stream buffer EB_n is examined at the time specified by $td_n(j)$, the complete data for the access unit may not be present in the buffer EB_n. When this case arises, the buffer shall be re-examined at intervals of two field-periods until the data for the complete access unit is present in the buffer. At this time the entire access unit shall be removed from buffer EB_n instantaneously. Overflow of buffer EB_n shall not occur.

When the low_delay_mode flag is set to '1', EB_n underflow is allowed to occur continuously without limit. The T-STD decoder shall remove access unit data from buffer EB_n at the earliest time consistent with the paragraph above and any DTS or PTS values encoded in the bit stream. Note that the decoder may be unable to re-establish correct decoding and display times as indicated by DTS and PTS until the EB_n buffer underflow situation ceases and a PTS or DTS is found in the bit stream.

Trick mode

When the DSM_trick_mode flag (2.4.3.6) is set to '1' in the PES Packet header of a packet containing the start of a B-type video access unit and the trick_mode_control field is set to '001' (slow motion) or '010' (freeze frame), or '100' (slow reverse) the B-picture access unit is not removed from the video data buffer EB_n until the last time of possibly multiple times that any field of the picture is decoded and presented. Repetition of the presentation of fields and pictures is defined in 2.4.3.8 under slow motion, slow reverse, and field_id_cntrl. The access unit is removed instantaneously from EB_n at the indicated time, which is dependent on the value of rep_cntrl.

When the DSM_trick_mode flag is set to '1' in the PES packet header of a packet containing the first byte of a picture start code, trick_mode status becomes true when that picture start code in the PES packet is removed from buffer EB_n Trick mode status remains true until a PES packet header is received by the T-STD in which the DSM_trick_mode flag is set to '0' and the first byte of the picture start code after that PES packet header is removed from buffer EB_n. When trick mode status is true, the buffer EB_n may underflow. All other constraints from normal streams are retained when trick mode status is true.

2.4.2.4 Decoding

Elementary streams buffered in B_1 through B_n and EB_1 through EB_n are decoded instantaneously by decoders D_1 through D_n and may be delayed in re-order buffers O_1 through O_n before being presented at the output of the T-STD. Re-order buffers are used only in the case of a video elementary stream when some access units are not carried in presentation order. These access units will need to be re-ordered before presentation. In particular, if $P_n(k)$ is an I-picture or a P-picture carried before one or more B-pictures, then it must be delayed in the re-order buffer, O_n, of the T-STD before being presented. Any picture previously stored in O_n is presented before the current picture can be stored. $P_n(k)$ should be delayed until the next I-picture or P-picture is decoded. While it is stored in the re-order buffer, the subsequent B-pictures are decoded and presented.

The time at which a presentation unit $P_n(k)$ is presented is $tp_n(k)$. For presentation units that do not require re-ordering delay, $tp_n(k)$ is equal to $td_n(j)$ since the access units are decoded instantaneously; this is the case, for example, for B-frames. For presentation units that are delayed, $tp_n(k)$ and $td_n(j)$ differ by the time that $P_n(k)$ is delayed in the re-order buffer, which is a multiple of the nominal picture period. Care should be taken to use adequate re-ordering delay from the beginning of video elementary streams to meet the requirements of the entire stream. For example, a stream which initially has only I- and P-pictures but later includes B-pictures should include re-ordering delay starting at the beginning of the stream.

ITU-T Rec. H.262 | ISO/IEC 13818-2 explains re-ordering of video pictures in greater detail.

2.4.2.5 Presentation

The function of a decoding system is to reconstruct presentation units from compressed data and to present them in a synchronized sequence at the correct presentation times. Although real audio and visual presentation devices generally have finite and different delays and may have additional delays imposed by post-processing or output functions, the system target decoder models these delays as zero.

In the T-STD in Figure 2-1 the display of a video presentation unit (a picture) occurs instantaneously at its presentation time, $tp_n(k)$.

In the T-STD the output of an audio presentation unit starts at its presentation time, $tp_n(k)$, when the decoder instantaneously presents the first sample. Subsequent samples in the presentation unit are presented in sequence at the audio sampling rate.

2.4.2.6 Buffer management

Transport Streams shall be constructed so that conditions defined in this subclause are satisfied. This subclause makes use of the notation defined for the System Target Decoder.

TB_n and TB_{sys} shall not overflow. TB_n and TB_{sys} shall empty at least once every second. B_n shall not overflow nor underflow. B_{sys} shall not overflow.

EB_n shall not underflow except when the low delay flag in the video sequence extension is set to '1' (refer to 6.2.2.3 in ITU-T Rec. H.262 | ISO/IEC 13818-2) or trick_mode status is true.

When the leak method for specifying transfers is in effect, MB_n shall not overflow, and shall empty at least once every second. EB_n shall not overflow.

When the vbv_delay method for specifying transfers is in effect, MB_n shall not overflow nor underflow, and EB_n shall not overflow.

The delay of any data through the System Target Decoder buffers shall be less than or equal to one second except for still picture video data and ISO/IEC 14496 streams. Specifically: $td_n(j) - t(i) \leq 1$ second for all j, and all bytes i in access unit $A_n(j)$.

For still picture video data, the delay is constrained by $td_n(j) - t(i) \leq 60$ seconds for all j, and all bytes i in access unit $A_n(j)$.

For ISO/IEC 14496 streams, the delay is constrained by $td_n(j) - t(i) \leq 10$ seconds for all j, and all bytes i in access unit $A_n(j)$.

Definition of overflow and underflow

Let $F_n(t)$ be the instantaneous fullness of T-STD buffer B_n.

$F_n(t) = 0$ instantaneously before $t = t(0)$

Overflow does not occur if:

$$F_n(t) \leq BS_n$$

for all t and n.

Underflow does not occur if:

$$0 \leq F_n(t)$$

for all t and n.

2.4.2.7 T-STD extensions for carriage of ISO/IEC 14496 data

For decoding of ISO/IEC 14496 data carried in a Transport Stream the T-STD model is extended. T-STD parameters for decoding of individual ISO/IEC 14496 elementary streams are defined in 2.11.2, while 2.11.3 defines T-STD extensions and parameters for decoding of ISO/IEC 14496 scenes and associated streams.

2.4.2.8 T-STD extensions for carriage of ITU-T Rec. H.264 | ISO/IEC 14496-10 video

To define the decoding in the T-STD of ITU-T Rec. H.264 | ISO/IEC 14496-10 video streams carried in a Transport Stream, the T-STD model needs to be extended. The T-STD extension and T-STD parameters for decoding of ITU-T Rec. H.264 | ISO/IEC 14496-10 video streams are defined in 2.14.3.1.

2.4.3 Specification of the Transport Stream syntax and semantics

The following syntax describes a stream of bytes. Transport Stream packets shall be 188 bytes long.

2.4.3.1 Transport Stream

See Table 2-1.

Table 2-1 – Transport Stream

Syntax	No. of bits	Mnemonic
MPEG_transport_stream() {		
do {		
transport_packet()		
} while (nextbits() = = sync_byte)		
}		

2.4.3.2 Transport Stream packet layer

See Table 2-2.

Table 2-2 – Transport packet of this Recommendation | International Standard

Syntax	No. of bits	Mnemonic
transport_packet(){		
sync_byte	**8**	**bslbf**
transport_error_indicator	**1**	**bslbf**
payload_unit_start_indicator	**1**	**bslbf**
transport_priority	**1**	**bslbf**
PID	**13**	**uimsbf**
transport_scrambling_control	**2**	**bslbf**
adaptation_field_control	**2**	**bslbf**
continuity_counter	**4**	**uimsbf**
if(adaptation_field_control = = '10' || adaptation_field_control = = '11'){		
adaptation_field()		
}		
if(adaptation_field_control = = '01' || adaptation_field_control = = '11') {		
for (i = 0; i < N; i++){		
data_byte	**8**	**bslbf**
}		
}		
}		

2.4.3.3 Semantic definition of fields in Transport Stream packet layer

sync_byte – The sync_byte is a fixed 8-bit field whose value is '0100 0111' (0x47). Sync_byte emulation in the choice of values for other regularly occurring fields, such as PID, should be avoided.

transport_error_indicator – The transport_error_indicator is a 1-bit flag. When set to '1' it indicates that at least 1 uncorrectable bit error exists in the associated Transport Stream packet. This bit may be set to '1' by entities external to the transport layer. When set to '1' this bit shall not be reset to '0' unless the bit value(s) in error have been corrected.

payload_unit_start_indicator – The payload_unit_start_indicator is a 1-bit flag which has normative meaning for Transport Stream packets that carry PES packets (refer to 2.4.3.6) or PSI data (refer to 2.4.4).

When the payload of the Transport Stream packet contains PES packet data, the payload_unit_start_indicator has the following significance: a '1' indicates that the payload of this Transport Stream packet will commence with the first byte of a PES packet and a '0' indicates no PES packet shall start in this Transport Stream packet. If the payload_unit_start_indicator is set to '1', then one and only one PES packet starts in this Transport Stream packet. This also applies to private streams of stream_type 6 (refer to Table 2-34).

When the payload of the Transport Stream packet contains PSI data, the payload_unit_start_indicator has the following significance: if the Transport Stream packet carries the first byte of a PSI section, the payload_unit_start_indicator value shall be '1', indicating that the first byte of the payload of this Transport Stream packet carries the pointer_field. If the Transport Stream packet does not carry the first byte of a PSI section, the payload_unit_start_indicator value shall be '0', indicating that there is no pointer_field in the payload. Refer to 2.4.4.1 and 2.4.4.2. This also applies to private streams of stream_type 5 (refer to Table 2-34).

For null packets the payload_unit_start_indicator shall be set to '0'.

The meaning of this bit for Transport Stream packets carrying only private data is not defined in this Specification.

transport_priority – The transport_priority is a 1-bit indicator. When set to '1' it indicates that the associated packet is of greater priority than other packets having the same PID which do not have the bit set to '1'. The transport mechanism can use this to prioritize its data within an elementary stream. Depending on the application the transport_priority field may be coded regardless of the PID or within one PID only. This field may be changed by channel-specific encoders or decoders.

PID – The PID is a 13-bit field, indicating the type of the data stored in the packet payload. PID value 0x0000 is reserved for the Program Association Table (see Table 2-30). PID value 0x0001 is reserved for the Conditional Access Table (see Table 2-32). PID value 0x0002 is reserved for Transport Stream Description Table (see Table 2-36), PID value 0x0003 is reserved for IPMP Control Information Table (see ISO/IEC 13818-11) and PID values 0x0004-0x000F are reserved. PID value 0x1FFF is reserved for null packets (see Table 2-3).

Table 2-3 – PID table

Value	Description
0x0000	Program Association Table
0x0001	Conditional Access Table
0x0002	Transport Stream Description Table
0x0003	IPMP Control Information Table
0x0004-0x000F	Reserved
0x0010 … 0x1FFE	May be assigned as network_PID, Program_map_PID, elementary_PID, or for other purposes
0x1FFF	Null packet
NOTE – The transport packets with PID values 0x0000, 0x0001, and 0x0010-0x1FFE are allowed to carry a PCR.	

transport_scrambling_control – This 2-bit field indicates the scrambling mode of the Transport Stream packet payload. The Transport Stream packet header, and the adaptation field when present, shall not be scrambled. In the case of a null packet the value of the transport_scrambling_control field shall be set to '00' (see Table 2-4).

Table 2-4 – Scrambling control values

Value	Description
00	Not scrambled
01	User-defined
10	User-defined
11	User-defined

adaptation_field_control – This 2-bit field indicates whether this Transport Stream packet header is followed by an adaptation field and/or payload (see Table 2-5).

Table 2-5 – Adaptation field control values

Value	Description
00	Reserved for future use by ISO/IEC
01	No adaptation_field, payload only
10	Adaptation_field only, no payload
11	Adaptation_field followed by payload

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoders shall discard Transport Stream packets with the adaptation_field_control field set to a value of '00'. In the case of a null packet the value of the adaptation_field_control shall be set to '01'.

continuity_counter – The continuity_counter is a 4-bit field incrementing with each Transport Stream packet with the same PID. The continuity_counter wraps around to 0 after its maximum value. The continuity_counter shall not be incremented when the adaptation_field_control of the packet equals '00' or '10'.

In Transport Streams, duplicate packets may be sent as two, and only two, consecutive Transport Stream packets of the same PID. The duplicate packets shall have the same continuity_counter value as the original packet and the adaptation_field_control field shall be equal to '01' or '11'. In duplicate packets each byte of the original packet shall be duplicated, with the exception that in the program clock reference fields, if present, a valid value shall be encoded.

The continuity_counter in a particular Transport Stream packet is continuous when it differs by a positive value of one from the continuity_counter value in the previous Transport Stream packet of the same PID, or when either of the non-incrementing conditions (adaptation_field_control set to '00' or '10', or duplicate packets as described above) are met. The continuity counter may be discontinuous when the discontinuity_indicator is set to '1' (refer to 2.4.3.4). In the case of a null packet the value of the continuity_counter is undefined.

data_byte – Data bytes shall be contiguous bytes of data from the PES packets (refer to 2.4.3.6), PSI sections (refer to 2.4.4), packet stuffing bytes after PSI sections, or private data not in these structures as indicated by the PID. In the case of null packets with PID value 0x1FFF, data_bytes may be assigned any value. The number of data_bytes, N, is specified by 184 minus the number of bytes in the adaptation_field(), as described in 2.4.3.4 below.

2.4.3.4 Adaptation field

See Table 2-6.

Table 2-6 – Transport Stream adaptation field

Syntax	No. of bits	Mnemonic
adaptation_field() {		
adaptation_field_length	**8**	**uimsbf**
if (adaptation_field_length > 0) {		
discontinuity_indicator	**1**	**bslbf**
random_access_indicator	**1**	**bslbf**
elementary_stream_priority_indicator	**1**	**bslbf**
PCR_flag	**1**	**bslbf**
OPCR_flag	**1**	**bslbf**
splicing_point_flag	**1**	**bslbf**
transport_private_data_flag	**1**	**bslbf**
adaptation_field_extension_flag	**1**	**bslbf**
if (PCR_flag = = '1') {		
program_clock_reference_base	**33**	**uimsbf**
Reserved	**6**	**bslbf**
program_clock_reference_extension	**9**	**uimsbf**
}		
if (OPCR_flag = = '1') {		
original_program_clock_reference_base	**33**	**uimsbf**
Reserved	**6**	**bslbf**
original_program_clock_reference_extension	**9**	**uimsbf**
}		
if (splicing_point_flag = = '1') {		
splice_countdown	**8**	**tcimsbf**
}		
if (transport_private_data_flag = = '1') {		
transport_private_data_length	**8**	**uimsbf**
for (i = 0; i < transport_private_data_length; i++) {		
private_data_byte	**8**	**bslbf**
}		
}		
if (adaptation_field_extension_flag = = '1') {		
adaptation_field_extension_length	**8**	**uimsbf**
ltw_flag	**1**	**bslbf**
piecewise_rate_flag	**1**	**bslbf**
seamless_splice_flag	**1**	**bslbf**
Reserved	**5**	**bslbf**
if (ltw_flag = = '1') {		
ltw_valid_flag	**1**	**bslbf**
ltw_offset	**15**	**uimsbf**
}		
if (piecewise_rate_flag = = '1') {		
reserved	**2**	**bslbf**
piecewise_rate	**22**	**uimsbf**
}		
if (seamless_splice_flag = = '1') {		
Splice_type	**4**	**bslbf**
DTS_next_AU[32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
DTS_next_AU[29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
DTS_next_AU[14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
for (i = 0; i < N; i++) {		
reserved	**8**	**bslbf**
}		
}		
for (i = 0; i < N; i++) {		
stuffing_byte	**8**	**bslbf**
}		
}		
}		

2.4.3.5 Semantic definition of fields in adaptation field

adaptation_field_length – The adaptation_field_length is an 8-bit field specifying the number of bytes in the adaptation_field immediately following the adaptation_field_length. The value '0' is for inserting a single stuffing byte in a Transport Stream packet. When the adaptation_field_control value is '11', the value of the adaptation_field_length shall be in the range 0 to 182. When the adaptation_field_control value is '10', the value of the adaptation_field_length shall be 183. For Transport Stream packets carrying PES packets, stuffing is needed when there is insufficient

PES packet data to completely fill the Transport Stream packet payload bytes. Stuffing is accomplished by defining an adaptation field longer than the sum of the lengths of the data elements in it, so that the payload bytes remaining after the adaptation field exactly accommodates the available PES packet data. The extra space in the adaptation field is filled with stuffing bytes.

This is the only method of stuffing allowed for Transport Stream packets carrying PES packets. For Transport Stream packets carrying PSI, an alternative stuffing method is described in 2.4.4.

discontinuity_indicator – This is a 1-bit field which when set to '1' indicates that the discontinuity state is true for the current Transport Stream packet. When the discontinuity_indicator is set to '0' or is not present, the discontinuity state is false. The discontinuity indicator is used to indicate two types of discontinuities, system time-base discontinuities and continuity_counter discontinuities.

A system time-base discontinuity is indicated by the use of the discontinuity_indicator in Transport Stream packets of a PID designated as a PCR_PID (refer to 2.4.4.9). When the discontinuity state is true for a Transport Stream packet of a PID designated as a PCR_PID, the next PCR in a Transport Stream packet with that same PID represents a sample of a new system time clock for the associated program. The system time-base discontinuity point is defined to be the instant in time when the first byte of a packet containing a PCR of a new system time-base arrives at the input of the T-STD. The discontinuity_indicator shall be set to '1' in the packet in which the system time-base discontinuity occurs. The discontinuity_indicator bit may also be set to '1' in Transport Stream packets of the same PCR_PID prior to the packet which contains the new system time-base PCR. In this case, once the discontinuity_indicator has been set to '1', it shall continue to be set to '1' in all Transport Stream packets of the same PCR_PID up to and including the Transport Stream packet which contains the first PCR of the new system time-base. After the occurrence of a system time-base discontinuity, no fewer than two PCRs for the new system time-base shall be received before another system time-base discontinuity can occur. Further, except when trick mode status is true, data from no more than two system time-bases shall be present in the set of T-STD buffers for one program at any time.

Prior to the occurrence of a system time-base discontinuity, the first byte of a Transport Stream packet which contains a PTS or DTS which refers to the new system time-base shall not arrive at the input of the T-STD. After the occurrence of a system time-base discontinuity, the first byte of a Transport Stream packet which contains a PTS or DTS which refers to the previous system time-base shall not arrive at the input of the T-STD.

A continuity_counter discontinuity is indicated by the use of the discontinuity_indicator in any Transport Stream packet. When the discontinuity state is true in any Transport Stream packet of a PID not designated as a PCR_PID, the continuity_counter in that packet may be discontinuous with respect to the previous Transport Stream packet of the same PID. When the discontinuity state is true in a Transport Stream packet of a PID that is designated as a PCR_PID, the continuity_counter may only be discontinuous in the packet in which a system time-base discontinuity occurs. A continuity counter discontinuity point occurs when the discontinuity state is true in a Transport Stream packet and the continuity_counter in the same packet is discontinuous with respect to the previous Transport Stream packet of the same PID. A continuity counter discontinuity point shall occur at most one time from the initiation of the discontinuity state until the conclusion of the discontinuity state. Furthermore, for all PIDs that are not designated as PCR_PIDs, when the discontinuity_indicator is set to '1' in a packet of a specific PID, the discontinuity_indicator may be set to '1' in the next Transport Stream packet of that same PID, but shall not be set to '1' in three consecutive Transport Stream packet of that same PID.

For the purpose of this clause, an elementary stream access point is defined as follows:

- ISO/IEC 11172-2 video and ITU-T Rec. H.262 | ISO/IEC 13818-2 video – The first byte of a video sequence header.
- ISO/IEC 14496-2 visual – The first byte of the visual object sequence header.
- ITU-T Rec. H.264 | ISO/IEC 14496-10 video – The first byte of an AVC access unit. The SPS and PPS parameter sets referenced in this and all subsequent AVC access units in the coded video stream shall be provided after this access point in the byte stream and prior to their activation.
- Audio – The first byte of an audio frame.

After a continuity counter discontinuity in a Transport packet which is designated as containing elementary stream data, the first byte of elementary stream data in a Transport Stream packet of the same PID shall be the first byte of an elementary stream access point. In the case of ISO/IEC 11172-2, or ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 14496-2 video, the first byte of an elementary stream access point may also be the first byte of a sequence_end_code followed by an elementary stream access point.

Each Transport Stream packet which contains elementary stream data with a PID not designated as a PCR_PID, and in which a continuity counter discontinuity point occurs, and in which a PTS or DTS occurs, shall arrive at the input of the T-STD after the system time-base discontinuity for the associated program occurs. In the case where the discontinuity state is true, if two consecutive Transport Stream packets of the same PID occur which have the same continuity_counter value and have adaptation_field_control values set to '01' or '11', the second packet may be

discarded. A Transport Stream shall not be constructed in such a way that discarding such a packet will cause the loss of PES packet payload data or PSI data.

After the occurrence of a discontinuity_indicator set to '1' in a Transport Stream packet which contains PSI information, a single discontinuity in the version_number of PSI sections may occur. At the occurrence of such a discontinuity, a version of the TS_program_map_sections of the appropriate program shall be sent with section_length = = 13 and the current_next_indicator = = 1, such that there are no program_descriptors and no elementary streams described. This shall then be followed by a version of the TS_program_map_section for each affected program with the version_number incremented by one and the current_next_indicator = = 1, containing a complete program definition. This indicates a version change in PSI data.

random_access_indicator – The random_access_indicator is a 1-bit field that indicates that the current Transport Stream packet, and possibly subsequent Transport Stream packets with the same PID, contain some information to aid random access at this point.

Specifically, when the bit is set to '1', the next PES packet to start in the payload of Transport Stream packets with the current PID shall contain an elementary stream access point as defined in the semantics for the discontinuity_indicator field. In addition, in the case of video, a presentation timestamp shall be present for the first picture following the elementary stream access point.

In the case of audio, the presentation timestamp shall be present in the PES packet containing the first byte of the audio frame. In the PCR_PID the random_access_indicator may only be set to '1' in Transport Stream packet containing the PCR fields.

elementary_stream_priority_indicator – The elementary_stream_priority_indicator is a 1-bit field. It indicates, among packets with the same PID, the priority of the elementary stream data carried within the payload of this Transport Stream packet. A '1' indicates that the payload has a higher priority than the payloads of other Transport Stream packets.

In the case of ISO/IEC 11172-2 or ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 14496-2 video, this field may be set to '1' only if the payload contains one or more bytes from an intra-coded slice.

In the case of ITU-T Rec. H.264 | ISO/IEC 14496-10 video, this field may be set to '1' only if the payload contains one or more bytes from a slice with slice_type set to 2, 4, 7, or 9.

A value of '0' indicates that the payload has the same priority as all other packets which do not have this bit set to '1'.

PCR_flag – The PCR_flag is a 1-bit flag. A value of '1' indicates that the adaptation_field contains a PCR field coded in two parts. A value of '0' indicates that the adaptation field does not contain any PCR field.

OPCR_flag – The OPCR_flag is a 1-bit flag. A value of '1' indicates that the adaptation_field contains an OPCR field coded in two parts. A value of '0' indicates that the adaptation field does not contain any OPCR field.

splicing_point_flag – The splicing_point_flag is a 1-bit flag. When set to '1', it indicates that a splice_countdown field shall be present in the associated adaptation field, specifying the occurrence of a splicing point. A value of '0' indicates that a splice_countdown field is not present in the adaptation field.

transport_private_data_flag – The transport_private_data_flag is a 1-bit flag. A value of '1' indicates that the adaptation field contains one or more private_data bytes. A value of '0' indicates the adaptation field does not contain any private_data bytes.

adaptation_field_extension_flag – The adaptation_field_extension_flag is a 1-bit field which when set to '1' indicates the presence of an adaptation field extension. A value of '0' indicates that an adaptation field extension is not present in the adaptation field.

program_clock_reference_base; program_clock_reference_extension – The program_clock_reference (PCR) is a 42-bit field coded in two parts. The first part, program_clock_reference_base, is a 33-bit field whose value is given by PCR_base(i), as given in equation 2-2. The second part, program_clock_reference_extension, is a 9-bit field whose value is given by PCR_ext(i), as given in equation 2-3. The PCR indicates the intended time of arrival of the byte containing the last bit of the program_clock_reference_base at the input of the system target decoder.

original_program_clock_reference_base; original_program_clock_reference_extension – The optional original program reference (OPCR) is a 42-bit field coded in two parts. These two parts, the base and the extension, are coded identically to the two corresponding parts of the PCR field. The presence of the OPCR is indicated by the OPCR_flag. The OPCR field shall be coded only in Transport Stream packets in which the PCR field is present. OPCRs are permitted in both single program and multiple program Transport Streams.

OPCR assists in the reconstruction of a single program Transport Stream from another Transport Stream. When reconstructing the original single program Transport Stream, the OPCR may be copied to the PCR field. The resulting

PCR value is valid only if the original single program Transport Stream is reconstructed exactly in its entirety. This would include at least any PSI and private data packets which were present in the original Transport Stream and would possibly require other private arrangements. It also means that the OPCR must be an identical copy of its associated PCR in the original single program Transport Stream.

The OPCR is expressed as follows:

$$OPCR(i) = OPCR\_base(i) \times 300 + OPCR\_ext(i) \tag{2-8}$$

where:

$$OPCR\_base(i) = ((system\_clock\_frequency \times t(i)) DIV 300) \% 2^{33} \tag{2-9}$$

$$OPCR\_ext(i) = ((system\_clock\_frequency \times t(i)) DIV 1) \% 300 \tag{2-10}$$

The OPCR field is ignored by the decoder. The OPCR field shall not be modified by any multiplexor or decoder.

splice_countdown – The splice_countdown is an 8-bit field, representing a value which may be positive or negative. A positive value specifies the remaining number of Transport Stream packets, of the same PID, following the associated Transport Stream packet until a splicing point is reached. Duplicate Transport Stream packets and Transport Stream packets which only contain adaptation fields are excluded. The splicing point is located immediately after the last byte of the Transport Stream packet in which the associated splice_countdown field reaches zero. In the Transport Stream packet where the splice_countdown reaches zero, the last data byte of the Transport Stream packet payload shall be the last byte of a coded audio frame or a coded picture. In the case of video, the corresponding access unit may or may not be terminated by a sequence_end_code. Transport Stream packets with the same PID, which follow, may contain data from a different elementary stream of the same type.

The payload of the next Transport Stream packet of the same PID (duplicate packets and packets without payload being excluded) shall commence with the first byte of a PES packet. In the case of audio, the PES packet payload shall commence with an access point. In the case of video, the PES packet payload shall commence with an access point, or with a sequence_end_code, followed by an access point. Thus, the previous coded audio frame or coded picture aligns with the packet boundary, or is padded to make this so. Subsequent to the splicing point, the countdown field may also be present. When the splice_countdown is a negative number whose value is minus n (–n), it indicates that the associated Transport Stream packet is the n-th packet following the splicing point (duplicate packets and packets without payload being excluded).

For the definition of an elementary stream access point, see the semantics of discontinuity_indicator.

transport_private_data_length – The transport_private_data_length is an 8-bit field specifying the number of private_data bytes immediately following the transport private_data_length field. The number of private_data bytes shall not be such that private data extends beyond the adaptation field.

private_data_byte – The private_data_byte is an 8-bit field that shall not be specified by ITU-T | ISO/IEC.

adaptation_field_extension_length – The adaptation_field_extension_length is an 8-bit field. It indicates the number of bytes of the extended adaptation field data immediately following this field, including reserved bytes if present.

ltw_flag (legal time window_flag) – This is a 1-bit field which when set to '1' indicates the presence of the ltw_offset field.

piecewise_rate_flag – This is a 1-bit field which when set to '1' indicates the presence of the piecewise_rate field.

seamless_splice_flag – This is a 1-bit flag which when set to '1' indicates that the splice_type and DTS_next_AU fields are present. A value of '0' indicates that neither splice_type nor DTS_next_AU fields are present. This field shall not be set to '1' in Transport Stream packets in which the splicing_point_flag is not set to '1'. Once it is set to '1' in a Transport Stream packet in which the splice_countdown is positive, it shall be set to '1' in all the subsequent Transport Stream packets of the same PID that have the splicing_point_flag set to '1', until the packet in which the splice_countdown reaches zero (including this packet).

When this flag is set, and if the elementary stream carried in this PID is not an ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream, then the splice_type field shall be set to '0000'. If the elementary stream carried in this PID is an ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream, it shall fulfil the constraints indicated by the splice_type value.

ltw_valid_flag (legal time window_valid_flag) – This is a 1-bit field which when set to '1' indicates that the value of the ltw_offset shall be valid. A value of '0' indicates that the value in the ltw_offset field is undefined.

ltw_offset (legal time window offset) – This is a 15-bit field, the value of which is defined only if the ltw_valid flag has a value of '1'. When defined, the legal time window offset is in units of (300/f_s) seconds, where f_s is the system clock frequency of the program that this PID belongs to, and fulfils:

$$offset = t_1(i) - t(i)$$

$$ltw\_offset = offset \,//\, 1$$

where i is the index of the first byte of this Transport Stream packet, offset is the value encoded in this field, t(i) is the arrival time of byte i in the T-STD, and $t_1(i)$ is the upper bound in time of a time interval called the Legal Time Window which is associated with this Transport Stream packet.

The Legal Time Window has the property that if this Transport Stream is delivered to a T-STD starting at time $t_1(i)$, i.e., at the end of its Legal Time Window, and all other Transport Stream packets of the same program are delivered at the end of their Legal Time Windows, then:

- For video – The MB_n buffer for this PID in the T-STD shall contain less than 184 bytes of elementary stream data at the time the first byte of the payload of this Transport Stream packet enters it, and no buffer violations in the T-STD shall occur.
- For audio – The B_n buffer for this PID in the T-STD shall contain less than BS_{dec} + 1 bytes of elementary stream data at the time the first byte of this Transport Stream packet enters it, and no buffer violations in the T-STD shall occur.

Depending on factors including the size of the buffer MB_n and the rate of data transfer between MB_n and EB_n, it is possible to determine another time $t_0(i)$, such that if this packet is delivered anywhere in the interval $[t_0(i), t_1(i)]$, no T-STD buffer violations will occur. This time interval is called the Legal Time Window. The value of t_0 is not defined in this Recommendation | International Standard.

The information in this field is intended for devices such as remultiplexers which may need this information in order to reconstruct the state of the buffers MB_n.

piecewise_rate – The meaning of this 22-bit field is only defined when both the ltw_flag and the ltw_valid_flag are set to '1'. When defined, it is a positive integer specifying a hypothetical bitrate R which is used to define the end times of the Legal Time Windows of Transport Stream packets of the same PID that follow this packet but do not include the legal_time_window_offset field.

Assume that the first byte of this Transport Stream packet and the N following Transport Stream packets of the same PID have indices A_i, A_{i+1}, ..., A_{i+N}, respectively, and that the N latter packets do not have a value encoded in the field legal_time_window_offset. Then the values $t_1(A_{i+j})$ shall be determined by:

$$t_1(A_{1+j}) = t_1(A_i) + j \times 188 \times 8 \; bits/byte/R$$

where j goes from 1 to N.

All packets between this packet and the next packet of the same PID to include a legal_time_window_offset field shall be treated as if they had the value:

$$offset = t_1(A_i) - t(A_i)$$

corresponding to the value $t_1(.)$ as computed by the formula above encoded in the legal_time_window_offset field. t(j) is the arrival time of byte j in the T-STD.

The meaning of this field is not defined when it is present in a Transport Stream packet with no legal_time_window_offset field.

splice_type – This is a 4-bit field. From the first occurrence of this field onwards, it shall have the same value in all the subsequent Transport Stream packets of the same PID in which it is present, until the packet in which the splice_countdown reaches zero (including this packet). If the elementary stream carried in that PID is not an ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream, then this field shall have the value '0000'. If the elementary stream carried in that PID is an ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream, then this field indicates the conditions that shall be respected by this elementary stream for splicing purposes. These conditions are defined as a function of profile, level and splice_type in Table 2-7 through Table 2-20.

In these tables, a value for 'splice_decoding_delay' and 'max_splice_rate' means that the following conditions shall be satisfied by the video elementary stream:

1) The last byte of the coded picture ending in the Transport Stream packet in which the splice_countdown reaches zero shall remain in the VBV buffer of the VBV model for an amount of time equal to (splice_decoding_delay $t_{n+1} - t_n$), where for the purpose of this subclause:
   - n is the index of the coded picture ending in the Transport Stream packet in which the splice_countdown reaches zero, i.e., the coded picture referred to above.
   - t_n is defined in C.3.1 of ITU-T Rec. H.262 | ISO/IEC 13818-2.
   - $(t_{n+1} - t_n)$ is defined in C.9 through C.12 of ITU-T Rec. H.262 | ISO/IEC 13818-2.

     NOTE – t_n is the time when coded picture n is removed from the VBV buffer, and $(t_{n+1} - t_n)$ is the duration for which picture n is presented.
2) The VBV buffer of the VBV model shall not overflow if its input is switched at the splicing point to a stream of a constant rate equal to 'max_splice_rate' for an amount of time equal to 'splice_decoding_delay'.

Table 2-7 – Splice parameters Table 1
Simple Profile Main Level, Main Profile Main Level, SNR Profile Main Level (both layers), Spatial Profile High-1440 Level (base layer), High Profile Main Level (middle + base layers), Multi-view Profile Main Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 15.0 × 10^6 bit/s
0001	splice_decoding_delay = 150 ms; max_splice_rate = 12.0 × 10^6 bit/s
0010	splice_decoding_delay = 225 ms; max_splice_rate = 8.0 × 10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 7.2 × 10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-8 – Splice parameters Table 2
Main Profile Low Level, SNR Profile Low Level (both layers), High Profile Main Level (base layer), Multi-view Profile Low Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 115 ms; max_splice_rate = 4.0 × 10^6 bit/s
0001	splice_decoding_delay = 155 ms; max_splice_rate = 3.0 × 10^6 bit/s
0010	splice_decoding_delay = 230 ms; max_splice_rate = 2.0 × 10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 1.8 ×10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-9 – Splice parameters Table 3
Main Profile High-1440 Level, Spatial Profile High-1440 Level (all layers), High Profile High-1440 Level (middle + base layers), Multi-view Profile High-1440 Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 60.0 × 10^6 bit/s
0001	splice_decoding_delay = 160 ms; max_splice_rate = 45.0 × 10^6bit/s
0010	splice_decoding_delay = 240 ms; max_splice_rate = 30.0 × 10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 28.5 × 10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-10 – Splice parameters Table 4
Main Profile High Level, High Profile High-1440 Level (all layers),
High Profile High Level (middle + base layers),
Multi-view Profile High Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 80.0×10^6 bit/s
0001	splice_decoding_delay = 160 ms; max_splice_rate = 60.0×10^6 bit/s
0010	splice_decoding_delay = 240 ms; max_splice_rate = 40.0×10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 38.0×10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-11 – Splice parameters Table 5
SNR Profile Low Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 115 ms; max_splice_rate = 3.0×10^6 bit/s
0001	splice_decoding_delay = 175 ms; max_splice_rate = 2.0×10^6 bit/s
0010	splice_decoding_delay = 250 ms; max_splice_rate = 1.4×10^6 bit/s
0011-1011	Reserved
1100-1111	User-defined

Table 2-12 – Splice parameters Table 6
SNR Profile Main Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 115 ms; max_splice_rate = 10.0×10^6 bit/s
0001	splice_decoding_delay = 145 ms; max_splice_rate = 8.0×10^6 bit/s
0010	splice_decoding_delay = 235 ms; max_splice_rate = 5.0×10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 4.7×10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-13 – Splice parameters Table 7
Spatial Profile High-1440 Level (middle + base layers) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 40.0×10^6 bit/s
0001	splice_decoding_delay = 160 ms; max_splice_rate = 30.0×10^6 bit/s
0010	splice_decoding_delay = 240 ms; max_splice_rate = 20.0×10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 19.0×10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-14 – Splice parameters Table 8
High Profile Main Level (all layers), High Profile High-1440 Level (base layer) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 20.0 × 10^6 bit/s
0001	splice_decoding_delay = 160 ms; max_splice_rate = 15.0 × 10^6 bit/s
0010	splice_decoding_delay = 240 ms; max_splice_rate = 10.0 × 10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 9.5 × 10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-15 – Splice parameters Table 9
High Profile High Level (base layer),
Multi-view Profile Main Level (both layers) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 25.0 × 10^6 bit/s
0001	splice_decoding_delay = 165 ms; max_splice_rate = 18.0 × 10^6 bit/s
0010	splice_decoding_delay = 250 ms; max_splice_rate = 12.0 × 10^6 bit/s
0011-1011	Reserved
1100-1111	User-defined

Table 2-16 – Splice parameters Table 10
High Profile High Level (all layers),
Multi-view Profile High-1440 Level (both layers) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 100.0 × 10^6 bit/s
0001	splice_decoding_delay = 160 ms; max_splice_rate = 75.0 × 10^6 bit/s
0010	splice_decoding_delay = 240 ms; max_splice_rate = 50.0 × 10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 48.0 × 10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-17 – Splice parameters Table 11
4:2:2 Profile Main Level Video

splice_type	Conditions
0000	splice_decoding_delay = 45 ms; max_splice_rate = 50.0 × 10^6 bit/s
0001	splice_decoding_delay = 90 ms; max_splice_rate = 50.0 × 10^6 bit/s
0010	splice_decoding_delay = 180 ms; max_splice_rate = 50.0 × 10^6 bit/s
0011	splice_decoding_delay = 225 ms; max_splice_rate = 40.0 × 10^6 bit/s
0100	splice_decoding_delay = 250 ms; max_splice_rate = 36.0 × 10^6 bit/s
0101-1011	Reserved
1100-1111	User-defined

Table 2-18 – Splice parameters Table 12
Multi-view Profile Low Level (both layers) Video

splice_type	Conditions
0000	splice_decoding_delay = 115 ms; max_splice_rate = 8.0×10^6 bit/s
0001	splice_decoding_delay = 155 ms; max_splice_rate = 6.0×10^6 bit/s
0010	splice_decoding_delay = 230 ms; max_splice_rate = 4.0×10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 3.7×10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-19 – Splice parameters Table 13
Multi-view Profile High Level (both layers) Video

splice_type	Conditions
0000	splice_decoding_delay = 120 ms; max_splice_rate = 130.0×10^6 bit/s
0001	splice_decoding_delay = 150 ms; max_splice_rate = 104.0×10^6 bit/s
0010	splice_decoding_delay = 240 ms; max_splice_rate = 65.0×10^6 bit/s
0011	splice_decoding_delay = 250 ms; max_splice_rate = 62.4×10^6 bit/s
0100-1011	Reserved
1100-1111	User-defined

Table 2-20 – Splice parameters Table 14
4:2:2 Profile High Level Video

splice_type	Conditions
0000	splice_decoding_delay = 45 ms; max_splice_rate = 300.0×10^6 bit/s
0001	splice_decoding_delay = 90 ms; max_splice_rate = 300.0×10^6 bit/s
0010-0011	Reserved
0100	splice_decoding_delay = 250 ms; max_splice_rate = 180.0×10^6 bit/s
0101-1011	Reserved
1100-1111	User-defined

DTS_next_AU (decoding time stamp next access unit) – This is a 33-bit field, coded in three parts. In the case of continuous and periodic decoding through this splicing point it indicates the decoding time of the first access unit following the splicing point. This decoding time is expressed in the time base which is valid in the Transport Stream packet in which the splice_countdown reaches zero. From the first occurrence of this field onwards, it shall have the same value in all the subsequent Transport Stream packets of the same PID in which it is present, until the packet in which the splice_countdown reaches zero (including this packet).

stuffing_byte – This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder. It is discarded by the decoder.

2.4.3.6 PES packet

See Table 2-21.

Table 2-21 – PES packet

Syntax	No. of bits	Mnemonic
PES_packet() {		
packet_start_code_prefix	**24**	**bslbf**
stream_id	**8**	**uimsbf**
PES_packet_length	**16**	**uimsbf**
if (stream_id != program_stream_map		
&& stream_id != padding_stream		
&& stream_id != private_stream_2		
&& stream_id != ECM		
&& stream_id != EMM		
&& stream_id != program_stream_directory		
&& stream_id != DSMCC_stream		
&& stream_id != ITU-T Rec. H.222.1 type E stream) {		
'10'	**2**	**bslbf**
PES_scrambling_control	**2**	**bslbf**
PES_priority	**1**	**bslbf**
data_alignment_indicator	**1**	**bslbf**
copyright	**1**	**bslbf**
original_or_copy	**1**	**bslbf**
PTS_DTS_flags	**2**	**bslbf**
ESCR_flag	**1**	**bslbf**
ES_rate_flag	**1**	**bslbf**
DSM_trick_mode_flag	**1**	**bslbf**
additional_copy_info_flag	**1**	**bslbf**
PES_CRC_flag	**1**	**bslbf**
PES_extension_flag	**1**	**bslbf**
PES_header_data_length	**8**	**uimsbf**
if (PTS_DTS_flags == '10') {		
'0010'	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
if (PTS_DTS_flags == '11') {		
'0011'	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
'0001'	**4**	**bslbf**
DTS [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
DTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
DTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
if (ESCR_flag == '1') {		
Reserved	**2**	**bslbf**
ESCR_base[32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
ESCR_base[29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
ESCR_base[14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
ESCR_extension	**9**	**uimsbf**
marker_bit	**1**	**bslbf**
}		

Syntax	No. of bits	Mnemonic
if (ES_rate_flag == '1') {		
marker_bit	**1**	**bslbf**
ES_rate	**22**	**uimsbf**
marker_bit	**1**	**bslbf**
}		
if (DSM_trick_mode_flag == '1') {		
trick_mode_control	**3**	**uimsbf**
if (trick_mode_control == fast_forward) {		
field_id	**2**	**bslbf**
intra_slice_refresh	**1**	**bslbf**
frequency_truncation	**2**	**bslbf**
}		
else if (trick_mode_control == slow_motion) {		
rep_cntrl	**5**	**uimsbf**
}		
else if (trick_mode_control == freeze_frame) {		
field_id	**2**	**uimsbf**
Reserved	**3**	**bslbf**
}		
else if (trick_mode_control == fast_reverse) {		
field_id	**2**	**bslbf**
intra_slice_refresh	**1**	**bslbf**
frequency_truncation	**2**	**bslbf**
else if (trick_mode_control == slow_reverse) {		
rep_cntrl	**5**	**uimsbf**
}		
Else		
Reserved	**5**	**bslbf**
}		
if (additional_copy_info_flag == '1') {		
marker_bit	**1**	**bslbf**
additional_copy_info	**7**	**bslbf**
}		
if (PES_CRC_flag == '1') {		
previous_PES_packet_CRC	**16**	**bslbf**
}		
if (PES_extension_flag == '1') {		
PES_private_data_flag	**1**	**bslbf**
pack_header_field_flag	**1**	**bslbf**
program_packet_sequence_counter_flag	**1**	**bslbf**
P-STD_buffer_flag	**1**	**bslbf**
Reserved	**3**	**bslbf**
PES_extension_flag_2	**1**	**bslbf**
if (PES_private_data_flag == '1') {		
PES_private_data	**128**	**bslbf**
}		
if (pack_header_field_flag == '1') {		
pack_field_length	**8**	**uimsbf**
pack_header()		
}		
if (program_packet_sequence_counter_flag == '1') {		
marker_bit	**1**	**bslbf**
program_packet_sequence_counter	**7**	**uimsbf**
marker_bit	**1**	**bslbf**
MPEG1_MPEG2_identifier	**1**	**bslbf**
original_stuff_length	**6**	**uimsbf**
}		
if (P-STD_buffer_flag == '1') {		
'01'	**2**	**bslbf**
P-STD_buffer_scale	**1**	**bslbf**
P-STD_buffer_size	**13**	**uimsbf**
}		
if (PES_extension_flag_2 == '1') {		
marker_bit	**1**	**bslbf**
PES_extension_field_length	**7**	**uimsbf**
stream_id_extension_flag	**1**	**bslbf**
If (stream id extension flag == '0') {		
stream_id_extension	**7**	**uimsbf**
for (i = 0; i < PES extension field length; i++){		
reserved	**8**	**bslbf**
}		
}		

Syntax	No. of bits	Mnemonic
}		
}		
for (i < 0; i < N1; i++) {		
stuffing_byte	**8**	**bslbf**
}		
for (i < 0; i < N2; i++) {		
PES_packet_data_byte	**8**	**bslbf**
}		
}		
else if (stream_id == program_stream_map		
|| stream_id == private_stream_2		
|| stream_id == ECM		
|| stream_id == EMM		
|| stream_id == program_stream_directory		
|| stream_id == DSMCC_stream		
|| stream_id == ITU-T Rec. H.222.1 type E stream) {		
for (i = 0; i < PES_packet_length; i++) {		
PES_packet_data_byte	**8**	**bslbf**
}		
}		
else if (stream_id == padding_stream) {		
for (i < 0; i < PES_packet_length; i++) {		
padding_byte	**8**	**bslbf**
}		
}		
}		

2.4.3.7 Semantic definition of fields in PES packet

packet_start_code_prefix – The packet_start_code_prefix is a 24-bit code. Together with the stream_id that follows it constitutes a packet start code that identifies the beginning of a packet. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001).

stream_id – In Program Streams, the stream_id specifies the type and number of the elementary stream as defined by the stream_id Table 2-22. In Transport Streams, the stream_id may be set to any valid value which correctly describes the elementary stream type as defined in Table 2-22. In Transport Streams, the elementary stream type is specified in the Program Specific Information as specified in 2.4.4.

PES_packet_length – A 16-bit field specifying the number of bytes in the PES packet following the last byte of the field. A value of 0 indicates that the PES packet length is neither specified nor bounded and is allowed only in PES packets whose payload consists of bytes from a video elementary stream contained in Transport Stream packets.

PES_scrambling_control – The 2-bit PES_scrambling_control field indicates the scrambling mode of the PES packet payload. When scrambling is performed at the PES level, the PES packet header, including the optional fields when present, shall not be scrambled (see Table 2-23).

Table 2-22 – Stream_id assignments

Stream_id	Note	stream coding
1011 1100	1	program_stream_map
1011 1101	2	private_stream_1
1011 1110		padding_stream
1011 1111	3	private_stream_2
110x xxxx		ISO/IEC 13818-3 or ISO/IEC 11172-3 or ISO/IEC 13818-7 or ISO/IEC 14496-3 audio stream number x xxxx
1110 xxxx		ITU-T Rec. H.262 | ISO/IEC 13818-2, ISO/IEC 11172-2, ISO/IEC 14496-2 or ITU-T Rec. H.264 | ISO/IEC 14496-10 video stream number xxxx
1111 0000	3	ECM_stream
1111 0001	3	EMM_stream
1111 0010	5	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A or ISO/IEC 13818-6_DSMCC_stream
1111 0011	2	ISO/IEC_13522_stream
1111 0100	6	ITU-T Rec. H.222.1 type A

Table 2-22 – Stream_id assignments

Stream_id	Note	stream coding
1111 0101	6	ITU-T Rec. H.222.1 type B
1111 0110	6	ITU-T Rec. H.222.1 type C
1111 0111	6	ITU-T Rec. H.222.1 type D
1111 1000	6	ITU-T Rec. H.222.1 type E
1111 1001	7	ancillary_stream
1111 1010		ISO/IEC 14496-1_SL-packetized_stream
1111 1011		ISO/IEC 14496-1_FlexMux_stream
1111 1100		metadata stream
1111 1101	8	extended_stream_id
1111 1110		reserved data stream
1111 1111	4	program_stream_directory

The notation x means that the values '0' or '1' are both permitted and results in the same stream type. The stream number is given by the values taken by the x's.

NOTE 1 – PES packets of type program_stream_map have unique syntax specified in 2.5.4.1.

NOTE 2 – PES packets of type private_stream_1 and ISO/IEC_13552_stream follow the same PES packet syntax as those for ITU-T Rec. H.262 | ISO/IEC 13818-2 video and ISO/IEC 13818-3 audio streams.

NOTE 3 – PES packets of type private_stream_2, ECM_stream and EMM_stream are similar to private_stream_1 except no syntax is specified after PES_packet_length field.

NOTE 4 – PES packets of type program_stream_directory have a unique syntax specified in 2.5.5.

NOTE 5 – PES packets of type DSM-CC_stream have a unique syntax specified in ISO/IEC 13818-6.

NOTE 6 – This stream_id is associated with stream_type 0x09 in Table 2-34.

NOTE 7 – This stream_id is only used in PES packets, which carry data from a Program Stream or an ISO/IEC 11172-1 System Stream, in a Transport Stream (refer to 2.4.3.8).

NOTE 8 – The use of stream_id 0xFD (extended_stream_id) identifies that this PES packet employs an extended syntax to permit additional stream types to be identified.

Table 2-23 – PES scrambling control values

Value	Description
00	Not scrambled
01	User-defined
10	User-defined
11	User-defined

PES_priority – This is a 1-bit field indicating the priority of the payload in this PES packet. A '1' indicates a higher priority of the payload of the PES packet payload than a PES packet payload with this field set to '0'. A multiplexor can use the PES_priority bit to prioritize its data within an elementary stream. This field shall not be changed by the transport mechanism.

data_alignment_indicator – This is a 1-bit flag. When set to a value of '1', it indicates that the PES packet header is immediately followed by the video syntax element or audio sync word indicated in the data_stream_alignment_descriptor in 2.6.10 if this descriptor is present. If set to a value of '1' and the descriptor is not present, alignment as indicated in alignment_type '01' in Table 2-53, Table 2-54 or Table 55 is required. When set to a value of '0', it is not defined whether any such alignment occurs or not.

copyright – This is a 1-bit field. When set to '1' it indicates that the material of the associated PES packet payload is protected by copyright. When set to '0' it is not defined whether the material is protected by copyright. A copyright descriptor described in 2.6.24 is associated with the elementary stream which contains this PES packet and the copyright flag is set to '1' if the descriptor applies to the material contained in this PES packet.

original_or_copy – This is a 1-bit field. When set to '1' the contents of the associated PES packet payload is an original. When set to '0' it indicates that the contents of the associated PES packet payload is a copy.

PTS_DTS_flags – This is a 2-bit field. When the PTS_DTS_flags field is set to '10', the PTS fields shall be present in the PES packet header. When the PTS_DTS_flags field is set to '11', both the PTS fields and DTS fields shall be present in the PES packet header. When the PTS_DTS_flags field is set to '00' no PTS or DTS fields shall be present in the PES packet header. The value '01' is forbidden.

ESCR_flag – A 1-bit flag, which when set to '1' indicates that ESCR base and extension fields are present in the PES packet header. When set to '0' it indicates that no ESCR fields are present.

ES_rate_flag – A 1-bit flag, which when set to '1' indicates that the ES_rate field is present in the PES packet header. When set to '0' it indicates that no ES_rate field is present.

DSM_trick_mode_flag – A 1-bit flag, which when set to '1' it indicates the presence of an 8-bit trick mode field. When set to '0' it indicates that this field is not present.

additional_copy_info_flag – A 1-bit flag, which when set to '1' indicates the presence of the additional_copy_info field. When set to '0' it indicates that this field is not present.

PES_CRC_flag – A 1-bit flag, which when set to '1' indicates that a CRC field is present in the PES packet. When set to '0' it indicates that this field is not present.

PES_extension_flag – A 1-bit flag, which when set to '1' indicates that an extension field exists in this PES packet header. When set to '0' it indicates that this field is not present.

PES_header_data_length – An 8-bit field specifying the total number of bytes occupied by the optional fields and any stuffing bytes contained in this PES packet header. The presence of optional fields is indicated in the byte that precedes the PES_header_data_length field.

marker_bit – A marker_bit is a 1-bit field that has the value '1'.

PTS (presentation time stamp) – Presentation times shall be related to decoding times as follows: The PTS is a 33-bit number coded in three separate fields. It indicates the time of presentation, $tp_n(k)$, in the system target decoder of a presentation unit k of elementary stream n. The value of PTS is specified in units of the period of the system clock frequency divided by 300 (yielding 90 kHz). The presentation time is derived from the PTS according to equation 2-11 below. Refer to 2.7.4 for constraints on the frequency of coding presentation timestamps.

$$PTS(k) = ((system\_clock\_frequency \times tp_n(k)) DIV\ 300) \% 2^{33} \qquad (2\text{-}11)$$

where $tp_n(k)$ is the presentation time of presentation unit $P_n(k)$.

In the case of audio, if a PTS is present in PES packet header it shall refer to the first access unit commencing in the PES packet. An audio access unit commences in a PES packet if the first byte of the audio access unit is present in the PES packet.

In the case of ISO/IEC 11172-2 video or ISO/IEC 14496-2 video, if a PTS is present in a PES packet header, it shall refer to the access unit containing the first picture start code that commences in this PES packet. A picture start code commences in a PES packet if the first byte of the picture start code is present in the PES packet. For I- and P-pictures in non-low_delay sequences and in the case when there is no decoding discontinuity between access units (AUs) k and k', the presentation time $t_{pn}(k)$ shall be equal to the decoding time $t_{dn}(k')$ of the next transmitted I- or P-picture (refer to 2.7.5). If there is a decoding discontinuity, or the stream ends, the difference between $t_{pn}(k)$ and $t_{dn}(k)$ shall be the same as if the original stream had continued without a discontinuity and without ending.

NOTE 1 – A low_delay sequence is an ISO/IEC 14496-2 video sequence in which the low_delay flag is set to '1' (refer to 6.2.3 of ISO/IEC 14496-2).

For ITU-T Rec. H.262 | ISO/IEC 13818-2 video, if a PTS is present in a PES packet header, it shall refer to the access unit containing the first picture start code that commences in this PES packet. A picture start code commences in a PES packet if the first byte of the picture start code is present in the PES packet. For I- and P-coded frames in non-low_delay sequences and in the case when there is no decoding discontinuity between access units (AUs) k and k', the presentation time $t_{pn}(k)$ shall be equal to the decoding time $t_{dn}(k')$ of the next transmitted I- or P-coded frame (refer to 2.7.5). If there is a decoding discontinuity, or the stream ends, the difference between $t_{pn}(k)$ and $t_{dn}(k)$ shall be the same as if the original stream had continued without a discontinuity and without ending.

NOTE 2 – A low_delay sequence is an ITU-T Rec. H.262 | ISO/IEC 13818-2 video sequence in which the low_delay flag is set to '1' (refer to 6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2). Also note that for field pictures the presentation time refers to the first field picture of the coded frame.

For ITU-T Rec. H.264 | ISO/IEC 14496-10 video, if a PTS is present in the PES packet header, it shall refer to the first AVC access unit that commences in this PES packet. An AVC access unit commences in a PES packet if the first byte of the AVC access unit is present in the PES packet. To achieve consistency between the STD model and the HRD

model defined in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10, for each decoded AVC access unit, the PTS value in the STD shall, within the accuracy of their respective clocks, indicate the same instant in time as the nominal DPB output time in the HRD, defined herein as $t_{o,n,dpb}(n) = t_{r,n}(n) + t_c * dpb\_output\_delay(n)$, where $t_{r,n}(n)$, t_c, and dpb_output_delay(n) are defined as in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10.

NOTE 3 – Different clocks may be used for derivation of PTS and $t_{o,n,dpb}(n)$.

The presentation time $t_{pn}(k)$ shall be equal to the decoding time $t_{dn}(k)$ for:

- audio access units;
- access units in ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 14496-2 low delay video sequences;
- B-pictures in ISO/IEC 11172-2, ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 14496-2 video streams.

If there is filtering in audio, it is assumed by the system model that filtering introduces no delay, hence the sample referred to by PTS at encoding is the same sample referred to by PTS at decoding. In the case of scalable coding refer to 2.7.6.

DTS (decoding time stamp) – The DTS is a 33-bit number coded in three separate fields. It indicates the decoding time, $td_n(j)$, in the system target decoder of an access unit j of elementary stream n. The value of DTS is specified in units of the period of the system clock frequency divided by 300 (yielding 90 kHz). The decoding time derived from the DTS according to equation 2-12 below:

$$DTS(j) = ((system\_clock\_frequency \times td_n(j)) DIV\ 300) \% 2^{33} \qquad (2\text{-}12)$$

where $td_n(j)$ is the decoding time of access unit $A_n(j)$.

In the case of ISO/IEC 11172-2 video, ITU-T Rec. H.262 | ISO/IEC 13818-2 video, or ISO/IEC 14496-2 video, if a DTS is present in a PES packet header, it shall refer to the access unit containing the first picture start code that commences in this PES packet. A picture start code commences in a PES packet if the first byte of the picture start code is present in the PES packet.

For ITU-T Rec. H.264 | ISO/IEC 14496-10 video, if a DTS is present in the PES packet header, it shall refer to the first AVC access unit that commences in this PES packet. An AVC access unit commences in a PES packet if the first byte of the AVC access unit is present in the PES packet. To achieve consistency between the STD model and the HRD model defined in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10, for each AVC access unit the DTS value in the STD shall, within the accuracy of their respective clocks, indicate the same instant in time as the nominal CPB removal time $t_{r,n}(n)$ in the HRD, as defined in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10.

NOTE 4 – Different clocks may be used for derivation of DTS and $t_{r,n}(n)$.

In the case of scalable coding refer to 2.7.6.

ESCR_base; ESCR_extension – The elementary stream clock reference is a 42-bit field coded in two parts. The first part, ESCR_base, is a 33-bit field whose value is given by ESCR_base(i), as given in equation 2-14. The second part, ESCR_ext, is a 9-bit field whose value is given by ESCR_ext(i), as given in equation 2-15. The ESCR field indicates the intended time of arrival of the byte containing the last bit of the ESCR_base at the input of the PES-STD for PES streams (refer to 2.5.2.4).

Specifically:

$$ESCR(i) = ESCR\_base(i) \times 300 + ESCR\_ext(i) \qquad (2\text{-}13)$$

where:

$$ESCR\_base(i) = ((system\_clock\_frequency \times t(i)) DIV\ 300) \% 2^{33} \qquad (2\text{-}14)$$

$$ESCR\_ext(i) = ((system\_clock\_frequency \times t(i)) DIV\ 1) \% 300 \qquad (2\text{-}15)$$

The ESCR and ES_rate field (refer to semantics immediately following) contain timing information relating to the sequence of PES streams. These fields shall satisfy the constraints defined in 2.7.3.

ES_rate (elementary stream rate) – The ES_rate field is a 22-bit unsigned integer specifying the rate at which the system target decoder receives bytes of the PES packet in the case of a PES stream. The ES_rate is valid in the PES

packet in which it is included and in subsequent PES packets of the same PES stream until a new ES_rate field is encountered. The value of the ES_rate is measured in units of 50 bytes/second. The value '0' is forbidden. The value of the ES_rate is used to define the time of arrival of bytes at the input of a P-STD for PES streams defined in 2.5.2.4. The value encoded in the ES_rate field may vary from PES_packet to PES_packet.

trick_mode_control – A 3-bit field that indicates which trick mode is applied to the associated video stream. In cases of other types of elementary streams, the meanings of this field and those defined by the following five bits are undefined. For the definition of trick_mode status, refer to the **trick mode** section of 2.4.2.3.

When trick_mode status is false, the number of times N, a picture is output by the decoding process for progressive sequences, is specified for each picture by the repeat_first_field and top_field_first fields in the case of ITU-T Rec. H.262 | ISO/IEC 13818-2 Video, and is specified through the sequence header in the case of ISO/IEC 11172-2 Video.

For interlaced sequences, when trick_mode status is false, the number of times N, a picture is output by the decoding process for progressive sequences, is specified for each picture by the repeat_first_field and progressive_frame fields in the case of ITU-T Rec. H.262 | ISO/IEC 13818-2 Video.

When trick mode status is true, the number of times that a picture shall be displayed depends on the value of N.

When the value of this field changes or trick mode operations cease, any combination of the following may occur:

- discontinuity in the time base;
- decoding discontinuity;
- continuity counter discontinuity.

Table 2-24 – Trick mode control values

Value	Description
'000'	Fast forward
'001'	Slow motion
'010'	Freeze frame
'011'	Fast reverse
'100'	Slow reverse
'101'-'111'	Reserved

In the context of trick mode, the non-normal speed of decoding and presentation may cause the values of certain fields defined in video elementary stream data to be incorrect. Likewise, the semantic constraint on the slice structure may be invalid. The video syntax elements to which this exception applies are:

- bit_rate;
- vbv_delay;
- repeat_first_field;
- v_axis_positive;
- field_sequence;
- subcarrier;
- burst_amplitude;
- subcarrier_phase.

A decoder cannot rely on the values encoded in these fields when in trick mode.

Decoders are not normatively required to decode the trick_mode_control field. However, the following normative requirements shall apply to decoders that do decode the trick_mode_control field.

fast forward – The value '000', in the trick_mode_control field. When this value is present it indicates a fast forward video stream and defines the meaning of the following five bits in the PES packet header. The intra_slice_refresh bit may be set to '1' indicating that there may be missing macroblocks which the decoder may replace with co-sited macroblocks of previously decoded pictures. The field_id field, defined in Table 2-25, indicates which field or fields should be displayed. The frequency_truncation field indicates that a restricted set of coefficients may be included. The meaning of the values of this field are shown in Table 2-26.

slow motion – The value '001', in the trick_mode_control field. When this value is present it indicates a slow motion video stream and defines the meaning of the following five bits in the PES packet header. In the case of progressive sequences, the picture should be displayed N × rep_cntrl times, where N is defined above.

In the case of ISO/IEC 11172-2 Video and ITU-T Rec. H.262 | ISO/IEC 13818-2 Video progressive sequences, the picture should be displayed for N × rep_cntrl picture duration.

In the case of ITU-T Rec. H.262 | ISO/IEC 13818-2 interlaced sequences, the picture should be displayed for N × rep_cntrl field duration. If the picture is a frame picture, the first field to be displayed is the top field if top_field_first is 1, and the bottom field if top_field_first is '0' (refer to ITU-T Rec. H.262 | ISO/IEC 13818-2). This field is displayed for N × rep_cntrl / 2 field duration. The other field of the picture is then displayed for N – N × rep_cntrl / 2 field duration.

freeze frame – The value '010', in the trick_mode_control field. When this value is present it indicates a freeze frame video stream and defines the meaning of the following five bits in the PES packet header. The field_id field, defined in Table 2-25, identifies which field(s) should be displayed. The field_id field refers to the first video access unit that commences in the PES packet which contains the field_id field, unless the PES packet contains zero payload bytes. In the latter case the field_id field refers to the most recent previous video access unit.

fast reverse – The value '011', in the trick_mode_control field. When this value is present it indicates a fast reverse video stream and defines the meaning of the following five bits in the PES packet header. The intra_slice_refresh bit may be set to '1' indicating that there may be missing macroblocks which the decoder may replace with co-sited macroblocks of previously decoded pictures. The field_id field, defined in Table 2-25, indicates which field or fields should be displayed. The frequency_truncation field indicates that a restricted set of coefficients may be included. The meaning of the values of this field are shown in Table 2-26.

slow reverse – The value '100', in the trick_mode_control field. When this value is present it indicates a slow reverse video stream and defines the meaning of the following five bits in the PES packet header. In the case of ISO/IEC 11172-2 Video and ITU-T Rec. H.262 | ISO/IEC 13818-2 Video progressive sequences, the picture should be displayed for N × rep_cntrl picture duration, where N is defined above.

In the case of ITU-T Rec. H.262 | ISO/IEC 13818-2 interlaced sequences, the picture should be displayed for N × rep_cntrl field duration. If the picture is a frame picture, the first field to be displayed is the bottom field if top_field_first is 1, and the top field if top_field_first is '0' (refer to ITU-T Rec. H.262 | ISO/IEC 13818-2). This field is displayed for N × rep_cntrl / 2 field duration. The other field of the picture is then displayed for N – N × rep_cntrl / 2 field duration.

field_id – A 2-bit field that indicates which field(s) should be displayed. It is coded according to Table 2-25.

Table 2-25 – Field_id field control values

Value	Description
'00'	Display from top field only
'01'	Display from bottom field only
'10'	Display complete frame
'11'	Reserved

intra_slice_refresh – A 1-bit flag, which when set to '1', indicates that there may be missing macroblocks between coded slices of video data in this PES packet. When set to '0' this may not occur. For more information, see ITU-T Rec. H.262 | ISO/IEC 13818-2. The decoder may replace missing macroblocks with co-sited macroblocks of previously decoded pictures.

frequency_truncation – A 2-bit field which indicates that a restricted set of coefficients may have been used in coding the video data in this PES packet. The values are defined in Table 2-26.

Table 2-26 – Coefficient selection values

Value	Description
'00'	Only DC coefficients are non-zero
'01'	Only the first three coefficients are non-zero
'10'	Only the first six coefficients are non-zero
'11'	All coefficients may be non-zero

rep_cntrl – A 5-bit field that indicates the number of times each field in an interlaced picture should be displayed, or the number of times that a progressive picture should be displayed. It is a function of the trick_mode_control field and the top_field_first bit in the video sequence header whether the top field or the bottom field should be displayed first in the case of interlaced pictures. The value '0' is forbidden.

additional_copy_info – This 7-bit field contains private data relating to copyright information.

previous_PES_packet_CRC – The previous_PES_packet_CRC is a 16-bit field that contains the CRC value that yields a zero output of the 16 registers in the decoder similar to the one defined in Annex A, but with the polynomial:

$$x^{16} + x^{12} + x^5 + 1$$

after processing the data bytes of the previous PES packet, exclusive of the PES packet header.

NOTE 5 – This CRC is intended for use in network maintenance such as isolating the source of intermittent errors. It is not intended for use by elementary stream decoders. It is calculated only over the data bytes because PES packet header data can be modified during transport.

PES_private_data_flag – A 1-bit flag which when set to '1' indicates that the PES packet header contains private data. When set to a value of '0' it indicates that private data is not present in the PES header.

pack_header_field_flag – A 1-bit flag which when set to '1' indicates that an ISO/IEC 11172-1 pack header or a Program Stream pack header is stored in this PES packet header. If this field is in a PES packet that is contained in a Program Stream, then this field shall be set to '0'. In a Transport Stream, when set to the value '0' it indicates that no pack header is present in the PES header.

program_packet_sequence_counter_flag – A 1-bit flag which when set to '1' indicates that the program_packet_sequence_counter, MPEG1_MPEG2_identifier, and original_stuff_length fields are present in this PES packet. When set to a value of '0' it indicates that these fields are not present in the PES header.

P-STD_buffer_flag – A 1-bit flag which when set to '1' indicates that the P-STD_buffer_scale and P-STD_buffer_size are present in the PES packet header. When set to a value of '0' it indicates that these fields are not present in the PES header.

PES_extension_flag_2 – A 1-bit field which when set to '1' indicates the presence of the PES_extension_field_length field and associated fields. When set to a value of '0' this indicates that the PES_extension_field_length field and any associated fields are not present.

PES_private_data – This is a 16-byte field which contains private data. This data, combined with the fields before and after, shall not emulate the packet_start_code_prefix (0x000001).

pack_field_length – This is an 8-bit field which indicates the length, in bytes, of the pack_header_field().

program_packet_sequence_counter – The program_packet_sequence_counter field is a 7-bit field. It is an optional counter that increments with each successive PES packet from a Program Stream or from an ISO/IEC 11172-1 Stream or the PES packets associated with a single program definition in a Transport Stream, providing functionality similar to a continuity counter (refer to 2.4.3.2). This allows an application to retrieve the original PES packet sequence of a Program Stream or the original packet sequence of the original ISO/IEC 11172-1 stream. The counter will wrap around to 0 after its maximum value. Repetition of PES packets shall not occur. Consequently, no two consecutive PES packets in the program multiplex shall have identical program_packet_sequence_counter values.

MPEG1_MPEG2_identifier – A 1-bit flag which when set to '1' indicates that this PES packet carries information from an ISO/IEC 11172-1 stream. When set to '0' it indicates that this PES packet carries information from a Program Stream.

original_stuff_length – This 6-bit field specifies the number of stuffing bytes used in the original ITU-T Rec. H.222.0 | ISO/IEC 13818-1 PES packet header or in the original ISO/IEC 11172-1 packet header.

P-STD_buffer_scale – The P-STD_buffer_scale is a 1-bit field, the meaning of which is only defined if this PES packet is contained in a Program Stream. It indicates the scaling factor used to interpret the subsequent P-STD_buffer_size field. If the preceding stream_id indicates an audio stream, P-STD_buffer_scale shall have the value '0'. If the preceding stream_id indicates a video stream, P-STD_buffer_scale shall have the value '1'. For all other stream types, the value may be either '1' or '0'.

P-STD_buffer_size – The P-STD_buffer_size is a 13-bit unsigned integer, the meaning of which is only defined if this PES packet is contained in a Program Stream. It defines the size of the input buffer, BS_n, in the P-STD. If P-STD_buffer_scale has the value '0', then the P-STD_buffer_size measures the buffer size in units of 128 bytes. If P-STD_buffer_scale has the value '1', then the P-STD_buffer_size measures the buffer size in units of 1024 bytes. Thus:

$$\text{if } (P-STD\_buffer\_scale == 0)$$
$$BS_n = P-STD\_buffer\_size \times 128 \tag{2-16}$$

else:

$$BS_n = P-STD\_buffer\_size \times 1024 \tag{2-17}$$

The encoded value of the P-STD buffer size takes effect immediately when the P-STD_buffer_size field is received by the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 System Target Decoder (refer to 2.7.7).

The size BS_n shall be larger than or equal to the size of the CPB signalled by the CpbSize[cpb_cnt_minus1] specified by the NAL hrd_parameters() in the AVC video stream. If the NAL hrd_parameters() are not present in the AVC video stream, then BS_n shall be larger than or equal to the size of the NAL CPB for the byte stream format defined in Annex A of ITU-T Rec. H.264 | ISO/IEC 14496-10 as 1200 × MaxCPB for the applied level.

PES_extension_field_length – This is a 7-bit field which specifies the length, in bytes, of the data following this field in the PES extension field up to and including any reserved bytes.

stream_id_extension_flag – A 1-bit flag, which when set to '0' indicates that a stream_id_extension field is present in the PES packet header. The value of '1' for this flag is reserved.

stream_id_extension – In Program Streams, the stream_id_extension specifies the type and number of the elementary stream as defined by the stream_id_extension in Table 2-27. In Transport Streams, the stream_id_extension may be set to any valid value which correctly describes the elementary stream type as defined in Table 2-27. In Transport Streams, the elementary stream type is specified in the Program Specific Information as specified in 2.4.4. Note that this field is used as an extension of the stream_id defined above. This field shall not be used unless the value of stream_id is 1111 1101.

Table 2-27 – Stream_id_extension assignments

stream_id_extension	Note	stream coding
000 0000	1	IPMP Control Information stream
000 0001	2	IPMP stream
000 0010 … 011 1111		reserved_data_stream
100 0000 … 111 1111		private_stream

NOTE 1 – PES packets of stream_id_extension 0b000 0000 (IPMP Control Information Stream) have a unique syntax specified in ISO/IEC 13818-11 (MPEG-2 IPMP).

NOTE 2 – PES packets of stream_id_extension 0b000 0001 (IPMP Stream) have a unique syntax specified in ISO/IEC 13818-11 (MPEG-2 IPMP).

stuffing_byte – This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder, for example to meet the requirements of the channel. It is discarded by the decoder. No more than 32 stuffing bytes shall be present in one PES packet header.

PES_packet_data_byte – PES_packet_data_bytes shall be contiguous bytes of data from the elementary stream indicated by the packet's stream_id or PID. When the elementary stream data conforms to ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 13818-3, the PES_packet_data_bytes shall be byte aligned to the bytes of this Recommendation | International Standard. The byte-order of the elementary stream shall be preserved. The number of PES_packet_data_bytes, N, is specified by the PES_packet_length field. N shall be equal to the value indicated in the PES_packet_length minus the number of bytes between the last byte of the PES_packet_length field and the first PES_packet_data_byte.

In the case of a private_stream_1, private_stream_2, ECM_stream, or EMM_stream, the contents of the PES_packet_data_byte field are user definable and will not be specified by ITU-T | ISO/IEC in the future.

padding_byte – This is a fixed 8-bit value equal to '1111 1111'. It is discarded by the decoder.

2.4.3.8 Carriage of Program Streams and ISO/IEC 11172-1 Systems streams in the Transport Stream

The Transport Stream contains optional fields to support the carriage of Program Streams and ISO/IEC 11172-1 Systems streams, in a way that allows simple reconstruction of the respective stream at the decoder.

When placing a Program Stream into a Transport Stream, Program Stream PES packets with stream_id values of private_stream_1, ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 video, and ISO/IEC 13818-3 or ISO/IEC 11172-3 audio, are carried in Transport Stream packets.

For these PES packets, when reconstructing the Program Stream at the Transport Stream decoder, the PES packet data is copied to the Program Stream being reconstructed.

For Program Streams PES packets with stream_id values of program_stream_map, padding_stream, private_stream_2, ECM, EMM, DSM_CC_stream, or program_stream_directory, all the bytes of the Program Stream PES packet, except for the packet_start_code_prefix, are placed into the data_bytes fields of a new PES packet. The stream_id of this new PES packet has the value of ancillary_stream (refer to Table 2-22). This new PES packet is then carried in Transport Stream packets.

When reconstructing the Program Stream at the Transport Stream decoder, for PES packets with a stream_id value of ancillary_stream_id, packet_start_code_prefix is written to the Program Stream being reconstructed, followed by the data_byte fields from these Transport Stream PES packets.

ISO/IEC 11172-1 streams are carried within Transport Streams by first replacing ISO/IEC 11172-1 packet headers with ITU-T Rec. H.262 | ISO/IEC 13818-2 PES packet headers. ISO/IEC 11172-1 packet header field values are copied to the equivalent ITU-T Rec. H.262 | ISO/IEC 13818-2 PES packet header fields.

The program_packet_sequence_counter field is included within the header of each PES packet carrying data from a Program Stream, or an ISO/IEC 11172-1 System stream. This allows the order of PES packets in the original Program Stream, or packets in the original ISO/IEC 11172-1 System stream, to be reproduced at the decoder.

The pack_header() field of a Program Stream, or an ISO/IEC 11172-1 System stream, is carried in the Transport Stream in the header of the immediately following PES packet.

2.4.4 Program specific information

Program Specific Information (PSI) includes both ITU-T Rec. H.222.0 | ISO/IEC 13818-1 normative data and private data that enable demultiplexing of programs by decoders. Programs are composed of one or more elementary streams, each labelled with a PID. Programs, elementary streams or parts thereof may be scrambled for conditional access. However, Program Specific Information shall not be scrambled.

In Transport Streams, Program Specific Information is classified into six table structures as shown in Table 2-28. While these structures may be thought of as simple tables, they shall be segmented into sections and inserted in Transport Stream packets, some with predetermined PIDs and others with user selectable PIDs.

Table 2-28 – Program specific information

Structure Name	Stream Type	Reserved PID #	Description
Program Association Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x00	Associates Program Number and Program Map Table PID
Program Map Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	Assigned in the PAT	Specifies PID values for components of one or more programs
Network Information Table	Private	Assigned in the PAT	Physical network parameters such as FDM frequencies, Transponder Numbers, etc.
Conditional Access Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x01	Associates one or more (private) EMM streams each with a unique PID value
Transport Stream Description Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x02	Associates one or more descriptors from Table 2-45 to an entire Transport Stream
IPMP Control Information Table	ITU-T Rec. H.222.0 | ISO/IEC 13818-1	0x03	Contains IPMP Tool List, Rights Container, Tool Container defined in ISO/IEC 13818-11

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined PSI tables shall be segmented into one or more sections that are carried within transports packets. A section is a syntactic structure that shall be used for mapping each ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined PSI table into Transport Stream packets.

Along with ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined PSI tables, it is possible to carry private data tables. The means by which private information is carried within Transport Stream packets is not defined by this Specification. It may be structured in the same manner used for carrying of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined PSI tables, such that the syntax for mapping this private data is identical to that used for the mapping of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined PSI tables. For this purpose, a private section is defined. If the private data is carried in Transport Stream packets with the same PID value as Transport Stream packets carrying Program Map Tables (as identified in the Program Association Table), then the private_section syntax and semantics shall be used. The data carried in the private_data_bytes may be scrambled. However, no other fields of the private_section shall be scrambled. This private_section allows data to be transmitted with a minimum of structure. When this structure is not used, the mapping of private data within Transport Stream packets is not defined by this Recommendation | International Standard.

Sections may be variable in length. The beginning of a section is indicated by a pointer_field in the Transport Stream packet payload. The syntax of this field is specified in Table 2-29.

Adaptation fields may occur in Transport Stream packets carrying PSI sections.

Within a Transport Stream, packet stuffing bytes of value 0xFF may be found in the payload of Transport Stream packets carrying PSI and/or private_sections only after the last byte of a section. In this case all bytes until the end of the Transport Stream packet shall also be stuffing bytes of value 0xFF. These bytes may be discarded by a decoder. In such a case, the payload of the next Transport Stream packet with the same PID value shall begin with a pointer_field of value 0x00 indicating that the next section starts immediately thereafter.

Each Transport Stream shall contain one or more Transport Stream packets with PID value 0x0000. These Transport Stream packets together shall contain a complete Program Association Table, providing a complete list of all programs within the Transport Stream. The most recently transmitted version of the table with the current_next_indicator set to a value of '1' shall always apply to the current data in the Transport Stream. Any changes in the programs carried within the Transport Stream shall be described in an updated version of the Program Association Table carried in Transport Stream packets with PID value 0x0000. These sections shall all use table_id value 0x00. Only sections with this value of table_id are permitted within Transport Stream packets with PID value of 0x0000. For a new version of the PAT to become valid, all sections (as indicated in the last_section_number) with a new version_number and with the current_next_indicator set to '1' must exit B_{sys} defined in the T-STD (refer to 2.4.2). The PAT becomes valid when the last byte of the section needed to complete the table exits B_{sys}.

Whenever one or more elementary streams within a Transport Stream are scrambled, Transport Stream packets with a PID value 0x0001 shall be transmitted containing a complete Conditional Access Table including CA_descriptors associated with the scrambled streams. The transmitted Transport Stream packets will together form one complete version of the conditional access table. The most recently transmitted version of the table with the current_next_indicator set to a value of '1' shall always apply to the current data in the Transport Stream. Any changes in scrambling making the existing table invalid or incomplete shall be described in an updated version of the conditional access table. These sections will all use table_id value 0x01. Only sections with this table_id value are permitted within Transport Stream packets with a PID value of 0x0001. For a new version of the CAT to become valid, all sections (as indicated in the last_section_number) with a new version_number and with the current_next_indicator set to '1' must exit B_{sys}. The CAT becomes valid when the last byte of the section needed to complete the table exits B_{sys}.

Each Transport Stream shall contain one or more Transport Stream packets with PID values which are labelled under the program association table as Transport Stream packets containing TS program map sections. Each program listed in the Program Association Table shall be described in a unique TS_program_map_section. Every program shall be fully defined within the Transport Stream itself. Private data which has an associated elementary_PID field in the appropriate Program Map Table section is part of the program. Other private data may exist in the Transport Stream without being listed in the Program Map Table section. The most recently transmitted version of the TS_program_map_section with the current_next_indicator set to a value of '1' shall always apply to the current data within the Transport Stream. Any changes in the definition of any of the programs carried within the Transport Stream shall be described in an updated version of the corresponding section of the program map table carried in Transport Stream packets with the PID value identified as the program_map_PID for that specific program. All Transport Stream packets which carry a given TS_program_map_section shall have the same PID value. During the continuous existence of a program, including all of its associated events, the program_map_PID shall not change. A program definition shall not span more than one TS_program_map_section. A new version of a TS_program_map_section becomes valid when the last byte of that section with a new version_number and with the current_next_indicator set to '1' exits B_{sys}.

Sections with a table_id value of 0x02 shall contain Program Map Table information. Such sections may be carried in Transport Stream packets with different PID values.

The Network Information Table is optional and its contents are private. If present it is carried within Transport Stream packets that will have the same PID value, called the network_PID. The network_PID value is defined by the user and,

when present, shall be found in the Program Association Table under the reserved program_number 0x0000. If the network information table exists, it shall take the form of one or more private_sections.

The maximum number of bytes in a section of a ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined PSI table is 1024 bytes. The maximum number of bytes in a private_section is 4096 bytes.

The Transport Stream Description Table is optional. When present, the Transport Stream Description is carried within Transport Stream packets that have a PID value 0x0002 as specified in Table 2-28 and shall apply to the entire Transport Stream. Sections of the Transport Stream Description shall use a table_id value of 0x03 as specified in Table 2-31 and its contents are restricted to descriptors specified in Table 2-45. The TS_description_section becomes valid when the last byte of the section required to complete the table exits B_{sys}.

There are no restrictions on the occurrence of start codes, sync bytes or other bit patterns in PSI data, whether this Recommendation | International Standard or private.

2.4.4.1 Pointer

The pointer_field syntax is defined in Table 2-29.

Table 2-29 – Program specific information pointer

Syntax	No. of bits	Mnemonic
pointer_field	**8**	**uimsbf**

2.4.4.2 Semantics definition of fields in pointer syntax

pointer_field – This is an 8-bit field whose value shall be the number of bytes, immediately following the pointer_field until the first byte of the first section that is present in the payload of the Transport Stream packet (so a value of 0x00 in the pointer_field indicates that the section starts immediately after the pointer_field). When at least one section begins in a given Transport Stream packet, then the payload_unit_start_indicator (refer to 2.4.3.2) shall be set to '1' and the first byte of the payload of that Transport Stream packet shall contain the pointer. When no section begins in a given Transport Stream packet, then the payload_unit_start_indicator shall be set to '0' and no pointer shall be sent in the payload of that packet.

2.4.4.3 Program Association Table

The Program Association Table provides the correspondence between a program_number and the PID value of the Transport Stream packets which carry the program definition. The program_number is the numeric label associated with a program.

The overall table is contained in one or more sections with the following syntax. It may be segmented to occupy multiple sections (see Table 2-30).

Table 2-30 – Program association section

Syntax	No. of bits	Mnemonic
program_association_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
transport_stream_id	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i = 0; i < N; i++) {		
program_number	**16**	**uimsbf**
reserved	**3**	**bslbf**
if (program_number = = '0') {		

Table 2-30 – Program association section

Syntax	No. of bits	Mnemonic
network_PID	**13**	**uimsbf**
}		
else {		
program_map_PID	**13**	**uimsbf**
}		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.4 Table_id assignments

The table_id field identifies the contents of a Transport Stream PSI section as shown in Table 2-31.

Table 2-31 – table_id assignment values

Value	Description
0x00	program_association_section
0x01	conditional_access_section (CA_section)
0x02	TS_program_map_section
0x03	TS_description_section
0x04	ISO_IEC_14496_scene_description_section
0x05	ISO_IEC_14496_object_descriptor_section
0x06	Metadata_section
0x07	IPMP_Control_Information_section (defined in ISO/IEC 13818-11)
0x08-0x3F	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 reserved
0x40-0xFE	User private
0xFF	Forbidden

2.4.4.5 Semantic definition of fields in program association section

table_id – This is an 8-bit field, which shall be set to 0x00 as shown in Table 2-31.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field which shall be set to '1'.

section_length – This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the section, starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021 (0x3FD).

transport_stream_id – This is a 16-bit field which serves as a label to identify this Transport Stream from any other multiplex within a network. Its value is defined by the user.

version_number – This 5-bit field is the version number of the whole Program Association Table. The version number shall be incremented by 1 modulo 32 whenever the definition of the Program Association Table changes. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable Program Association Table. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable Program Association Table.

current_next_indicator – A 1-bit indicator, which when set to '1' indicates that the Program Association Table sent is currently applicable. When the bit is set to '0', it indicates that the table sent is not yet applicable and shall be the next table to become valid.

section_number – This 8-bit field gives the number of this section. The section_number of the first section in the Program Association Table shall be 0x00. It shall be incremented by 1 with each additional section in the Program Association Table.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the complete Program Association Table.

program_number – Program_number is a 16-bit field. It specifies the program to which the program_map_PID is applicable. When set to 0x0000, then the following PID reference shall be the network PID. For all other cases the value of this field is user defined. This field shall not take any single value more than once within one version of the Program Association Table.

NOTE – The program_number may be used as a designation for a broadcast channel, for example.

network_PID – The network_PID is a 13-bit field, which is used only in conjunction with the value of the program_number set to 0x0000, specifies the PID of the Transport Stream packets which shall contain the Network Information Table. The value of the network_PID field is defined by the user, but shall only take values as specified in Table 2-3. The presence of the network_PID is optional.

program_map_PID – The program_map_PID is a 13-bit field specifying the PID of the Transport Stream packets which shall contain the program_map_section applicable for the program as specified by the program_number. No program_number shall have more than one program_map_PID assignment. The value of the program_map_PID is defined by the user, but shall only take values as specified in Table 2-3.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire program association section.

2.4.4.6 Conditional access Table

The Conditional Access (CA) Table provides the association between one or more CA systems, their EMM streams and any special parameters associated with them. Refer to 2.6.16 for a definition of the descriptor() field in Table 2-32.

The table is contained in one or more sections with the following syntax. It may be segmented to occupy multiple sections.

Table 2-32 – Conditional access section

Syntax	No. of bits	Mnemonic
CA_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
reserved	**18**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i = 0; i < N; i++) {		
descriptor()		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.7 Semantic definition of fields in conditional access section

table_id – This is an 8-bit field, which shall be set to 0x01 as specified in Table 2-31.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field which shall be set to '1'.

section_length – This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10-bits specify the number of bytes of the section starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021 (0x3FD).

version_number – This 5-bit field is the version number of the entire conditional access table. The version number shall be incremented by 1 modulo 32 when a change in the information carried within the CA table occurs. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable Conditional Access Table. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable Conditional Access Table.

current_next_indicator – A 1-bit indicator, which when set to '1' indicates that the Conditional Access Table sent is currently applicable. When the bit is set to '0', it indicates that the Conditional Access Table sent is not yet applicable and shall be the next Conditional Access Table to become valid.

section_number – This 8-bit field gives the number of this section. The section_number of the first section in the Conditional Access Table shall be 0x00. It shall be incremented by 1 with each additional section in the Conditional Access Table.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the Conditional Access Table.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire conditional access section.

2.4.4.8 Program Map Table

The Program Map Table provides the mappings between program numbers and the program elements that comprise them. A single instance of such a mapping is referred to as a "program definition". The program map table is the complete collection of all program definitions for a Transport Stream. This table shall be transmitted in packets, the PID values of which are selected by the encoder. More than one PID value may be used, if desired. The table is contained in one or more sections with the following syntax. It may be segmented to occupy multiple sections. In each section, the section number field shall be set to zero. Sections are identified by the program_number field.

Definition for the descriptor() fields may be found in 2.6 (see Table 2-33).

Table 2-33 – Transport Stream program map section

Syntax	No. of bits	Mnemonic
TS_program_map_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
program_number	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
reserved	**3**	**bslbf**
PCR_PID	**13**	**uimsbf**
reserved	**4**	**bslbf**
program_info_length	**12**	**uimsbf**
for (i = 0; i < N; i++) {		
descriptor()		
}		
for (i = 0; i < N1; i++) {		
stream_type	**8**	**uimsbf**
reserved	**3**	**bslbf**
elementary_PID	**13**	**uimsbf**
reserved	**4**	**bslbf**
ES_info_length	**12**	**uimsbf**
for (i = 0; i < N2; i++) {		
descriptor()		
}		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.9 Semantic definition of fields in Transport Stream program map section

table_id – This is an 8-bit field, which in the case of a TS_program_map_section shall be always set to 0x02 as shown in Table 2-31.

section_syntax_indicator – The section_syntax_indicator is a 1-bit field which shall be set to '1'.

section_length – This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the section starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021 (0x3FD).

program_number – program_number is a 16-bit field. It specifies the program to which the program_map_PID is applicable. One program definition shall be carried within only one TS_program_map_section. This implies that a program definition is never longer than 1016 (0x3F8). See Informative Annex C for ways to deal with the cases when that length is not sufficient. The program_number may be used as a designation for a broadcast channel, for example. By describing the different program elements belonging to a program, data from different sources (e.g., sequential events) can be concatenated together to form a continuous set of streams using a program_number. For examples of applications refer to Annex C.

version_number – This 5-bit field is the version number of the TS_program_map_section. The version number shall be incremented by 1 modulo 32 when a change in the information carried within the section occurs. Version number refers to the definition of a single program, and therefore to a single section. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable TS_program_map_section. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable TS_program_map_section.

current_next_indicator – A 1-bit field, which when set to '1' indicates that the TS_program_map_section sent is currently applicable. When the bit is set to '0', it indicates that the TS_program_map_section sent is not yet applicable and shall be the next TS_program_map_section to become valid.

section_number – The value of this 8-bit field shall be 0x00.

last_section_number – The value of this 8-bit field shall be 0x00.

PCR_PID – This is a 13-bit field indicating the PID of the Transport Stream packets which shall contain the PCR fields valid for the program specified by program_number. If no PCR is associated with a program definition for private streams, then this field shall take the value of 0x1FFF. Refer to the semantic definition of PCR in 2.4.3.5 and Table 2-3 for restrictions on the choice of PCR_PID value.

program_info_length – This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the descriptors immediately following the program_info_length field.

stream_type – This is an 8-bit field specifying the type of program element carried within the packets with the PID whose value is specified by the elementary_PID. The values of stream_type are specified in Table 2-34.

NOTE – An ITU-T Rec. H.222.0 | ISO/IEC 13818-1 auxiliary stream is available for data types defined by this Specification, other than audio, video, and DSM-CC, such as Program Stream Directory and Program Stream Map.

Table 2-34 – Stream type assignments

Value	Description
0x00	ITU-T | ISO/IEC Reserved
0x01	ISO/IEC 11172-2 Video
0x02	ITU-T Rec. H.262 | ISO/IEC 13818-2 Video or ISO/IEC 11172-2 constrained parameter video stream
0x03	ISO/IEC 11172-3 Audio
0x04	ISO/IEC 13818-3 Audio
0x05	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 private_sections
0x06	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 PES packets containing private data
0x07	ISO/IEC 13522 MHEG
0x08	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Annex A DSM-CC
0x09	ITU-T Rec. H.222.1
0x0A	ISO/IEC 13818-6 type A
0x0B	ISO/IEC 13818-6 type B
0x0C	ISO/IEC 13818-6 type C

Table 2-34 – Stream type assignments

Value	Description
0x0D	ISO/IEC 13818-6 type D
0x0E	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 auxiliary
0x0F	ISO/IEC 13818-7 Audio with ADTS transport syntax
0x10	ISO/IEC 14496-2 Visual
0x11	ISO/IEC 14496-3 Audio with the LATM transport syntax as defined in ISO/IEC 14496-3
0x12	ISO/IEC 14496-1 SL-packetized stream or FlexMux stream carried in PES packets
0x13	ISO/IEC 14496-1 SL-packetized stream or FlexMux stream carried in ISO/IEC 14496_sections
0x14	ISO/IEC 13818-6 Synchronized Download Protocol
0x15	Metadata carried in PES packets
0x16	Metadata carried in metadata_sections
0x17	Metadata carried in ISO/IEC 13818-6 Data Carousel
0x18	Metadata carried in ISO/IEC 13818-6 Object Carousel
0x19	Metadata carried in ISO/IEC 13818-6 Synchronized Download Protocol
0x1A	IPMP stream (defined in ISO/IEC 13818-11, MPEG-2 IPMP)
0x1B	AVC video stream as defined in ITU-T Rec. H.264 | ISO/IEC 14496-10 Video
0x1C-0x7E	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
0x7F	IPMP stream
0x80-0xFF	User Private

elementary_PID – This is a 13-bit field specifying the PID of the Transport Stream packets which carry the associated program element.

ES_info_length – This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the descriptors of the associated program element immediately following the ES_info_length field.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex B after processing the entire Transport Stream program map section.

2.4.4.10 Syntax of the Private section

When private data is sent in Transport Stream packets with a PID value designated as a Program Map Table PID in the Program Association Table the private_section shall be used. The private_section allows data to be transmitted with a minimum of structure while enabling a decoder to parse the stream. The sections may be used in two ways: if the section_syntax_indicator is set to '1', then the whole structure common to all tables shall be used; if the indicator is set to '0', then only the fields 'table_id' through 'private_section_length' shall follow the common structure syntax and semantics and the rest of the private_section may take any form the user determines. Examples of extended use of this syntax are found in Informative Annex C.

A private table may be made of several private_sections, all with the same table_id (see Table 2-35).

Table 2-35 – Private section

Syntax	No. of bits	Mnemonic
private_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
private_indicator	**1**	**bslbf**
Reserved	**2**	**bslbf**
private_section_length	**12**	**uimsbf**
if (section_syntax_indicator = = '0') {		
for (i = 0; i < N; i++) {		
private_data_byte	**8**	**bslbf**
}		
}		
else {		
table_id_extension	**16**	**uimsbf**
Reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i = 0; i < private_section_length-9; i++) {		
private_data_byte	**8**	**bslbf**
}		
CRC_32	**32**	**rpchof**
}		
}		

2.4.4.11 Semantic definition of fields in private section

table_id – This 8-bit field, the value of which identifies the Private Table this section belongs to. Only values defined in Table 2-31 as "user private" may be used.

section_syntax_indicator – This is a 1-bit indicator. When set to '1', it indicates that the private section follows the generic section syntax beyond the private_section_length field. When set to '0', it indicates that the private_data_bytes immediately follow the private_section_length field.

private_indicator – This is a 1-bit user-definable flag that shall not be specified by ITU-T | ISO/IEC in the future.

private_section_length – A 12-bit field. It specifies the number of remaining bytes in the private section immediately following the private_section_length field up to the end of the private_section. The value in this field shall not exceed 4093 (0xFFD).

private_data_byte – The private_data_byte field is user definable and shall not be specified by ITU-T | ISO/IEC in the future.

table_id_extension – This is a 16-bit field. Its use and value are defined by the user.

version_number – This 5-bit field is the version number of the private_section. The version_number shall be incremented by 1 modulo 32 when a change in the information carried within the private_section occurs. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable private_section with the same table_id and section_number.

current_next_indicator – A 1-bit field, which when set to '1' indicates that the private_section sent is currently applicable. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable private_section. When the bit is set to '0', it indicates that the private_section sent is not yet applicable and shall be the next private_section with the same section_number and table_id to become valid.

section_number – This 8-bit field gives the number of the private_section. The section_number of the first section in a private table shall be 0x00. The section_number shall be incremented by 1 with each additional section in this private table.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the private table of which this section is a part.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire private section.

2.4.4.12 Syntax of the Transport Stream section

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 compliant bitstreams may carry the information defined in Table 2-36. ITU-T Rec. H.222.0 | ISO/IEC 13818-1 compliant decoders may decode the information defined in this table.

The Transport Stream Description Table is defined to support the carriage of descriptors as found in 2.6 for an entire Transport Stream. The descriptors shall apply to the entire Transport Stream. This table uses a table_id value of 0x03 as specified in Table 2-31 and is carried in Transport Stream packets whose PID value is 0x0002 as specified in Table 2-3.

Table 2-36 – The Transport Stream Description Table

Syntax	No. of bits	Mnemonic
TS_description_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
'0'	**1**	**bslbf**
Reserved	**2**	**bslbf**
section_length	**12**	**uimsbf**
Reserved	**18**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i = 0; i < N; i++) {		
descriptor()		
}		
CRC_32	**32**	**rpchof**
}		

2.4.4.13 Semantic definition of fields in the Transport Stream section

table_id – This is an 8-bit field, which shall be set to '0x03' as specified in Table 2-31.

section_length – This is a 12-bit field, the first two bits of which shall be '00'. The remaining 10 bits specify the number of bytes of the section, starting immediately following the section_length field, and including the CRC. The value in this field shall not exceed 1021 (0x3FD).

version_number – This 5-bit field is the version number of the whole Transport Stream Description Table. The version number shall be incremented by 1 modulo 32 whenever the definition of the Transport Stream Description Table changes. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable Transport Stream Description Table. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable Transport Stream Description Table.

current_next_indicator – A 1-bit indicator, which, when set to '1', indicates that the Transport Stream Description Table sent is currently applicable. When the bit is set to '0', it indicates that the table sent is not yet applicable and shall be the next table to become valid.

section_number – This 8-bit field gives the number of this section. The section_number of the first section in the Transport Stream Description Table shall be 0x00. It shall be incremented by 1 with each additional section in the Transport Stream Description Table.

last_section_number – This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the complete Transport Stream Description Table.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire Transport Stream Description section.

2.5 Program Stream bitstream requirements

2.5.1 Program Stream coding structure and parameters

The ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Stream coding layer allows one program of one or more elementary streams to be combined into a single stream. Data from each elementary stream are multiplexed together with information that allows synchronized presentation of the elementary streams within the program.

A Program Stream consists of one or more elementary streams from one program multiplexed together. Audio and video elementary streams consist of access units.

Elementary Stream data is carried in PES packets. A PES packet consists of a PES packet header followed by packet data. PES packets are inserted into Program Stream packs.

The PES packet header begins with a 32-bit start-code that also identifies the stream (refer to Table 2-22) to which the packet data belongs. The PES packet header may contain just a Presentation Time Stamp (PTS) or both a presentation timestamp and a Decoding Time Stamp (DTS). The PES packet header also contains other optional fields. The packet data contains a variable number of contiguous bytes from one elementary stream.

In a Program Stream, PES packets are organized in packs. A pack commences with a pack header and is followed by zero or more PES packets. The pack header begins with a 32-bit start-code. The pack header is used to store timing and bitrate information.

The Program Stream begins with a system header that optionally may be repeated. The system header carries a summary of the system parameters defined in the stream.

This Recommendation | International Standard does not specify the coded data which may be used as part of conditional access systems. This Recommendation | International Standard does, however, provide mechanisms for program service providers to transport and identify this data for decoder processing, and to correctly reference data which are here specified.

2.5.2 Program Stream system target decoder

The semantics of the Program Stream and the constraints on these semantics require exact definitions of decoding events and the times at which these events occur. The definitions needed are set out in this Specification using a hypothetical decoder known as the Program Stream system target decoder (P-STD).

The P-STD is a conceptual model used to define these terms precisely and to model the decoding process during the construction of Program Streams. The P-STD is defined only for this purpose. Neither the architecture of the P-STD nor the timing described precludes uninterrupted, synchronized playback of Program Streams from a variety of decoders with different architectures or timing schedules.

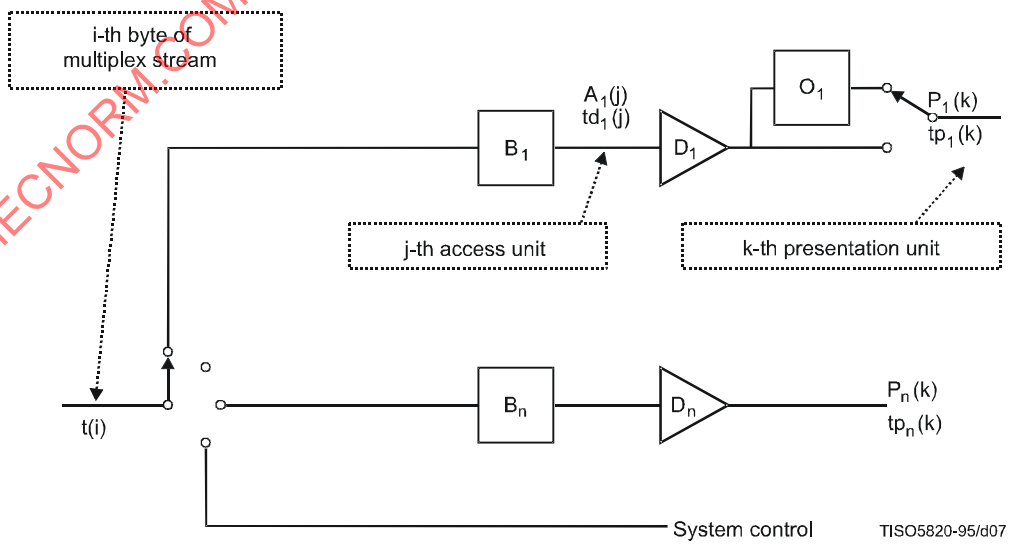

Figure 2-2 – Program Stream system target decoder notation

The following notation is used to describe the Program Stream system target decoder and is partially illustrated in Figure 2-2.

- i, i′ are indices to bytes in the Program Stream. The first byte has index 0.
- j is an index to access units in the elementary streams.
- k, k′, k″ are indices to presentation units in the elementary streams.
- n is an index to the elementary streams.
- t(i) indicates the time in seconds at which the i-th byte of the Program Stream enters the system target decoder. The value t(0) is an arbitrary constant.
- SCR(i) is the time encoded in the SCR field measured in units of the 27 MHz system clock where i is the byte index of the final byte of the system_clock_reference_base field.
- $A_n(j)$ is the j-th access unit in elementary stream n. $A_n(j)$ is indexed in decoding order.
- $td_n(j)$ is the decoding time, measured in seconds, in the system target decoder of the j-th access unit in elementary stream n.
- $P_n(k)$ is the k-th presentation unit in elementary stream n. $P_n(k)$ is indexed in presentation order.
- $tp_n(k)$ is the presentation time, measured in seconds, in the system target decoder of the k-th presentation unit in elementary stream n.
- t is time measured in seconds.
- $F_n(t)$ is the fullness, measured in bytes, of the system target decoder input buffer for elementary stream n at time t.
- B_n the input buffer in the system target decoder for elementary stream n.
- BS_n is the size of the system target decoder input buffer, measured in bytes, for elementary stream n.
- D_n is the decoder for elementary stream n.
- O_n is the reorder buffer for video elementary stream n.

2.5.2.1 System clock frequency

Timing information referenced in P-STD is carried by several data fields defined in this Specification. The fields are defined in 2.5.3.3 and 2.4.3.6. This information is coded as the sampled value of a system clock.

The value of the system clock frequency is measured in Hz and shall meet the following constraints:

- 27 000 000 – 810 <= system_clock_frequency <= 27 000 000 + 810;
- rate of change of system_clock_frequency with time <= 75×10^{-3} Hz/s.

The notation "system_clock_frequency" is used in several places in this Recommendation | International Standard to refer to the frequency of a clock meeting these requirements. For notational convenience, equations in which SCR, PTS, or DTS appear, lead to values of time which are accurate to some integral multiple of (300×2^{33}/system_clock_frequency) seconds. This is due to the encoding of SCR timing information as 33 bits of 1/300 of the system clock frequency plus 9 bits for the remainder, and encoding as 33 bits of the system clock frequency divided by 300 for PTS and DTS.

2.5.2.2 Input to the Program Stream system target decoder

Data from the Program Stream enters the system target decoder. The i-th byte enters at time t(i). The time at which this byte enters the system target decoder can be recovered from the input stream by decoding the input System Clock Reference (SCR) fields and the program_mux_rate field encoded in the pack header. The SCR, as defined in equation 2-18, is coded in two parts: one, in units the period of 1/300 × the system clock frequency, called system_clock_reference_base (see equation 2-19), and one, called system_clock_reference_ext equation (see equation 2-20), in units of the period of the system clock frequency. In the following the values encoded in these fields are denoted by SCR_base(i) and SCR_ext(i). The value encoded in the SCR field indicates time t(i), where i refers to the byte containing the last bit of the system_clock_reference_base field.

Specifically:

$$SCR(i) = SCR\_base(i) \times 300 + SCR\_ext(i) \qquad (2\text{-}18)$$

where:

$$SCR\_base(i) = ((system\_clock\_frequency \times t(i))DIV\ 300)\%2^{33} \quad (2\text{-}19)$$

$$SCR\_ext(i) = ((system\_clock\_frequency \times t(i))DIV\ 1)\%300 \quad (2\text{-}20)$$

The input arrival time, t(i), as given in equation 2-21, for all other bytes shall be constructed from SCR(i) and the rate at which data arrives, where the arrival rate within each pack is the value represented in the program_mux_rate field in that pack's header.

$$t(i) = \frac{SCR(i')}{system\_clock\_frequency} + \frac{i - i'}{program\_mux\_rate \times 50} \quad (2\text{-}21)$$

where:

i′ is the index of the byte containing the last bit of the system_clock_reference_base field in the pack header

i is the index of any byte in the pack, including the pack header

SCR(i′) is the time encoded in the system clock reference base and extension fields in units of the system clock

program_mux_rate is a field defined in 2.5.3.3.

After delivery of the last byte of a pack there may be a time interval during which no bytes are delivered to the input of the P-STD.

2.5.2.3 Buffering

The PES packet data from elementary stream n is passed to the input buffer for stream n, B_n. Transfer of byte i from the system target decoder input to B_n is instantaneous, so that byte i enters the buffer for stream n, of size BS_n, at time t(i).

Bytes present in the pack header, system headers, Program Stream Maps, Program Stream Directories, or PES packet headers of the Program Stream such as SCR, DTS, PTS, and packet_length fields, are not delivered to any of the buffers, but may be used to control the system.

The input buffer sizes BS_1 through BS_n are given by the P-STD buffer size parameter in the syntax in equations 2-16 and 2-17.

At the decoding time, $td_n(j)$, all data for the access unit that has been in the buffer longest, $A_n(j)$, and any stuffing bytes that immediately precede it that are present in the buffer at the time $td_n(j)$, are removed instantaneously at time $td_n(j)$. The decoding time $td_n(j)$ is specified in the DTS or PTS fields. Decoding times $td_n(j + 1)$, $td_n(j + 2)$, ... of access units without encoded DTS or PTS fields which directly follow access unit j may be derived from information in the elementary stream. Refer to Annex C of ITU-T Rec. H.262 | ISO/IEC 13818-2, ISO/IEC 13818-3, ISO/IEC 11172-2 or ISO/IEC 11172-3. Also refer to 2.7.5. As the access unit is removed from the buffer, it is instantaneously decoded to a presentation unit.

The Program Stream shall be constructed and t(i) shall be chosen so that the input buffers of size BS_1 through BS_n neither overflow nor underflow in the program system target decoder. That is:

$$0 \le F_n(t) \le BS_n$$

for all t and n,

and:

$$F_n(t) = 0$$

instantaneously before t = t(0).

$F_n(t)$ is the instantaneous fullness of P-STD buffer B_n.

An exception to this condition is that the P-STD buffer B_n may underflow when the low_delay flag in the video sequence header is set to '1' (refer to 2.4.2.6) or when trick_mode status is true (refer to 2.4.3.8).

For all Program Streams, the delay caused by system target decoder input buffering shall be less than or equal to one second except for still picture video data and ISO/IEC 14496 streams. The input buffering delay is the difference in time between a byte entering the input buffer and when it is decoded.

Specifically: in the case of no still picture video data and no ISO/IEC 14496 stream the delay is constrained by:

$$tdn(j) - t(i) <= 1 \text{ s}$$

in the case of still picture video data the delay is constrained by:

$$tdn(j) - t(i) <= 60 \text{ s}$$

in the case of ISO/IEC 14496 streams the delay is constrained by:

$$tdn(j) - t(i) <= 10 \text{ s}$$

for all bytes contained in access unit j.

For Program Streams, all bytes of each pack shall enter the P-STD before any byte of a subsequent pack.

When the **low_delay** flag in the video sequence extension is set to '1' (refer to 6.2.2.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2), the VBV buffer may underflow. In this case when the P-STD elementary stream buffer B_n is examined at the time specified by $td_n(j)$, the complete data for the access unit may not be present in the buffer B_n. When this case arises, the buffer shall be re-examined at intervals of two field-periods until the data for the complete access unit is present in the buffer. At this time the entire access unit shall be removed from buffer B_n instantaneously.

VBV buffer underflow is allowed to occur continuously without limit. The P-STD decoder shall remove access unit data from buffer B_n at the earliest time consistent with the paragraph above and any DTS or PTS values encoded in the bitstream. The decoder may be unable to re-establish correct decoding and display times as indicated by DTS and PTS until the VBV buffer underflow situation ceases and a PTS or DTS is found in the bitstream.

2.5.2.4 PES streams

It is possible to construct a stream of data as a contiguous stream of PES packets each containing data of the same elementary stream and with the same stream_id. Such a stream is called a PES stream. The PES-STD model for a PES stream is identical to that for the Program Stream, with the exception that the Elementary Stream Clock Reference (ESCR) is used in place of the SCR, and ES_rate in place of program_mux_rate. The demultiplexor sends data to only one elementary stream buffer.

Buffer sizes BS_n in the PES-STD model are defined as follows:

– *For ITU-T Rec. H.262 | ISO/IEC 13818-2 video:*

$$BS_n = VBV_{max}[\text{profile, level}] + BS_{oh}$$

BS_{oh} = (1/750) seconds × R_{max}[profile, level], where VBV_{max}[profile, level] and R_{max}[profile, level] are the maximum VBV size and bit rate per profile, level, and layer as defined in Tables 8-14 and 8-13, respectively, of ITU-T Rec. H.262 | ISO/IEC 13818-2. BS_{oh} is allocated for PES packet header overhead.

– *For ISO/IEC 11172-2 video:*

$$BS_n = VBV_{max} + BS_{oh}$$

BS_{oh} = (1/750) seconds × R_{max}, where R_{max} and vbv_max refer to the maximum bitrate and maximum vbv_buffer_size for a constrained parameter bitstream in ISO/IEC 11172-2 respectively.

– *For ISO/IEC 11172-3 or ISO/IEC 13818-3 audio:*

$$BS_n = 2848 \text{ bytes}$$

– *For ITU-T Rec. H.264 | ISO/IEC 14496-10 video:*

$$BS_n = 1200 \times MaxCPB[level] + BS_{oh}$$

where MaxCPB[level] is defined in Table A.1 (Level Limits) in ITU-T Rec. H.264 | ISO/IEC 14496-10 for each level.

2.5.2.5 Decoding and presentation

Decoding and presentation in the Program Stream system target decoder are the same as defined for the Transport Stream system target decoder in 2.4.2.4 and 2.4.2.5 respectively.

2.5.2.6 P-STD extensions for carriage of ISO/IEC 14496 data

For decoding of ISO/IEC 14496 data carried in a Program Stream the P-STD model is extended. For decoding of individual ISO/IEC 14496 elementary streams in the P-STD see 2.11.2. Clause 2.11.3 defines P-STD extensions and parameters for decoding of ISO/IEC 14496 scenes and associated streams.

2.5.2.7 P-STD extensions for carriage of ITU-T Rec. H.264 | ISO/IEC 14496-10 Video

For decoding of ITU-T Rec. H.264 | ISO/IEC 14496-10 video streams carried in a Program Stream in the P-STD model, see 2.14.3.2.

2.5.3 Specification of the Program Stream syntax and semantics

The following syntax describes a stream of bytes.

2.5.3.1 Program Stream

See Table 2-37.

Table 2-37 – Program Stream

Syntax	No. of bits	Mnemonic
MPEG2_program_stream() {		
do {		
pack()		
} while (nextbits() = = pack_start_code)		
MPEG_program_end_code	**32**	**bslbf**
}		

2.5.3.2 Semantic definition of fields in Program Stream

MPEG_program_end_code – The MPEG_program_end_code is the bit string '0000 0000 0000 0000 0000 0001 1011 1001' (0x000001B9). It terminates the Program Stream.

2.5.3.3 Pack layer of Program Stream

See Tables 2-38 and 2-39.

Table 2-38 – Program Stream pack

Syntax	No. of bits	Mnemonic
pack() {		
pack_header()		
while (nextbits() = -= packet_start_code_prefix) {		
PES_packet()		
}		
}		

Table 2-39 – Program Stream pack header

Syntax	No. of bits	Mnemonic
pack_header() {		
pack_start_code	**32**	**bslbf**
'01'	**2**	**bslbf**
system_clock_reference_base [32..30]	**3**	**bslbf**
marker_bit	**1**	**bslbf**
system_clock_reference_base [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
system_clock_reference_base [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
system_clock_reference_extension	**9**	**uimsbf**
marker_bit	**1**	**bslbf**
program_mux_rate	**22**	**uimsbf**
marker_bit	**1**	**bslbf**
marker_bit	**1**	**bslbf**
reserved	**5**	**bslbf**
pack_stuffing_length	**3**	**uimsbf**
for (i = 0; i < pack_stuffing_length; i++) {		
stuffing_byte	**8**	**bslbf**
}		
if (nextbits() = = system_header_start_code) {		
system_header ()		
}		
}		

2.5.3.4 Semantic definition of fields in program stream pack

pack_start_code – The pack_start_code is the bit string '0000 0000 0000 0000 0000 0001 1011 1010' (0x000001BA). It identifies the beginning of a pack.

system_clock_reference_base; system_clock_reference_extension – The system clock reference (SCR) is a 42-bit field coded in two parts. The first part, system_clock_reference_base, is a 33-bit field whose value is given by SCR_base(i) as given in equation 2-19. The second part, system_clock_reference_extension, is a 9-bit field whose value is given by SCR_ext(i), as given in equation 2-20. The SCR indicates the intended time of arrival of the byte containing the last bit of the system_clock_reference_base at the input of the program target decoder.

The frequency of coding requirements for the SCR field are given in 2.7.1.

marker_bit – A marker_bit is a 1-bit field that has the value '1'.

program_mux_rate – This is a 22-bit integer specifying the rate at which the P-STD receives the Program Stream during the pack in which it is included. The value of program_mux_rate is measured in units of 50 bytes/second. The value '0' is forbidden. The value represented in program_mux_rate is used to define the time of arrival of bytes at the input to the P-STD in 2.5.2. The value encoded in the program_mux_rate field may vary from pack to pack in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program multiplexed stream.

pack_stuffing_length – A 3-bit integer specifying the number of stuffing bytes which follow this field.

stuffing_byte – This is a fixed 8-bit value equal to '1111 1111' that can be inserted by the encoder, for example to meet the requirements of the channel. It is discarded by the decoder. In each pack header no more than 7 stuffing bytes shall be present.

2.5.3.5 System header

See Table 2-40.

Table 2-40 – Program Stream system header

Syntax	No. of bits	Mnemonic
system_header () {		
system_header_start_code	**32**	**bslbf**
header_length	**16**	**uimsbf**
marker_bit	**1**	**bslbf**
rate_bound	**22**	**uimsbf**
marker_bit	**1**	**bslbf**
audio_bound	**6**	**uimsbf**
fixed_flag	**1**	**bslbf**
CSPS_flag	**1**	**bslbf**
system_audio_lock_flag	**1**	**bslbf**
system_video_lock_flag	**1**	**bslbf**
marker_bit	**1**	**bslbf**
video_bound	**5**	**uimsbf**
packet_rate_restriction_flag	**1**	**bslbf**
reserved_bits	**7**	**bslbf**
while (nextbits () = = '1') {		
stream_id	**8**	**uimsbf**
'11'	**2**	**bslbf**
P-STD_buffer_bound_scale	**1**	**bslbf**
P-STD_buffer_size_bound	**13**	**uimsbf**
}		
}		

2.5.3.6 Semantic definition of fields in system header

system_header_start_code – The system_header_start_code is the bit string '0000 0000 0000 0000 0000 0001 1011 1011' (0x000001BB). It identifies the beginning of a system header.

header_length – This 16-bit field indicates the length in bytes of the system header following the header_length field. Future extensions of this Specification may extend the system header.

rate_bound – A 22-bit field. The rate_bound is an integer value greater than or equal to the maximum value of the program_mux_rate field coded in any pack of the Program Stream. It may be used by a decoder to assess whether it is capable of decoding the entire stream.

audio_bound – A 6-bit field. The audio_bound is an integer in the inclusive range from 0 to 32 and is set to a value greater than or equal to the maximum number of ISO/IEC 13818-3 and ISO/IEC 11172-3 audio streams in the Program Stream for which the decoding processes are simultaneously active. For the purpose of this subclause, the decoding process of an ISO/IEC 13818-3 or ISO/IEC 11172-3 audio stream is active if the STD buffer is not empty or if a Presentation Unit is being presented in the P-STD model.

fixed_flag – The fixed_flag is a 1-bit flag. When set to '1' fixed bitrate operation is indicated. When set to '0' variable bitrate operation is indicated. During fixed bitrate operation, the value encoded in all system_clock_reference fields in the multiplexed ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream shall adhere to the following linear equation:

$$SCR\_base(i) = ((c1 \times i + c2)\ DIV\ 300)\ \%\ 2^{33} \quad (2\text{-}22)$$

$$SCR\_ext(i) = ((c1 \times i + c2)\ DIV\ 300)\ \%\ 300 \quad (2\text{-}23)$$

where:

c1 is a real-valued constant valid for all i.

c2 is a real-valued constant valid for all i.

i is the index in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 multiplexed stream of the byte containing the final bit of any system_clock_reference field in the stream.

CSPS_flag – The CSPS_flag is a 1-bit field. If its value is set to '1' the Program Stream meets the constraints defined in 2.7.9.

system_audio_lock_flag – The system_audio_lock_flag is a 1-bit field indicating that there is a specified, constant rational relationship between the audio sampling rate and the system_clock_frequency in the system target decoder. The system_clock_frequency is defined in 2.5.2.1 and the audio sampling rate is specified in ISO/IEC 13818-3. The system_audio_lock_flag may only be set to '1' if, for all presentation units in all audio elementary streams in the Program Stream, the ratio of system_clock_frequency to the actual audio sampling rate, SCASR, is constant and equal to the value indicated in the following table at the nominal sampling rate indicated in the audio stream.

$$SCASR = \frac{system\_clock\_frequency}{audio\_sample\_rate\_in\_the\_P-STD} \tag{2-24}$$

The notation $\frac{X}{Y}$ denotes real division.

Nominal audio sampling frequency (kHz)	16	32	22.05	44.1	24	48
SCASR	27 000 000 / 16 000	27 000 000 / 32 000	27 000 000 / 22 050	27 000 000 / 44 100	27 000 000 / 24 000	27 000 000 / 48 000

system_video_lock_flag – The system_video_lock_flag is a 1-bit field indicating that there is a specified, constant rational relationship between the video time base and the system clock frequency in the system target decoder. The system_video_lock_flag may only be set to '1' if, for all presentation units in all video elementary streams in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program, the ratio of system_clock_frequency to the frequency of the actual video time base is constant.

For ISO/IEC 11172-2 and ITU-T Rec. H.262 | ISO/IEC 13818-2 video streams, if the system_video_lock_flag is set to '1', then the ratio of system_clock_frequency to the actual video frame rate, SCFR, shall be constant and equal to the value indicated in the following table at the nominal frame rate indicated in the video stream.

For ISO/IEC 14496-2 video streams, if the system_video_lock_flag is set to '1', then the time base of the ISO/IEC 14496-2 video stream, as defined by vop_time_increment_resolution, shall be locked to the STC and shall be exactly equal to N times system_clock_frequency divided by K, with N and K integers that have a fixed value within each visual object sequence, with K greater than or equal to N.

For ITU-T Rec. H.264 | ISO/IEC 14496-10 video streams, the frequency of the AVC time base is defined by the AVC parameter time_scale. If the system_video_lock_flag is set to '1' for an AVC video stream, then the frequency of the AVC time base shall be locked to the STC and shall be exactly equal to N times system_clock_frequency divided by K, with N and K integers that have a fixed value within each AVC video sequence, with K greater than or equal to N.

$$SCFR = \frac{system\_clock\_frequency}{frame\_rate\_in\_the\_P-STD} \tag{2-25}$$

Nominal frame rate (Hz)	23.976	24	25	29.97	30	50	59.94	60
SCFR	1 126 125	1 125 000	1 080 000	900 900	900 000	540 000	450 450	450 000

The values of the ratio SCFR are exact. The actual frame rate differs slightly from the nominal rate in cases where the nominal rate is 23.976, 29.97, or 59.94 frames per second.

video_bound – The video_bound is a 5-bit integer in the inclusive range from 0 to 16 and is set to a value greater than or equal to the maximum number of video streams in the Program Stream of which the decoding processes are simultaneously active. For the purpose of this subclause, the decoding process of a video stream is active if one of the buffers in the P-STD model is not empty, or if a Presentation Unit is being presented in the P-STD model.

packet_rate_restriction_flag – The packet_rate_restriction_flag is a 1-bit flag. If the CSPS flag is set to '1', the packet_rate_restriction_flag indicates which constraint is applicable to the packet rate, as specified in 2.7.9. If the CSPS flag is set to value of '0', then the meaning of the packet_rate_restriction_flag is undefined.

reserved_bits – This 7-bit field is reserved for future use by ISO/IEC. Until otherwise specified by ITU-T | ISO/IEC it shall have the value '111 1111'.

stream_id – The stream_id is an 8-bit field that indicates the coding and elementary stream number of the stream to which the following P-STD_buffer_bound_scale and P-STD_buffer_size_bound fields refer.

If stream_id equals '1011 1000' the P-STD_buffer_bound_scale and P-STD_buffer_size_bound fields following the stream_id refer to all audio streams in the Program Stream.

If stream_id equals '1011 1001' the P-STD_buffer_bound_scale and P-STD_buffer_size_bound fields following the stream_id refer to all video streams in the Program Stream.

If the stream_id takes on any other value it shall be a byte value greater than or equal to '1011 1100' and shall be interpreted as referring to the stream coding and elementary stream number according to Table 2-22.

Each elementary stream present in the Program Stream shall have its P-STD_buffer_bound_scale and P-STD_buffer_size_bound specified exactly once by this mechanism in each system header.

P-STD_buffer_bound_scale – The P-STD_buffer_bound_scale is a 1-bit field that indicates the scaling factor used to interpret the subsequent P-STD_buffer_size_bound field. If the preceding stream_id indicates an audio stream, P-STD_buffer_bound_scale shall have the value '0'. If the preceding stream_id indicates a video stream, P-STD_buffer_bound_scale shall have the value '1'. For all other stream types, the value of the P-STD_buffer_bound_scale may be either '1' or '0'.

P-STD_buffer_size_bound – The P-STD_buffer_size_bound is a 13-bit unsigned integer defining a value greater than or equal to the maximum P-STD input buffer size, BS_n, over all packets for stream n in the Program Stream. If P-STD_buffer_bound_scale has the value '0', then P-STD_buffer_size_bound measures the buffer size bound in units of 128 bytes. If P-STD_buffer_bound_scale has the value '1', then P-STD_buffer_size_bound measures the buffer size bound in units of 1024 bytes. Thus:

$$if\,(P-STD\_buffer\_bound\_scale == 0)$$
$$BS_n \le P-STD\_buffer\_size\_bound \times 128$$

else:

$$BS_n \le P-STD\_buffer\_size\_bound \times 1024$$

2.5.3.7 Packet layer of Program Stream

The packet layer of the Program Stream is defined by the PES packet layer in 2.4.3.6.

2.5.4 Program Stream map

The Program Stream Map (PSM) provides a description of the elementary streams in the Program Stream and their relationship to one another. When carried in a Transport Stream this structure shall not be modified. The PSM is present as a PES packet when the stream_id value is 0xBC (refer to Table 2-22).

NOTE – This syntax differs from the PES packet syntax described in 2.4.3.6.

Definition for the descriptor() fields may be found in 2.6.

2.5.4.1 Syntax of Program Stream map

See Table 2-41.

Table 2-41 – Program Stream map

Syntax	No. of bits	Mnemonic
program_stream_map() {		
packet_start_code_prefix	**24**	**bslbf**
map_stream_id	**8**	**uimsbf**
program_stream_map_length	**16**	**uimsbf**
current_next_indicator	**1**	**bslbf**
reserved	**2**	**bslbf**
program_stream_map_version	**5**	**uimsbf**
reserved	**7**	**bslbf**
marker_bit	**1**	**bslbf**
program_stream_info_length	**16**	**uimsbf**
for (i = 0; i < N; i++) {		
descriptor()		
}		
elementary_stream_map_length	**16**	**uimsbf**
for (i = 0; i < N1; i++) {		
stream_type	**8**	**uimsbf**
elementary_stream_id	**8**	**uimsbf**
elementary_stream_info_length	**16**	**uimsbf**
for (i = 0; i < N2; i++) {		
descriptor()		
}		
}		
CRC_32	**32**	**rpchof**
}		

2.5.4.2 Semantic definition of fields in Program Stream map

packet_start_code_prefix – The packet_start_code_prefix is a 24-bit code. Together with the map_stream_id that follows it constitutes a packet start code that identifies the beginning of a packet. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001 in hexadecimal).

map_stream_id – This is an 8-bit field whose value shall be 0xBC.

program_stream_map_length – The program_stream_map_length is a 16-bit field indicating the total number of bytes in the program_stream_map immediately following this field. The maximum value of this field is 1018 (0x3FA).

current_next_indicator – This is a 1-bit field, when set to '1' indicates that the Program Stream Map sent is currently applicable. When the bit is set to '0', it indicates that the Program Stream Map sent is not yet applicable and shall be the next table to become valid.

program_stream_map_version – This 5-bit field is the version number of the whole Program Stream Map. The version number shall be incremented by 1 modulo 32 whenever the definition of the Program Stream Map changes. When the current_next_indicator is set to '1', then the program_stream_map_version shall be that of the currently applicable Program Stream Map. When the current_next_indicator is set to '0', then the program_stream_map_version shall be that of the next applicable Program Stream Map.

program_stream_info_length – The program_stream_info_length is a 16-bit field indicating the total length of the descriptors immediately following this field.

marker_bit – A marker_bit is a 1-bit field that has the value '1'.

elementary_stream_map_length – This is a 16-bit field specifying the total length, in bytes, of all elementary stream information in this program stream map. It includes the stream_type, elementary_stream_id, and elementary_stream_info_length fields.

stream_type – This 8-bit field specifies the type of the stream according to Table 2-34. The stream_type field shall only identify elementary streams contained in PES packets. A value of 0x05 is prohibited.

elementary_stream_id – The elementary_stream_id is an 8-bit field indicating the value of the stream_id field in the PES packet headers of PES packets in which this elementary stream is stored.

elementary_stream_info_length – The elementary_stream_info_length is a 16-bit field indicating the length in bytes of the descriptors immediately following this field.

CRC_32 – This is a 32-bit field that contains the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire program stream map.

2.5.5 Program Stream directory

The directory for an entire stream is made up of all the directory data carried by Program Stream Directory packets identified with the directory_stream_id. The syntax for program_stream_directory packets is defined in Table 2-42.

NOTE 1 – This syntax differs from the PES packet syntax described in 2.4.3.6.

Directory entries may be required to reference I-pictures in a video stream as defined in ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 11172-2. If an I-picture that is referenced in a directory entry is preceded by a sequence header with no intervening picture headers, the directory entry shall reference the first byte of the sequence header. If an I-picture that is referenced in a directory entry is preceded by a group of pictures header with no intervening picture headers and no immediately preceding sequence header, the directory entry shall reference the first byte of the group of pictures header. Any other picture that a directory entry references shall be referenced by the first byte of the picture header.

NOTE 2 – It is recommended that I-pictures immediately following a sequence header should be referenced in directory structures so that the directory contains an entry at every point where the decoder may be reset completely.

Directory entries may be required to reference IDR picture or pictures associated with a recovery point SEI message in an AVC video stream. Each such directory entry shall refer to the first byte of an AVC access unit.

Directory references to audio streams as defined in ISO/IEC 13818-3 and ISO/IEC 11172-3 shall be the syncword of the audio frame.

NOTE 3 – It is recommended that the distance between referenced access units not exceed half a second.

Access units shall be referenced in a program_stream_directory packet in the same order that they appear in the bitstream.

2.5.5.1 Syntax of Program Stream directory packet

See Table 2-42.

Table 2-42 – Program Stream directory packet

Syntax	No. of bits	Mnemonic
directory_PES_packet(){		
packet_start_code_prefix	**24**	**bslbf**
directory_stream_id	**8**	**uimsbf**
PES_packet_length	**16**	**uimsbf**
number_of_access_units	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
prev_directory_offset[44..30]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
prev_directory_offset[29..15]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
prev_directory_offset[14..0]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
next_directory_offset[44..30]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
next_directory_offset[29..15]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
next_directory_offset[14..0]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**

Table 2-42 – Program Stream directory packet

Syntax	No. of bits	Mnemonic
for (i = 0; i < number_of_access_units; i++) {		
packet_stream_id	**8**	**uimsbf**
PES_header_position_offset_sign	**1**	**tcimsbf**
PES_header_position_offset[43..30]	**14**	**uimsbf**
marker_bit	**1**	**bslbf**
PES_header_position_offset[29..15]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
PES_header_position_offset[14..0]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
reference_offset	**16**	**uimsbf**
marker_bit	**1**	**bslbf**
reserved	**3**	**bslbf**
PTS[32..30]	**3**	**uimsbf**
marker_bit	**1**	**bslbf**
PTS[29..15]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
PTS[14..0]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
bytes_to_read[22..8]	**15**	**uimsbf**
marker_bit	**1**	**bslbf**
bytes_to_read[7..0]	**8**	**uimsbf**
marker_bit	**1**	**bslbf**
intra_coded_indicator	**1**	**bslbf**
coding_parameters_indicator	**2**	**bslbf**
reserved	**4**	**bslbf**
}		
}		

2.5.5.2 Semantic definition of fields in Program Stream directory

packet_start_code_prefix – The packet_start_code_prefix is a 24-bit code. Together with the stream_id that follows, it constitutes a packet start code that identifies the beginning of a packet. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001 in hexadecimal).

directory_stream_id – This 8-bit field shall have a value '1111 1111' (0xFF).

PES_packet_length – The PES_packet_length is a 16-bit field indicating the total number of bytes in the program_stream_directory immediately following this field (refer to Table 2-22).

number_of_access_units – This 15-bit field is the number of access_units that are referenced in this Directory PES packet.

prev_directory_offset – This 45-bit unsigned integer gives the byte address offset of the first byte of the packet start code of the previous Program Stream Directory packet. This address offset is relative to the first byte of the start code of the packet which contains this previous_directory_offset field. The value '0' indicates that there is no previous Program Stream Directory packet.

next_directory_offset – This 45-bit unsigned integer gives the byte address offset of the first byte of the packet start code of the next Program Stream Directory packet. This address offset is relative to the first byte of the start code of the packet which contains this next_directory_offset field. The value '0' indicates that there is no next Program Stream Directory packet.

packet_stream_id – This 8-bit field is the stream_id of the elementary stream that contains the access unit referenced by this directory entry.

PES_header_position_offset_sign – This 1-bit field is the arithmetic sign for the PES_header_position_offset described immediately following. A value of '0' indicates that the PES_header_position_offset is a positive offset. A value of '1' indicates that the PES_header_position_offset is a negative offset.

PES_header_position_offset – This 44-bit unsigned integer gives the byte offset address of the first byte of the PES packet containing the access unit referenced. The offset address is relative to the first byte of the start-code of the packet containing this PES_header_position_offset field. The value '0' indicates that no access unit is referenced.

reference_offset – This 16-bit field is an unsigned integer indicating the position of the first byte of the referenced access unit, measured in bytes relative to the first byte of the PES packet containing the first byte of the referenced access unit.

PTS (presentation_time_stamp) – This 33-bit field is the PTS of the access unit that is referenced. The semantics of the coding of the PTS field are as described in 2.4.3.6.

bytes_to_read – This 23-bit unsigned integer is the number of bytes in the Program Stream after the byte indicated by reference_offset that are needed to decode the access unit completely. This value includes any bytes multiplexed at the systems layer including those containing information from other streams.

intra_coded_indicator – This is a 1-bit flag. When set to '1' it indicates that the referenced access unit is not predictively coded. This is independent of other coding parameters that might be needed to decode the access unit. For example, this field shall be coded as '1' for video Intra frames, whereas for 'P' and 'B' frames this bit shall be coded as '0'. For all PES packets containing data which is not from an ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream, this field is undefined (see Table 2-43).

Table 2-43 – Intra_coded indicator

Value	Meaning
0	Not Intra
1	Intra

coding_parameters_indicator – This 2-bit field is used to indicate the location of coding parameters that are needed to decode the access units referenced. For example, this field can be used to determine the location of quantization matrices for video frames.

Table 2-44 – Coding_parameters indicator

Value	Meaning
00	All coding parameters are set to their default values
01	All coding parameters are set in this access unit, at least one of them is not set to a default
10	Some coding parameters are set in this access unit
11	No coding parameters are coded in this access unit

2.6 Program and program element descriptors

Program and program element descriptors are structures which may be used to extend the definitions of programs and program elements. All descriptors have a format which begins with an 8-bit tag value. The tag value is followed by an 8-bit descriptor length and data fields.

2.6.1 Semantic definition of fields in program and program element descriptors

The following semantics apply to the descriptors defined in 2.6.2 through 2.6.34.

descriptor_tag – The descriptor_tag is an 8-bit field which identifies each descriptor.

Table 2-45 provides the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 defined, ITU-T Rec. H.222.0 | ISO/IEC 13818-1 reserved, and user available descriptor tag values. An 'X' in the TS or PS columns indicates the applicability of the descriptor to either the Transport Stream or Program Stream respectively. Note that the meaning of fields in a descriptor may depend on which stream it is used in. Each case is specified in the descriptor semantics below.

descriptor_length – The descriptor_length is an 8-bit field specifying the number of bytes of the descriptor immediately following descriptor_length field.

Table 2-45 – Program and program element descriptors

descriptor_tag	TS	PS	Identification
0	n/a	n/a	Reserved
1	n/a	n/a	Reserved
2	X	X	video_stream_descriptor
3	X	X	audio_stream_descriptor
4	X	X	hierarchy_descriptor
5	X	X	registration_descriptor
6	X	X	data_stream_alignment_descriptor
7	X	X	target_background_grid_descriptor
8	X	X	video_window_descriptor
9	X	X	CA_descriptor
10	X	X	ISO_639_language_descriptor
11	X	X	system_clock_descriptor
12	X	X	multiplex_buffer_utilization_descriptor
13	X	X	copyright_descriptor
14	X		maximum_bitrate_descriptor
15	X	X	private_data_indicator_descriptor
16	X	X	smoothing_buffer_descriptor
17	X		STD_descriptor
18	X	X	IBP_descriptor
19-26	X		Defined in ISO/IEC 13818-6
27	X	X	MPEG-4_video_descriptor
28	X	X	MPEG-4_audio_descriptor
29	X	X	IOD_descriptor
30	X		SL_descriptor
31	X	X	FMC_descriptor
32	X	X	external_ES_ID_descriptor
33	X	X	MuxCode_descriptor
34	X	X	FmxBufferSize_descriptor
35	X		multiplexbuffer_descriptor
36	X	X	content_labeling_descriptor
37	X	X	metadata_pointer_descriptor
38	X	X	metadata_descriptor
39	X	X	metadata_STD_descriptor
40	X	X	AVC video descriptor
41	X	X	IPMP_descriptor (defined in ISO/IEC 13818-11, MPEG-2 IPMP)
42	X	X	AVC timing and HRD descriptor
43	X	X	MPEG-2_AAC_audio_descriptor
44	X	X	FlexMuxTiming_descriptor
45-63	n/a	n/a	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Reserved
64-255	n/a	n/a	User Private

2.6.2 Video stream descriptor

The video stream descriptor provides basic information which identifies the coding parameters of a video elementary stream as described in ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 (see Table 2-46).

Table 2-46 – Video stream descriptor

Syntax	No. of bits	Mnemonic
video_stream_descriptor(){		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
multiple_frame_rate_flag	**1**	**bslbf**
frame_rate_code	**4**	**uimsbf**
MPEG_1_only_flag	**1**	**bslbf**
constrained_parameter_flag	**1**	**bslbf**
still_picture_flag	**1**	**bslbf**
if (MPEG_1_only_flag = = '0'){		
profile_and_level_indication	**8**	**uimsbf**
chroma_format	**2**	**uimsbf**
frame_rate_extension_flag	**1**	**bslbf**
Reserved	**5**	**bslbf**
}		
}		

2.6.3 Semantic definitions of fields in video stream descriptor

multiple_frame_rate_flag – This 1-bit field when set to '1' indicates that multiple frame rates may be present in the video stream. When set to a value of '0' only a single frame rate is present.

frame_rate_code – This is a 4-bit field as defined in 6.3.3 of ITU-T Rec. H.262 | ISO/IEC 13818-2, except that when the multiple_frame_rate_flag is set to a value of '1' the indication of a particular frame rate also permits certain other frame rates to be present in the video stream, as specified in Table 2-47:

Table 2-47 – Frame rate code

Coded as	Also includes
23.976	
24.0	23.976
25.0	
29.97	23.976
30.0	23.976 24.0 29.97
50.0	25.0
59.94	23.976 29.97
60.0	23.976 24.0 29.97 30.0 59.94

MPEG_1_only_flag – This is a 1-bit field which when set to '1' indicates that the video stream contains only ISO/IEC 11172-2 data. If set to '0' the video stream may contain both ITU-T Rec. H.262 | ISO/IEC 13818-2 video data and constrained parameter ISO/IEC 11172-2 video data.

constrained_parameter_flag – This is a 1-bit field which when set to '1' indicates that the video stream shall not contain unconstrained ISO/IEC 11172-2 video data. If this field is set to '0' the video stream may contain both constrained parameters and unconstrained ISO/IEC 11172-2 video streams. If the MPEG_1_only_flag is set to '0', the constrained_parameter_flag shall be set to '1'.

still_picture_flag – This is a 1-bit field, which when set to '1' indicates that the video stream contains only still pictures. If the bit is set to '0' then the video stream may contain either moving or still picture data.

profile_and_level_indication – This 8-bit field is coded in the same manner as the profile_and_level_indication fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream. The value of this field indicates a profile and level that is equal to or higher than any profile and level in any sequence in the associated video stream. For the purposes of this

subclause, an ISO/IEC 11172-2 constrained parameter stream is considered to a be a Main Profile at Low Level stream (MP @ LL).

chroma_format – This 2-bit field is coded in the same manner as the chroma_format fields in the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream. The value of this field shall be at least equal to or higher than the value of the chroma_format field in any video sequence of the associated video stream. For the purposes of this subclause, an ISO/IEC 11172-2 video stream is considered to have chroma_format field with the value '01', indicating 4:2:0.

frame_rate_extension_flag – This is a 1-bit flag which when set to '1' indicates that either or both the frame_rate_extension_n and the frame_rate_extension_d fields are non-zero in any video sequences of the ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream. For the purposes of this subclause, an ISO/IEC 11172-2 video stream is constrained to have both fields set to zero.

2.6.4 Audio stream descriptor

The audio stream descriptor provides basic information which identifies the coding version of an audio elementary stream as described in ISO/IEC 13818-3 or ISO/IEC 11172-3 (see Table 2-48).

Table 2-48 – Audio stream descriptor

Syntax	No. of bits	Mnemonic
audio_stream_descriptor(){		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
free_format_flag	**1**	**bslbf**
ID	**1**	**bslbf**
layer	**2**	**bslbf**
variable_rate_audio_indicator	**1**	**bslbf**
reserved	**3**	**bslbf**
}		

2.6.5 Semantic definition of fields in audio stream descriptor

free_format_flag – This 1-bit field when set to '1' indicates that the audio stream may contain one or more audio frames with the bitrate_index set to '0000'. If set to '0', then the bitrate_index is not '0000' (refer to 2.4.2.3 of ISO/IEC 13818-3) in any audio frame of the audio stream.

ID – This 1-bit field when set to '1' indicates that the ID field is set to '1' in each audio frame in the audio stream (refer to 2.4.2.3 of ISO/IEC 13818-3).

layer – This 2-bit field is coded in the same manner as the layer field in the ISO/IEC 13818-3 or ISO/IEC 11172-3 audio streams (refer to 2.4.2.3 of ISO/IEC 13818-3). The layer indicated in this field shall be equal to or higher than the highest layer specified in any audio frame of the audio stream.

variable_rate_audio_indicator – This 1-bit flag, when set to '0' indicates that the encoded value of the bit rate field shall not change in consecutive audio frames which are intended to be presented without discontinuity.

2.6.6 Hierarchy descriptor

The hierarchy descriptor provides information to identify the program elements containing components of hierarchically-coded video, audio, and private streams. (See Table 2-49.)

Table 2-49 – Hierarchy descriptor

Syntax	No. of bits	Mnemonic
hierarchy_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
reserved	**4**	**bslbf**
hierarchy_type	**4**	**uimsbf**
reserved	**2**	**bslbf**
hierarchy_layer_index	**6**	**uimsbf**
reserved	**2**	**bslbf**
hierarchy_embedded_layer_index	**6**	**uimsbf**
reserved	**2**	**bslbf**
hierarchy_channel	**6**	**uimsbf**
}		

2.6.7 Semantic definition of fields in hierarchy descriptor

hierarchy_type – The hierarchical relation between the associated hierarchy layer and its hierarchy embedded layer is defined in Table 2-50.

hierarchy_layer_index – The hierarchy_layer_index is a 6-bit field that defines a unique index of the associated program element in a table of coding layer hierarchies. Indices shall be unique within a single program definition.

hierarchy_embedded_layer_index – The hierarchy_embedded_layer_index is a 6-bit field that defines the hierarchy table index of the program element that needs to be accessed before decoding of the elementary stream associated with this hierarchy_descriptor. This field is undefined if the hierarchy_type value is 15 (base layer).

hierarchy_channel – The hierarchy_channel is a 6-bit field that indicates the intended channel number for the associated program element in an ordered set of transmission channels. The most robust transmission channel is defined by the lowest value of this field with respect to the overall transmission hierarchy definition.

NOTE – A given hierarchy_channel may at the same time be assigned to several program elements.

Table 2-50 – Hierarchy_type field values

Value	Description
0	Reserved
1	Spatial Scalability
2	SNR Scalability
3	Temporal Scalability
4	Data partitioning
5	Extension bitstream
6	Private Stream
7	Multi-view Profile
8-14	Reserved
15	Base layer

2.6.8 Registration descriptor

The registration_descriptor provides a method to uniquely and unambiguously identify formats of private data (see Table 2-51).

Table 2-51 – Registration descriptor

Syntax	No. of bits	Identifier
registration_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
format_identifier	**32**	**uimsbf**
for (i = 0; i < N; i++){		
additional_identification_info	**8**	**bslbf**
}		
}		

2.6.9 Semantic definition of fields in registration descriptor

format_identifier – The format_identifier is a 32-bit value obtained from a Registration Authority as designated by ISO/IEC JTC 1/SC 29.

additional_identification_info – The meaning of additional_identification_info bytes, if any, are defined by the assignee of that format_identifier, and once defined they shall not change.

2.6.10 Data stream alignment descriptor

The data stream alignment descriptor describes which type of alignment is present in the associated elementary stream. If the data_alignment_indicator in the PES packet header is set to '1' and the descriptor is present, alignment – as specified in this descriptor – is required (see Table 2-52).

Table 2-52 – Data stream alignment descriptor

Syntax	No. of bits	Mnemonic
data_stream_alignment_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
alignment_type	**8**	**uimsbf**
}		

2.6.11 Semantics of fields in data stream alignment descriptor

alignment_type – Table 2-53 describes the alignment type for ISO/IEC 11172-2 video, ITU-T Rec. H.262 | ISO/IEC 13818-2 video, or ISO/IEC 14496-2 visual streams when the data_alignment_indicator in the PES packet header has a value of '1'. For these video streams, the first PES_packet_data_byte following the PES header shall be the first byte of a start code of the type indicated in Table 2-53. At the beginning of a video sequence, the alignment shall occur at the start code of the first sequence header.

NOTE – Specifying alignment type '01' from Table 2-53 does not preclude the alignment from beginning at a GOP or SEQ header.

The definition of an access unit is given in 2.1.1.

Table 2-53 – Video stream alignment values

Alignment type	Description
00	Reserved
01	Slice, or video access unit
02	Video access unit
03	GOP, or SEQ
04	SEQ
05-FF	Reserved

Table 2-54 describes the alignment type for ITU-T Rec. H.264 | ISO/IEC 14496-10 video when the data_alignment_indicator in the PES packet header has a value of '1'. In this case the first PES_packet_data_byte following the PES header shall be the first byte of an AVC access unit or the first byte of an AVC slice, as signalled by the alignment_type value.

Table 2-54 – AVC video stream alignment values

Alignment type	Description
00	Reserved
01	AVC slice or AVC access unit
02	AVC access unit
03-FF	Reserved

Table 2-55 describes the audio alignment type when the data_alignment_indicator in the PES packet header has a value of '1'. In this case the first PES_packet_data_byte following the PES header is the first byte of an audio sync word.

Table 2-55 – Audio stream alignment values

Alignment type	Description
00	Reserved
01	Sync word
02-FF	Reserved

2.6.12 Target background grid descriptor

It is possible to have one or more video streams which, when decoded, are not intended to occupy the full display area (e.g., a monitor). The combination of target_background_grid_descriptor and video_window_descriptors allows the display of these video windows in their desired locations. The target_background_grid_descriptor is used to describe a grid of unit pixels projected on to the display area. The video_window_descriptor is then used to describe, for the associated stream, the location on the grid at which the top left pixel of the display window or display rectangle of the video presentation unit should be displayed. This is represented in Figure 2-3.

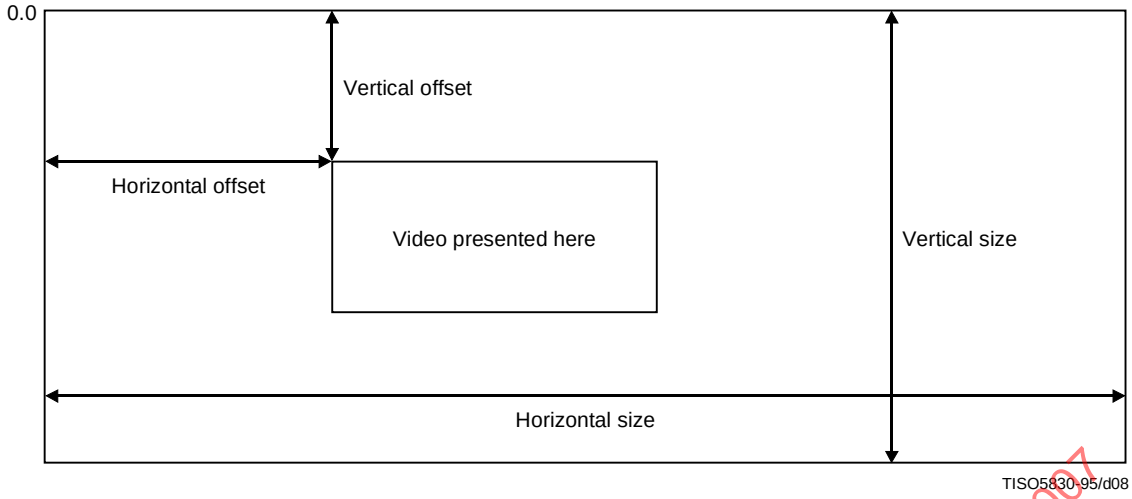

Figure 2-3 – Target background grid descriptor display area

2.6.13 Semantics of fields in target background grid descriptor

horizontal_size – The horizontal size of the target background grid in pixels.

vertical_size – The vertical size of the target background grid in pixels.

aspect_ratio_information – Specifies the sample aspect ratio or display aspect ratio of the target background grid. Aspect_ratio_information is defined in ITU-T Rec. H.262 | ISO/IEC 13818-2 (see Table 2-56).

Table 2-56 – Target background grid descriptor

Syntax	No. of bits	Mnemonic
target_background_grid_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
horizontal_size	**14**	**uimsbf**
vertical_size	**14**	**uimsbf**
aspect_ratio_information	**4**	**uimsbf**
}		

2.6.14 Video window descriptor

The video window descriptor is used to describe the window characteristics of the associated video elementary stream. Its values reference the target background grid descriptor for the same stream. Also see target_background_grid_descriptor in 2.6.12 (see Table 2-57).

Table 2-57 – Video window descriptor

Syntax	No. of bits	Mnemonic
video_window_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
horizontal_offset	**14**	**uimsbf**
vertical_offset	**14**	**uimsbf**
window_priority	**4**	**uimsbf**
}		

2.6.15 Semantic definition of fields in video window descriptor

horizontal_offset – The value indicates the horizontal position of the top left pixel of the current video display window or display rectangle if indicated in the picture display extension on the target background grid for display as defined in the target_background_grid_descriptor. The top left pixel of the video window shall be one of the pixels of the target background grid (refer to Figure 2-3).

vertical_offset – The value indicates the vertical position of the top left pixel of the current video display window or display rectangle if indicated in the picture display extension on the target background grid for display as defined in the target_background_grid_descriptor. The top left pixel of the video window shall be one of the pixels of the target background grid (refer to Figure 2-3).

window_priority – The value indicates how windows overlap. A value of 0 being lowest priority and a value of 15 is the highest priority, i.e., windows with priority 15 are always visible.

2.6.16 Conditional access descriptor

The conditional access descriptor is used to specify both system-wide conditional access management information such as EMMs and elementary stream-specific information such as ECMs. It may be used in both the TS_program_map_section (refer to 2.4.4.8) and the program_stream_map (refer to 2.5.3). If any elementary stream is scrambled, a CA descriptor shall be present for the program containing that elementary stream. If any system-wide conditional access management information exists within a Transport Stream, a CA descriptor shall be present in the conditional access table.

When the CA descriptor is found in the TS_program_map_section (table_id = 0x02), the CA_PID points to packets containing program related access control information, such as ECMs. Its presence as program information indicates applicability to the entire program. In the same case, its presence as extended ES information indicates applicability to the associated program element. Provision is also made for private data.

When the CA descriptor is found in the CA_section (table_id = 0x01), the CA_PID points to packets containing system-wide and/or access control management information, such as EMMs.

The contents of the Transport Stream packets containing conditional access information are privately defined (see Table 2-58).

Table 2-58 – Conditional access descriptor

Syntax	No. of bits	Mnemonic
CA_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
CA_system_ID	**16**	**uimsbf**
reserved	**3**	**bslbf**
CA_PID	**13**	**uimsbf**
for (i = 0; i < N; i++) {		
private_data_byte	**8**	**uimsbf**
}		
}		

2.6.17 Semantic definition of fields in conditional access descriptor

CA_system_ID – This is a 16-bit field indicating the type of CA system applicable for either the associated ECM and/or EMM streams. The coding of this is privately defined and is not specified by ITU-T | ISO/IEC.

CA_PID – This is a 13-bit field indicating the PID of the Transport Stream packets which shall contain either ECM or EMM information for the CA systems as specified with the associated CA_system_ID. The contents (ECM or EMM) of the packets indicated by the CA_PID is determined from the context in which the CA_PID is found, i.e., a TS_program_map_section or the CA table in the Transport Stream, or the stream_id field in the Program Stream.

In Transport Streams, the presence of PID 0x03 indicates that there is IPMP as described in ISO/IEC 13818-11 used by components in the Transport Stream. In Program Streams, the presence of stream_ID_extension value 0x00 indicates that IPMP as described in ISO/IEC 13818-11 is used by components in the Program Stream. Within a given ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, components could use both IPMP as described in ISO/IEC 13818-11 as well as CA as defined in ISO/IEC 13818-1:2006. Compatibility between the two schemes is described in ISO/IEC 13818-11.

2.6.18 ISO 639 language descriptor

The language descriptor is used to specify the language of the associated program element (see Table 2-59).

Table 2-59 – ISO 639 language descriptor

Syntax	No. of bits	Mnemonic
ISO_639_language_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
for (i = 0; i < N; i++) {		
ISO_639_language_code	**24**	**bslbf**
audio_type	**8**	**bslbf**
}		
}		

2.6.19 Semantic definition of fields in ISO 639 language descriptor

ISO_639_language_code – Identifies the language or languages used by the associated program element. The ISO_639_language_code contains a 3-character code as specified by ISO 639, Part 2. Each character is coded into 8 bits according to ISO/IEC 8859-1 and inserted in order into this 24-bit field. In the case of multilingual audio streams the sequence of ISO_639_language_code fields shall reflect the content of the audio stream.

audio_type – The audio_type is an 8-bit field which specifies the type of stream defined in Table 2-60.

Table 2-60 – Audio type values

Value	Description
0x00	Undefined
0x01	Clean effects
0x02	Hearing impaired
0x03	Visual impaired commentary
0x04-0x7F	User Private
0x80-0xFF	Reserved

clean effects – This field indicates that the referenced program element has no language.

hearing impaired – This field indicates that the referenced program element is prepared for the hearing impaired.

visual_impaired_commentary – This field indicates that the referenced program element is prepared for the visually impaired viewer.

2.6.20 System clock descriptor

This descriptor conveys information about the system clock that was used to generate the timestamps.

If an external clock reference was used, the external_clock_reference_indicator may be set to '1'. The decoder optionally may use the same external reference if it is available.

If the system clock is more accurate than the 30-ppm accuracy required, then the accuracy of the clock can be communicated by encoding it in the clock_accuracy fields. The clock frequency accuracy is:

$$clock\_accuracy\_integer \times 10^{-clock\_accuracy\_exponent}\ ppm \qquad (2\text{-}26)$$

If clock_accuracy_integer is set to '0', then the system clock accuracy is 30 ppm. When the external_clock_reference_indicator is set to '1', the clock accuracy pertains to the external reference clock (see Table 2-61).

Table 2-61 – System clock descriptor

Syntax	No. of bits	Mnemonic
system_clock_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
external_clock_reference_indicator	**1**	**bslbf**
reserved	**1**	**bslbf**
clock_accuracy_integer	**6**	**uimsbf**
clock_accuracy_exponent	**3**	**uimsbf**
reserved	**5**	**bslbf**
}		

2.6.21 Semantic definition of fields in system clock descriptor

external_clock_reference_indicator – This is a 1-bit indicator. When set to '1', it indicates that the system clock has been derived from an external frequency reference that may be available at the decoder.

clock_accuracy_integer – This is a 6-bit integer. Together with the clock_accuracy_exponent, it gives the fractional frequency accuracy of the system clock in parts per million.

clock_accuracy_exponent – This is a 3-bit integer. Together with the clock_accuracy_integer, it gives the fractional frequency accuracy of the system clock in parts per million.

2.6.22 Multiplex buffer utilization descriptor

The multiplex buffer utilization descriptor provides bounds on the occupancy of the STD multiplex buffer. This information is intended for devices such as remultiplexers, which may use this information to support a desired re-multiplexing strategy (see Table 2-62).

Table 2-62 – Multiplex buffer utilization descriptor

Syntax	No. of bits	Mnemonic
Multiplex_buffer_utilization_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
bound_valid_flag	**1**	**bslbf**
LTW_offset_lower_bound	**15**	**uimsbf**
reserved	**1**	**bslbf**
LTW_offset_upper_bound	**15**	**uimsbf**
}		

2.6.23 Semantic definition of fields in multiplex buffer utilization descriptor

bound_valid_flag – A value of '1' indicates that the LTW_offset_lower_bound and the LTW_offset_upper_bound fields are valid.

LTW_offset_lower_bound – This 15-bit field is defined only if the bound_valid flag has a value of '1'. When defined, this field has the units of (27 MHz/300) clock periods, as defined for the LTW_offset (refer to 2.4.3.4). The LTW_offset_lower_bound represents the lowest value that any LTW_offset field would have, if that field were coded in every packet of the stream or streams referenced by this descriptor. Actual LTW_offset fields may or may not be coded in the bitstream when the multiplex buffer utilization descriptor is present. This bound is valid until the next occurrence of this descriptor.

LTW_offset_upper_bound – This 15-bit field is defined only if the bound_valid has a value of '1'. When defined, this field has the units of (27 MHz/300) clock periods, as defined for the LTW_offset (refer to 2.4.3.4). The LTW_offset_upper_bound represents the largest value that any LTW_offset field would have, if that field were coded in every packet of the stream or streams referenced by this descriptor. Actual LTW_offset fields may or may not be

coded in the bitstream when the multiplex buffer utilization descriptor is present. This bound is valid until the next occurrence of this descriptor.

2.6.24 Copyright descriptor

The copyright_descriptor provides a method to enable audiovisual works identification. This copyright_descriptor applies to programs or program elements within programs (see Table 2-63).

Table 2-63 – Copyright descriptor

Syntax	No. of bits	Identifier
copyright_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
copyright_identifier	**32**	**uimsbf**
for (i = 0; i < N; i++){		
additional_copyright_info	**8**	**bslbf**
}		
}		

2.6.25 Semantic definition of fields in copyright descriptor

copyright_identifier – This field is a 32-bit value obtained from the Registration Authority.

additional_copyright_info – The meaning of additional_copyright_info bytes, if any, are defined by the assignee of that copyright_identifier, and once defined, they shall not change.

2.6.26 Maximum bitrate descriptor

See Table 2-64.

Table 2-64 – Maximum bitrate descriptor

Syntax	No. of bits	Identifier
maximum_bitrate_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
reserved	**2**	**bslbf**
maximum_bitrate	**22**	**uimsbf**
}		

2.6.27 Semantic definition of fields in maximum bitrate descriptor

maximum_bitrate – The maximum bitrate is coded as a 22-bit positive integer in this field. The value indicates an upper bound of the bitrate, including transport overhead, that will be encountered in this program element or program. The value of maximum_bitrate is expressed in units of 50 bytes/second. The maximum_bitrate_descriptor is included in the Program Map Table (PMT). Its presence as extended program information indicates applicability to the entire program. Its presence as ES information indicates applicability to the associated program element.

2.6.28 Private data indicator descriptor

See Table 2-65.

Table 2-65 – Private data indicator descriptor

Syntax	No. of bits	Identifier
private_data_indicator_descriptor() {		
descriptor_tag	**38**	**uimsbf**
descriptor_length	**38**	**uimsbf**
private_data_indicator	**32**	**uimsbf**
}		

2.6.29 Semantic definition of fields in Private data indicator descriptor

private_data_indicator – The value of the private_data_indicator is private and shall not be defined by ITU-T | ISO/IEC.

2.6.30 Smoothing buffer descriptor

This descriptor is optional and conveys information about the size of a smoothing buffer, SB_n, associated with this descriptor, and the associated leak rate out of that buffer, for the program element(s) that it refers to.

In the case of Transport Streams, bytes of Transport Stream packets of the associated program element(s) present in the Transport Stream are input to a buffer SB_n of size given by sb_size, at the time defined by equation 2-4.

In the case of Program Streams, bytes of all PES packets of the associated elementary streams, are input to a buffer SB_n of size given by sb_size, at the time defined by equation 2-21.

When there is data present in this buffer, bytes are removed from this buffer at a rate defined by sb_leak_rate. The buffer, SB_n shall never overflow. During the continuous existence of a program, the value of the elements of the Smoothing Buffer descriptor of the different program element(s) in the program, shall not change.

The meaning of the smoothing buffer_descriptor is only defined when it is included in the PMT or the Program Stream Map.

If, in the case of a Transport Stream, it is present in the ES info in the Program Map Table, all Transport Stream packets of the PID of that program element enter the smoothing buffer.

If, in the case of a Transport Stream, it is present in the program information, the following Transport Stream packets enter the smoothing buffer:

- all Transport Stream packets of all PIDs listed as elementary_PIDs in the extended program information as well as;
- all Transport Stream packets of the PID which is equal to the PMT_PID of this section;
- all Transport Stream packets of the PCR_PID of the program.

All bytes that enter the associated buffer also exit it.

At any given time there shall be at most one descriptor referring to any individual program element and at most one descriptor referring to the program in its entirety.

Table 2-66 – Smoothing buffer descriptor

Syntax	No. of bits	Mnemonic
smoothing_buffer_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
reserved	**2**	**bslbf**
sb_leak_rate	**22**	**uimsbf**
reserved	**2**	**bslbf**
sb_size	**22**	**uimsbf**
}		

2.6.31 Semantic definition of fields in smoothing buffer descriptor

sb_leak_rate – This 22-bit field is coded as a positive integer. Its contents indicate the value of the leak rate out of the SB_n buffer for the associated elementary stream or other data in units of 400 bits/s.

sb_size – This 22-bit field is coded as a positive integer. Its contents indicate the value of the size of the multiplexing buffer smoothing buffer SB_n for the associated elementary stream or other data in units of 1 byte (see Table 2-66).

2.6.32 STD descriptor

This descriptor is optional and applies only to the T-STD model and to ITU-T Rec. H.262 | ISO/IEC 13818-2 video elementary streams, and is used as specified in 2.4.2. This descriptor does not apply to Program Streams (see Table 2-67).

Table 2-67 – STD descriptor

Syntax	No. of bits	Mnemonic
STD_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
reserved	**7**	**bslbf**
leak_valid_flag	**1**	**bslbf**
}		

2.6.33 Semantic definition of fields in STD descriptor

leak_valid_flag – The leak_valid_flag is a 1-bit flag. When set to '1', the transfer of data from the buffer MB_n to the buffer EB_n in the T-STD uses the leak method as defined in 2.4.2.3. If this flag has a value equal to '0', and the vbv_delay fields present in the associated video stream do not have the value 0xFFFF, the transfer of data from the buffer MB_n to the buffer EB_n uses the vbv_delay method as defined in 2.4.2.3.

2.6.34 IBP descriptor

This optional descriptor provides information about some characteristics of the sequence of frame types in an ISO/IEC 11172-2, ITU-T Rec. H.262 | ISO/IEC 13818-2, or ISO/IEC 14496-2 video stream (see Table 2-68).

Table 2-68 – IBP descriptor

Syntax	No. of bits	Mnemonic
ibp_descriptor() {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
closed_gop_flag	**1**	**uimsbf**
identical_gop_flag	**1**	**uimsbf**
max_gop-length	**14**	**uimsbf**
}		

2.6.35 Semantic definition of fields in IBP descriptor

closed_gop_flag – This 1-bit flag when set to '1' indicates that a group of pictures header is encoded before every I-frame and that the closed_gop flag is set to '1' in all group of pictures headers in the video sequence.

identical_gop_flag – This 1-bit flag when set to '1' indicates that the number of P-frames and B-frames between I-frames, and the picture coding types and sequence of picture types between I-pictures is the same throughout the sequence, except possibly for the pictures up to the second I-picture.

max_gop_length – This 14-bit unsigned integer indicates the maximum number of the coded pictures between any two consecutive I-pictures in the sequence. The value of '0' is forbidden.

2.6.36 MPEG-4 video descriptor

For individual ISO/IEC 14496-2 streams directly carried in PES packets, as defined in 2.11.2, the MPEG-4 video descriptor provides basic information for identifying the coding parameters of such visual elementary streams. The MPEG-4 video descriptor does not apply to ISO/IEC 14496-2 streams encapsulated in SL-packets and in FlexMux packets, as defined in 2.11.3.

Table 2-69 – MPEG-4 video descriptor

Syntax	No. of bits	Mnemonic
MPEG-4_video_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
MPEG-4_visual_profile_and_level	**8**	**uimsbf**
}		

2.6.37 Semantic definition of fields in MPEG-4 video descriptor

MPEG-4_video_profile_and_level – This 8-bit field shall identify the profile and level of the ISO/IEC 14496-2 video stream. This field shall be coded with the same value as the profile_and_level_indication field in the Visual Object Sequence Header in the associated ISO/IEC 14496-2 stream.

2.6.38 MPEG-4 audio descriptor

For individual ISO/IEC 14496-3 streams directly carried in PES packets, as defined in 2.11.2, the MPEG-4 audio descriptor provides basic information for identifying the coding parameters of such audio elementary streams. The MPEG-4 audio descriptor does not apply to ISO/IEC 14496-3 streams encapsulated in SL-packets and in FlexMux packets, as defined in 2.11.3.

Table 2-70 – MPEG-4 audio descriptor

Syntax	No. of bits	Mnemonic
MPEG-4_audio_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
MPEG-4_audio_profile_and_level	**8**	**uimsbf**
}		

2.6.39 Semantic definition of fields in MPEG-4 audio descriptor

MPEG-4_audio_profile_and_level – This 8-bit field shall identify the profile and level of the ISO/IEC 14496-3 audio stream corresponding to the Table 2-71.

Table 2-71 – MPEG-4_audio_profile_and_level assignment values

Value	Description
0x00-0x0F	Reserved
0x10	Main profile, level 1
0x11	Main profile, level 2
0x12	Main profile, level 3
0x13	Main profile, level 4
0x14-0x17	Reserved
0x18	Scalable Profile, level 1
0x19	Scalable Profile, level 2
0x1A	Scalable Profile, level 3
0x1B	Scalable Profile, level 4
0x1C-0x1F	Reserved

Table 2-71 – MPEG-4_audio_profile_and_level assignment values

Value	Description
0x20	Speech profile, level 1
0x21	Speech profile, level 2
0x22-0x27	Reserved
0x28	Synthesis profile, level 1
0x29	Synthesis profile, level 2
0x2A	Synthesis profile, level 3
0x2B-0x2F	Reserved
0x30	High quality audio profile, level 1
0x31	High quality audio profile, level 2
0x32	High quality audio profile, level 3
0x33	High quality audio profile, level 4
0x34	High quality audio profile, level 5
0x35	High quality audio profile, level 6
0x36	High quality audio profile, level 7
0x37	High quality audio profile, level 8
0x38	Low delay audio profile, level 1
0x39	Low delay audio profile, level 2
0x3A	Low delay audio profile, level 3
0x3B	Low delay audio profile, level 4
0x3C	Low delay audio profile, level 5
0x3D	Low delay audio profile, level 6
0x3E	Low delay audio profile, level 7
0x3F	Low delay audio profile, level 8
0x40	Natural audio profile, level 1
0x41	Natural audio profile, level 2
0x42	Natural audio profile, level 3
0x43	Natural audio profile, level 4
0x44-0x47	Reserved
0x48	Mobile audio internetworking profile, level 1
0x49	Mobile audio internetworking profile, level 2
0x4A	Mobile audio internetworking profile, level 3
0x4B	Mobile audio internetworking profile, level 4
0x4C	Mobile audio internetworking profile, level 5
0x4D	Mobile audio internetworking profile, level 6
0x4E-0x4F	Reserved
0x50	AAC profile, level 1
0x51	AAC profile, level 2
0x52	AAC profile, level 4
0x53	AAC profile, level 5
0x54-0x57	Reserved
0x58	High efficiency AAC profile, level 2
0x59	High efficiency AAC profile, level 3
0x5A	High efficiency AAC profile, level 4
0x5B	High efficiency AAC profile, level 5
0x5C-0xFF	Reserved

2.6.40 IOD descriptor

The IOD descriptor encapsulates the InitialObjectDescriptor structure. An initial object descriptor allows access to a set of ISO/IEC 14496 streams by identifying the ES_ID values of the ISO/IEC 14496-1 scene description and object descriptor streams. Both the scene description stream and the object descriptor stream contain further information about the ISO/IEC 14496 streams that are part of the scene. See Annex R for a description of the content access procedure. The InitialObjectDescriptor is specified in 8.6.3 of ISO/IEC 14496-1.

Within a Transport Stream, the IOD descriptor shall be conveyed in the descriptor loop immediately following the program_info_length field in the Program Map Table. If a Program Stream Map is present in a Program Stream, the IOD descriptor shall be conveyed in the descriptor loop immediately following the program_stream_info_length field in the Program Stream Map. More than one IOD descriptor may be associated to a program.

NOTE – This Specification does not specify how the IOD_label may be used by higher level service information to uniquely select one of the ISO/IEC 14496 presentations identified by multiple IOD descriptors.

Table 2-72 – IOD descriptor

Syntax	No. of bits	Mnemonic
IOD_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
Scope_of_IOD_label	**8**	**uimsbf**
IOD_label	**8**	**uimsbf**
InitialObjectDescriptor ()	**8**	**uimsbf**
}		

2.6.41 Semantic definition of fields in IOD descriptor

Scope_of_IOD_label – This 8-bit field specifies the scope of the IOD_label field. A value of 0x10 indicates that the IOD_label is unique within the Program Stream or within the specific program in a Transport Stream in which the IOD descriptor is carried. A value of 0x11 indicates that the IOD_label is unique within the Transport Stream in which the IOD descriptor is carried. All other values of the Scope_of_IOD_label field are reserved.

IOD_label – This 8-bit field specifies the label of the IOD descriptor.

InitialObjectDescriptor () – This structure is defined in 8.6.3.1 of ISO/IEC 14496-1.

2.6.42 SL descriptor

The SL descriptor shall be used when a single ISO/IEC 14496-1 SL-packetized stream is encapsulated in PES packets. The SL descriptor associates the ES_ID of this SL-packetized stream to an elementary_PID in case of a Transport Stream or to an elementary_stream_id in case of a Program Stream. Within a Transport Stream, the SL descriptor shall be conveyed for the corresponding elementary stream in the descriptor loop immediately following the ES_info_length field in the Program Map Table. If a Program Stream Map is present in a Program Stream, the SL descriptor shall be conveyed in the descriptor loop immediately following the elementary_stream_info_length field within the Program Stream Map.

NOTE – SL packetized streams may be used in a Program Stream. However, only one stream_id exists for ISO/IEC 14496-1 SL-packetized streams. In order to associate multiple such streams within a Program Stream to an ISO/IEC 14496-1 scene, FlexMux has to be used and signalled appropriately by an FMC descriptor. This limitation does not exist in a Transport Stream where the SL descriptor provides unambiguous mapping between an ISO/IEC 14496-1 ES_ID value and an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 elementary_PID value.

Table 2-73 – SL descriptor

Syntax	No. of bits	Mnemonic
SL_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
ES_ID	**16**	**uimsbf**
}		

2.6.43 Semantic definition of fields in SL descriptor

ES_ID – This 16-bit field shall specify the identifier of an ISO/IEC 14496-1 SL-packetized stream.

2.6.44 FMC descriptor

The FMC descriptor indicates that the ISO/IEC 14496-1 FlexMux tool has been used to multiplex ISO/IEC 14496-1 SL-packetized streams into a FlexMux stream before encapsulation in PES packets or ISO//IEC14496_sections. The FMC descriptor associates FlexMux channels to the ES_ID values of the SL-packetized streams in the FlexMux stream.

An FMC descriptor is required for each program element referenced by an elementary_PID value in a Transport Stream and for each elementary_stream_id in a Program Stream that conveys a FlexMux stream. Within a Transport Stream, the FMC descriptor shall be conveyed for the corresponding elementary stream in the descriptor loop immediately following the ES_info_length field in the Program Map Table. If a Program Stream Map is present in a Program Stream, the FMC descriptor shall be conveyed in the descriptor loop immediately following the elementary_stream_info_length field in the Program Stream Map.

For each SL_packetized stream in a FlexMux stream, the FlexMux channel shall be identified by a single entry in the FMC descriptor.

Table 2-74 – FMC descriptor

Syntax	No. of bits	Mnemonic
FMC_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
for (i = 0; i < descriptor_length; i + = 3) {		
ES_ID	**16**	**uimsbf**
FlexMuxChannel	**8**	**uimsbf**
}		
}		

2.6.45 Semantic definition of fields in FMC descriptor

ES_ID – This 16-bit field specifies the identifier of an ISO/IEC 14496-1 SL-packetized stream.

FlexMuxChannel – This 8-bit field specifies the number of the FlexMux channel used for this SL-packetized stream.

2.6.46 External_ES_ID descriptor

The External_ES_ID descriptor assigns an ES_ID, as defined in ISO/IEC 14496-1, to a program element to which no ES_ID value has been assigned by other means. This ES_ID allows reference to a non-ISO/IEC 14496 component in the scene description or, for example, to associate a non-ISO/IEC 14496 component with an IPMP stream.

Within a Transport stream, the assignment of an ES_ID shall be made by conveying an External_ES_ID descriptor for the corresponding elementary stream in the descriptor loop immediately following the ES_info_length field in the Program Map Table. If a Program Stream Map is present in a Program Stream, the External_ES_ID descriptor shall be conveyed in the descriptor loop immediately following the elementary_stream_info_length field in the Program Stream Map.

Table 2-75 – External_ES_ID descriptor

Syntax	No. of bits	Mnemonic
External_ES_ID_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
External_ES_ID	**16**	**uimsbf**
}		

2.6.47 Semantic definition of fields in External_ES_ID descriptor

External_ES_ID – This 16-bit field assigns an ES_ID identifier, as defined in ISO/IEC 14496-1, to a component of a program.

2.6.48 Muxcode descriptor

The Muxcode descriptor conveys MuxCodeTableEntry structures as defined in 11.2.4.3 of ISO/IEC 14496-1. MuxCodeTableEntries configure the MuxCode mode of FlexMux.

One or more Muxcode descriptors may be associated to each elementary_PID or elementary_stream_id, respectively, conveying an ISO/IEC 14496-1 FlexMux stream that utilizes the MuxCode mode. Within a Transport stream, the Muxcode descriptor shall be conveyed for the corresponding elementary stream in the descriptor loop immediately following the ES_info_length field in the Program Map Table. If a Program Stream Map is present in a Program Stream, the Muxcode descriptor shall be conveyed in the descriptor loop immediately following the elementary_stream_info_length field in the Program Stream Map.

MuxCodeTableEntries may be updated with new versions. In case of such updates, the version_number of each Program Map Table or the program_stream_map_version of each Program Stream Map, respectively, carrying the MuxCode descriptor in their descriptor loop shall be incremented by 1 modulo 32.

Table 2-76 – Muxcode descriptor

Syntax	No. of bits	Mnemonic
Muxcode_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
for (i = 0; i < N; i++) {		
MuxCodeTableEntry ()		
}		
}		

2.6.49 Semantic definition of fields in Muxcode descriptor

MuxCodeTableEntry () – This structure is defined in 11.2.4.3 of ISO/IEC 14496-1.

2.6.50 FmxBufferSize descriptor

The FmxBufferSize descriptor conveys the size of the FlexMux buffer (FB) for each SL packetized stream multiplexed in a FlexMux stream.

One FmxBufferSize descriptor shall be associated to each elementary_PID or elementary_stream_id, respectively, conveying an ISO/IEC 14496-1 FlexMux stream. Within a Transport stream, the FmxBufferSize descriptor shall be conveyed for the corresponding elementary stream in the descriptor loop immediately following the ES_info_length field in the Program Map Table. If a Program Stream Map is present in a Program Stream, the FmxBufferSize descriptor shall be conveyed in the descriptor loop immediately following the elementary_stream_info_length field within the Program Stream Map.

Table 2-77 – FmxBufferSize descriptor

Syntax	No. of bits	Mnemonic
FmxBufferSize_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
DefaultFlexMuxBufferDescriptor()		
for (i=0; i<descriptor_length; i += 4) {		
FlexMuxBufferDescriptor()		
}		
}		

2.6.51 Semantic definition of fields in FmxBufferSize descriptor

FlexMuxBufferDescriptor() – This descriptor specifies the FlexMux buffer size for one SL-packetized stream carried within the FlexMux stream. It is defined in 11.2 of ISO/IEC 14496-1.

DefaultFlexMuxBufferDescriptor() – This descriptor specifies the default FlexMux buffer size for this FlexMux stream. It is defined in 11.2 of ISO/IEC 14496-1.

2.6.52 MultiplexBuffer descriptor

The MultiplexBuffer descriptor conveys the size of the multiplex buffer MB_n, as well as the leak rate Rx_n at which data is transferred from transport buffer TB_n into buffer MB_n for a specific ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program element referenced by an elementary_PID value in the Program Map Table.

One MultiplexBuffer descriptor shall be associated to each elementary_PID that contains an ISO/IEC 14496 FlexMux stream or SL-packetized stream, including those containing ISO_IEC_14496_sections. See 2.11.3.9 for the definition of buffers and rates in the T-STD model for decoding of ISO/IEC 14496 content.

The MultiplexBuffer descriptor shall be conveyed in the descriptor loop immediately following the ES_info_length field in the Program Map Table.

Table 2-78 – MultiplexBuffer descriptor

Syntax	No. of bits	Mnemonic
MultiplexBuffer_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
MB_buffer_size	**24**	**uimsbf**
TB_leak_rate	**24**	**uimsbf**
}		

2.6.53 Semantic definition of fields in MultiplexBuffer descriptor

MB_buffer_size – This 24-bit field shall specify the size in byte of buffer MB_n of the elementary stream n that is associated with this descriptor.

TB_leak_rate – This 24-bit field shall specify in units of 400 bits per second the rate at which data is transferred from transport buffer TB_n to multiplex buffer MB_n for the elementary stream n that is associated with this descriptor.

2.6.54 FlexMuxTiming descriptor

See Table 2-79.

Table 2-79 – FlexMuxTiming descriptor

Syntax	No. of bits	Mnemonic
FlexMuxTiming_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
FCR_ES_ID	**16**	**uimsbf**
FCRResolution	**32**	**uimsbf**
FCRLength	**8**	**uimsbf**
FmxRateLength	**8**	**uimsbf**
}		

2.6.55 Semantic definition of fields in FlexMuxTiming descriptor

FCR_ES_ID – Is the ES_ID associated to this clock reference stream.

FCRResolution – Is the resolution of the object time base in cycles per second.

FCRLength – Is the length of the fmxClockReference field in FlexMux packets with index = 238. A length of zero shall indicate that no FlexMux packets with index = 238 are present in this FlexMux stream. FCRlength shall take values between zero and 64.

FmxRateLength – Is the length of the fmxRate field in FlexMux packets with index = 238. FmxRateLength shall take values between 1 and 32.

2.6.56 Content labelling descriptor

The content labelling descriptor assigns a label to content; the label can be used by metadata to reference the associated content. This label, the content_reference_id_record, is metadata application format specific. The content labelling

descriptor is associated with a content segment. For the purpose of this clause, a content segment is defined as a portion in time of a program, an elementary stream (such as audio or video) or any combination of programs or elementary streams. The descriptor may be included in the PMT in the descriptor loop for either the program or an elementary stream, but may also be contained in tables not defined in this Specification, for example tables to describe segments of programs or elementary streams. The content labelling descriptor also provides information on which content time base is used and on the offset between the content time base and the metadata time base. When the Normal Play Time (NPT) concept of DSM-CC, as specified in ISO/IEC 13818-6, is used as the content time base, the ID of the NPT time base is provided. The descriptor allows for carriage of private data. See Table 2-80.

Table 2-80 – Content labelling descriptor

Syntax	No. of bits	Mnemonic
Content_labeling_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
metadata_application_format	**16**	**uimsbf**
if (metadata_application_format== 0xFFFF){		
metadata_application_format_identifier	**32**	**uimsbf**
}		
content_reference_id_record_flag	**1**	**bslbf**
content_time_base_indicator	**4**	**uimsbf**
reserved	**3**	**bslbf**
if (content_reference_id_record_flag == '1'){		
content_reference_id_record_length	**8**	**uimsbf**
for (i=0; i<content_reference_id_record_length;i++){		
content_reference_id_byte	**8**	**bslbf**
}		
}		
if (content_time_base_indicator== 1|2){		
reserved	**7**	**bslbf**
content_time_base_value	**33**	**uimsbf**
reserved	**7**	**bslbf**
metadata_time_base_value	**33**	**uimsbf**
}		
if (content_time_base_indicator== 2){		
reserved	**1**	**bslbf**
contentId	**7**	**uimsbf**
}		
if (content_time_base_indicator==3|4|5|6|7){		
time_base_association_data_length	**8**	**uimsbf**
for (i=0; i< time_base_association_data_length;i++){		
reserved	**8**	**bslbf**
}		
}		
for (i=0; i<N;i++){		
private_data_byte	**8**	**bslbf**
}		
}		

2.6.57 Semantic definition of fields in content labelling descriptor

metadata_application_format: The metadata_application_format is a 16-bit field, coded as defined in Table 2-81, that specifies the application responsible for defining usage, syntax and semantics of the content_reference_id record and of any other privately defined fields in this descriptor. See also 2.12.1. The value 0xFFFF indicates that the format is signalled by the value carried in the metadata_application_format_identifier field.

Table 2-81 – Metadata_application_format

Value	Description
0x0000-0x000F	Reserved
0x0010	ISO 15706 (ISAN) encoded in its binary form (see Notes 1 and 3)
0x0011	ISO 15706-2 (V-ISAN) encoded in its binary form (see Notes 2 and 3)
0x0012-0x00FF	Reserved
0x0100-0xFFFE	User defined
0xFFFF	Defined by the metadata_application_format_identifier field

Table 2-81 – Metadata_application_format

NOTE 1 – For ISAN, the content_reference_id_byte is set to binary encoding and the content_reference_id_record_length is set to 0x08.

NOTE 2 – For V-ISAN, the content_reference_id_byte is set to binary encoding and the content_reference_id_record_length is set to 0x0C.

NOTE 3 – For interoperability amongst metadata applications that use the metadata_application_format values of 0x0010 and 0x0011, it is recommended that the content_reference_id_flag be set to '1' and the content_time_base_indicator be set to '00'.

metadata_application_format_identifier: The coding of this 32-bit field is fully equivalent to the coding of the format_identifier field in the registration_descriptor, as defined in 2.6.8.

NOTE – The assigned Registration Authority for the format_identifier field is SMPTE.

content_reference_id_record_flag: The content_reference_id_record_flag is a 1-bit flag that signals the presence of a content_reference_id_record in this descriptor.

content_time_base_indicator: The content_time_base_indicator is a 4-bit field which specifies the used content time base. If the descriptor is associated with a program, then the content time base applies to all streams that are part of that program. A value of 1 indicates usage of the STC, while a value of '2' indicates usage of NPT, the Normal Play Time as defined in ISO/IEC 13818-6. The values between 8 and 15 indicate usage of a privately defined content time base. If coded with a value of '0', no content time base is defined in this descriptor. If no content time base is specified for a program or stream, then the mapping of time references in the metadata to the content is not defined in this Specification.

Table 2-82 – Content_time_base_indicator values

Value	Description
0	No content time base defined in this descriptor
1	Use of STC
2	Use of NPT
3-7	Reserved
8-15	Use of privately defined content time base

content_reference_id_record_length: The content_reference_id_record_length is an 8-bit field that specifies the number of content_reference_id_bytes immediately following this field. This field shall not be coded with the value '0'.

content_reference_id_byte: The content_reference_id_byte is part of a string of one or more contiguous bytes that assigns one or more reference identifications (labels) to the content to which this descriptor is associated. The format of this byte string is defined by the body indicated by the coded value in the metadata_application_format field.

content_time_base_value: The content_time_base_value is a 33-bit field that specifies a value in units of 90 kHz of the content time base indicated by the content_time_base_indicator field.

metadata_time_base_value: The metadata_time_base_value is a 33-bit field that is coded in units of 90 kHz. The field is coded with the value of the metadata time base at the instant in time in which the time base indicated by content_time_base_indicator reaches the value encoded in the content_time_base_value field. Note that the metadata time base may use any time-scale, but that its value is to be coded in units of 90 kHz. For example, if a SMPTE type of time code is used, then the number of hours, minutes, seconds and frames is expressed in the corresponding number of 90-kHz units.

contentId: The contentId is a 7-bit field that specifies the value of the content_Id field in the NPT Reference Descriptor for the applied NPT time base.

time_base_association_data_length: The time_base_association_data_length is an 8-bit field that specifies the number of reserved bytes immediately following this field. The reserved bytes can be used to carry time base association data for time bases defined in future.

private_data_byte: The private_data_byte is an 8-bit field. The private_data_bytes represent data, the format of which is defined privately. These bytes can be used to provide additional information as deemed appropriate. The use of these bytes is defined by the metadata application format.

2.6.58 Metadata pointer descriptor

The metadata pointer descriptor points to a single metadata service and associates this metadata service with audiovisual content in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. The metadata is associated with the content within the context of the descriptor. The context is defined by the location of the descriptor. In a transport stream, the descriptor may be located in the PMT in the descriptor loop for either the program or an elementary stream, but may also be located in tables not defined in this Specification, such as tables describing bouquets of broadcast services. The metadata may be located in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, but the same metadata may also be provided on alternative locations, such as the Internet.

The descriptor may contain location information of metadata that is not carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream; the coding of the location information is metadata application format specific. The descriptor allows for carriage of private data.

For metadata carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, the descriptor specifies the tools used for such carriage. If the metadata is carried in PES packets, metadata sections, or ISO/IEC 13818-6 synchronized download sections, the metadata_service_id field identifies the metadata service in the referenced metadata stream. If an ISO/IEC 13818-6 carousel is used to carry the metadata, then the private data may provide information to signal the metadata service, such as the applied value of the module_id for carriage of the metadata in a data carousel, and the file name of the metadata when the object carousel is used.

Receivers should be aware that multiple metadata services may be pointed to from the same program or audiovisual stream (as defined by the context of the descriptor). A unique metadata pointer descriptor shall be used to point to each metadata service used by the program or audiovisual stream. Similarly, the same metadata service can be pointed to from several programs or audiovisual streams by using a separate metadata pointer descriptors for each association.

Table 2-83 – Metadata pointer descriptor

Syntax	No. of bits	Mnemonic
Metadata_pointer_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
metadata_application_format	**16**	**uimsbf**
if (metadata_application_format== 0xFFFF){		
metadata_application_format_identifier	**32**	**uimsbf**
}		
metadata_format	**8**	**uimsbf**
if (metadata_format== 0xFF){		
metadata_format_identifier	**32**	**uimsbf**
}		
metadata_service_id	**8**	**uimsbf**
metadata_locator_record_flag	**1**	**bslbf**
MPEG_carriage_flags	**2**	**uimsbf**
reserved	**5**	**bslbf**
if (metadata_locator_record_flag == '1'){		
metadata_locator_record_length	**8**	**uimsbf**
for (i = 0; i < metadata_locator_record_length; i + +){		
metadata_locator_record_byte	**8**	**bslbf**
}		
}		
if (MPEG_carriage_flags == 0|1|2){		
program_number	**16**	**uimsbf**
}		
if (MPEG_carriage_flags == 1){		
transport_stream_location	**16**	**uimsbf**
transport_stream_id	**16**	**uimsbf**
}		
for (i=0; i<N;i++){		
private_data_byte	**8**	**bslbf**
}		
}		

2.6.59 Semantic definition of fields in metadata pointer descriptor

metadata_application_format: The metadata_application_format is a 16-bit field that specifies the application responsible for defining usage, syntax and semantics of the metadata_locator_record record and any other privately defined fields in this descriptor. The coding of this field is defined in Table 2-81 in 2.6.57.

metadata_application_format_identifier: The coding of this field is defined in subclause 2.6.57.

metadata_format: The metadata_format is an 8-bit field that indicates the format and coding of the metadata. The coding of this field is specified in Table 2-84.

Table 2-84 – Metadata format values

Value	Description
0x00-0x0F	Reserved
0x10	ISO/IEC 15938-1 TeM
0x11	ISO/IEC 15938-1 BiM
0x12-0x3E	Reserved
0x3F	Defined by metadata application format
0x40-0xFE	Private use
0xFF	Defined by metadata_format_identifier field

The values 0x10 and 0x11 identify ISO/IEC 15938-1 defined data. The value 0x3F indicates that the format is defined by the body indicated by the metadata_application_format field. The values in the inclusive range of 0x40 up to 0xFE are available to signal use of private formats. The value 0xFF indicates that the format is signalled by the metadata_format_identifier field.

metadata_format_identifier: The coding of this 32-bit field is fully equivalent to the coding of the format_identifier field in the registration_descriptor, as defined in 2.6.8.

NOTE – SMPTE is assigned as Registration Authority for the format_identifier field.

metadata_service_id: This 8-bit field references the metadata service. It is used for retrieving a metadata service from within a metadata stream.

metadata_locator_record_flag: The metadata_locator_record_flag is a 1-bit field which, when set to '1' indicates that associated metadata is available on a location outside of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, specified in a metadata_locator_record.

MPEG_carriage_flags: The MPEG_carriage_flags is a 2-bit field which specifies if the metadata stream containing the associated metadata service is carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, and if so, whether the associated metadata is carried in a Transport Stream or Program Stream. The coding of this field is defined in Table 2-85.

Table 2-85 – MPEG_carrier_flags

Value	Description
0	Carriage in the same Transport Stream where this metadata pointer descriptor is carried.
1	Carriage in a different Transport Stream from where this metadata pointer descriptor is carried.
2	Carriage in a Program Stream. This may or may not be the same Program Stream in which this metadata pointer descriptor is carried.
3	None of the above.

metadata_locator_record_length: The metadata_locator_record_length is an 8-bit field that specifies the number of metadata_locator_record_bytes immediately following. This field shall not be coded with the value '0'.

metadata_locator_record_byte: The metadata_locator_record_byte is part of a string of one or more contiguous bytes that form the metadata locator record. This record specifies one or more locations outside of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. The format of the metadata locator record is defined by the metadata application signalled by the metadata_application_format field. The record may for example contain Internet URLs that specify where the metadata can be found, possibly in addition to their location(s) in the Transport Stream. If the MPEG_carriage_flags is coded with the value '0', '1' or '2' and the metadata locator record is present, then this signals alternative locations for the same metadata.

program_number: The program_number is a 16-bit field that identifies the program_number of the MPEG-2 program in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream in which associated metadata is carried. If the MPEG_carriage_flags have the value '0', then the transport stream is the current one, and if the MPEG_carriage_flags have the value '1', it is the transport stream signalled by the fields transport_stream_location and transport_stream_id.

transport_stream_location: The transport_stream_location is a 16-bit field that is defined privately. For example, this field may be used by applications to signal the original_network_id defined by ETSI.

transport_stream_id: The transport_stream_id is a 16-bit field that identifies the Transport Stream in which associated metadata is carried.

private_data_byte: The private_data_byte is an 8-bit field. The private_data_bytes represent data, the format of which is defined privately. These bytes can be used to provide additional information as deemed appropriate.

2.6.60 Metadata descriptor

The metadata descriptor specifies parameters of a metadata service carried in an MPEG-2 TS or PS. In an MPEG-2 TS, the descriptor is included in the PMT in the descriptor loop for the elementary stream that carries the metadata service. The descriptor specifies the format of the associated metadata, and contains the value of the metadata_service_id to identify the metadata service to which the metadata descriptor applies. As needed, the descriptor can convey information to identify the metadata service from a collection of metadata transmitted in a DSM-CC carousel. Optionally metadata application format specific private data can be carried.

The metadata descriptor also signals whether decoder configuration is required and is able to carry the decoder configuration bytes, but this is only practical if the number of these bytes is small. If the decoder configuration information is too large to be carried by the descriptor, it shall be contained in a metadata service. This may be within the metadata service itself, or in another metadata service within the same program. Identification of the metadata service that contains the decoder configuration is provided by the metadata descriptor. If a DSM-CC carousel is used to carry the decoder configuration, then information can be provided how to retrieve the decoder configuration from the carousel.

Table 2-86 – Metadata descriptor

Syntax	No. of bits	Mnemonic
Metadata_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
metadata_application_format	**16**	**uimsbf**
if (metadata_application_format == 0xFFFF) {		
metadata_application_format_identifier	**32**	**uimsbf**
}		
metadata_format	**8**	**uimsbf**
if (metadata_format== 0xFF){		
metadata_format_identifier	**32**	**uimsbf**
}		
metadata_service_id	**8**	**uimsbf**
decoder_config_flags	**3**	**bslbf**
DSM-CC_flag	**1**	**bslbf**
reserved	**4**	**bslbf**
if (DSM-CC_flag == '1'){		
service_identification_length	**8**	**uimsbf**
for(i=0; i<service_identification_length; i++) {		
service_identification_record_byte	**8**	**bslbf**
}		
}		
if (decoder_config_flags == '001') {		
decoder_config_length	**8**	**uimsbf**
for(i=0; i<decoder_config_length; i++) {		
decoder_config_byte	**8**	**bslbf**
}		
}		
if (decoder_config_flags == '011') {		
dec_config_identification_record_length	**8**	**uimsbf**
for(i=0;i<dec_config_id_record_length;i++) {		
dec_config_identification_record_byte	**8**	**bslbf**
}		
}		
if (decoder_config_flags == '100') {		
decoder_config_metadata_service_id	**8**	**uimsbf**
}		
if (decoder_config_flags == '101'|'110') {		
reserved_data_length	**8**	**uimsbf**
for(i=0;i<reserved_data_length;i++) {		
reserved	**8**	**bslbf**
}		
}		
for (i=0; i<N;i++) {		
private_data_byte	**8**	**bslbf**
}		
}		

2.6.61 Semantic definition of fields in metadata descriptor

metadata_application_format: The metadata_application_format is a 16-bit field that specifies the application responsible for defining usage, syntax and semantics of the service_identification_record and any privately defined bytes in this descriptor. The coding of this field is defined in Table 2-81.

metadata_application_format_identifier: The coding of this field is defined in 2.6.57.

metadata_format: The coding of this field is defined in 2.6.59.

metadata_format_identifier: The coding of this field is defined in 2.6.59.

metadata_service_id. This 8-bit field identifies the metadata service to which this metadata descriptor applies.

decoder_config_flags: The decoder_config_flags is a 3-bit field which indicates whether and how decoder configuration information is conveyed.

Table 2-87 – decoder_config_flags

Value	Description
000	No decoder configuration is needed.
001	The decoder configuration is carried in this descriptor in the decoder_config_byte field.
010	The decoder configuration is carried in the same metadata service as to which this metadata descriptor applies.
011	The decoder configuration is carried in a DSM-CC carousel. This value shall only be used if the metadata service to which this descriptor applies is using the same type of DSM-CC carousel.
100	The decoder configuration is carried in another metadata service within the same program, as identified by the decoder_config_metadata_service_id field in this metadata descriptor.
101, 110	Reserved.
111	Privately defined.

DSM-CC_flag: This is a one-bit flag that is set to '1' if the stream with which this descriptor is associated is carried in an ISO/IEC 13818-6 data or object carousel.

NOTE 1 – The use of the object or data carousel is indicated by the applied stream-type value for this metadata stream.

service_identification length: This field specifies the number of service_identification_record_bytes immediately following.

service_identification_record_byte: This byte is part of a string of one or more contiguous bytes that specify the service_identification_record. This record contains data on retrieval of the metadata service from a DSM-CC carousel. The format of the metadata locator record is defined by the application indicated by the metadata application format. When a DSM-CC object carousel is used, the record may for example comprise the unique object identifier (the IOP:IOR() from 11.3.1 and 5.7.2.3 of ISO/IEC 13818-6 DSM-CC) for the metadata service. Similarly, in case of a DSM-CC data carousel, the record can for example provide the transaction_id and the module_id of the metadata service.

decoder_config_length: This field specifies the number of decoder_config_bytes immediately following.

decoder_config_byte: These bytes comprise the decoder configuration information. This sequence of bytes comprises the configuration information needed by the receiver to decode this service. It is intended that carriage in the metadata descriptor is only used when the configuration information is very small.

decoder_config_DSM-CC_id: This is the download identifier of the decoder configuration information when it is transmitted in a DSM-CC data carousel, or the object identifier of the decoder configuration information if it is carried in a DSM-CC object carousel.

NOTE 2 – The use of the object or data carousel is indicated by the applied stream-type value for this metadata stream.

dec_config_identification_record_length: This field specifies the immediately following number of dec_config_identification_record_bytes.

dec_config_identification_record_byte: This byte is part of a string of one or more contiguous bytes that specify the dec_config_identification_record. This record specifies how to retrieve the required decoder configuration from a DSM-CC carousel. The format of the metadata locator record is defined by the metadata application format. When a DSM-CC object carousel is used, the record may for example comprise the unique object identifier (the IOP:IOR() from 11.3.1 and 5.7.2.3 of ISO/IEC 13818-6 DSM-CC) for the decoder configuration. Similarly, in case of a DSM-CC data carousel, the record may for example provide the transaction_id and the module_id of the decoder configuration.

decoder_config_metadata_service_id: This is the value of the metadata_service_id that is assigned to the metadata service that contains the decoder configuration. The metadata service indicated by the decoder_config_metadata_service_id and the metadata service that uses that decoder configuration shall be in the same program. Hence in a Transport Stream, the metadata descriptors for both these metadata services shall be in the same PMT. The metadata descriptor of the metadata service indicated by the decoder_config_metadata_service_id shall have a decoder_config_flag field with a value of either '001', '010' or '011'.

reserved_data_length: This field specifies the number of reserved bytes immediately following.

private_data_byte: The private_data_byte is an 8-bit field. The private_data_bytes represent data, the format of which is defined privately. These bytes can be used to provide additional information as deemed appropriate.

2.6.62 Metadata STD descriptor

This descriptor defines parameters of the STD model (defined in 2.12.10) for the processing of the metadata stream to which this descriptor is associated.

Table 2-88 – Metadata STD descriptor

Syntax	No. of bits	Mnemonic
Metadata_STD_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
reserved	**2**	**bslbf**
metadata_input_leak_rate	**22**	**uimsbf**
reserved	**2**	**bslbf**
metadata_buffer_size	**22**	**uimsbf**
reserved	**2**	**bslbf**
metadata_output_leak_rate	**22**	**uimsbf**
}		

2.6.63 Semantic definition of fields in metadata STD descriptor

metadata_input_leak_rate: The metadata_input_leak_rate is a 22-bit field that specifies the leak rate for the associated metadata stream in the T-STD model out of the buffer TB_n into buffer B_n. The leak rate is specified in units of 400 bits/s. For metadata carried in a program stream, the coding of the metadata_input_leak_rate field is not specified, as the rate into B_n equals the rate of the program stream.

metadata_buffer_size: The metadata_buffer_size is a 22-bit field that specifies the size of buffer B_n in the STD model for the associated metadata stream. The size of B_n is specified in units of 1024 bytes.

metadata_output_leak_rate: The metadata_output_leak_rate is a 22-bit field that specifies for the associated metadata service the leak rate in the STD model out of buffer B_n to the decoder. The leak rate is specified in units of 400 bits/s. For metadata streams transported synchronously (stream-type 0x15 or 0x19), the metadata access units are instantaneously removed from B_n under the control of PTS timestamps and in that case the coding of the metadata_output_leak_rate field is not specified.

2.6.64 AVC video descriptor

For ITU-T Rec. H.264 | ISO/IEC 14496-10 video streams, the AVC video descriptor provides basic information for identifying coding parameters of the associated AVC video stream, such as on profile and level parameters included in the SPS of an AVC video stream.

The AVC video descriptor also signals the presence of AVC still pictures and the presence of AVC 24-hour pictures in the AVC video stream. If this descriptor is not included in the PMT for an AVC video stream in a transport stream or in the PSM, if present, for an AVC video stream in a program stream, then such AVC video stream shall not contain AVC still pictures and shall not contain AVC 24-hour pictures. (See Table 2-89.)

Table 2-89 – AVC video descriptor

Syntax	No. of bits	Mnemonic
AVC_video_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
profile_idc	**8**	**uimsbf**
constraint_set0_flag	**1**	**bslbf**
constraint_set1_flag	**1**	**bslbf**
constraint_set2_flag	**1**	**bslbf**
AVC_compatible_flags	**5**	**bslbf**
level_idc	**8**	**uimsbf**
AVC_still_present	**1**	**bslbf**
AVC_24_hour_picture_flag	**1**	**bslbf**
reserved	**6**	**bslbf**
}		

2.6.65 Semantic definition of fields in AVC video descriptor

profile_idc, constraint_set0_flag, **constraint_set1_flag**, **constraint_set2_flag, AVC_compatible_flags and level_idc** – These fields, with the exception of AVC_compatible_flags shall be coded according to the semantics for these fields defined in ITU-T Rec. H.264 | ISO/IEC 14496-10. The semantics of AVC_compatible_flags are exactly equal to the semantics of the field(s) defined for the 5 bits between the constraint_set2 flag and the level_idc field in the Sequence Parameter Set, as defined in ITU-T Rec. H.264 | ISO/IEC 14496-10. The entire AVC video stream to which the AVC descriptor is associated shall conform to the profile, level and constraints signalled by these fields.

NOTE – In one or more sequences in the AVC video stream the level may be lower than the level signalled in the AVC video descriptor, while also a profile may occur that is a subset of the profile signalled in the AVC video descriptor. However, in the entire AVC video stream, only tools shall be used that are included in the profile signalled in the AVC video descriptor, if present. For example, if the main profile is signalled, then the baseline profile may be used in some sequences, but only using those tools that are in the main profile. If the sequence parameter sets in an AVC video stream signal different profiles, and no additional constraints are signalled, then the stream may need examination to determine which profile, if any, the entire stream conforms to. If an AVC video descriptor is to be associated with an AVC video stream that does not conform to a single profile, then the AVC video stream must be partitioned into two or more sub-streams, so that AVC video descriptors can signal a single profile for each such sub-stream.

AVC_still_present – This 1-bit field when set to '1' indicates that the AVC video stream may include AVC still pictures. When set to '0', then the associated AVC video stream shall not contain AVC still pictures.

AVC_24_hour_picture_flag – This 1-bit flag when set to '1' indicates that the associated AVC video stream may contain AVC 24-hour pictures. For the definition of an AVC 24-hour picture, see 2.1.2. If this flag is set to '0', the associated AVC video stream shall not contain any AVC 24-hour picture.

2.6.66 AVC timing and HRD descriptor

The AVC timing and HRD descriptor provides timing and HRD parameters of the associated AVC video stream. For each AVC video stream carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, the AVC timing and HRD descriptor shall be included in the PMT or in the PSM, if PSM is present in the program stream, unless the AVC video stream carries VUI parameters with the timing_info_present_flag set to '1':

- for each IDR picture; and
- for each picture that is associated with a recovery point SEI message.

Absence of the AVC timing and HRD descriptor in the PMT for an AVC video stream signals usage of the leak method in the T-STD is defined in 2.14.3.1 for the transfer from MB_n to EB_n, but such usage can also be signalled by the hrd_management_valid_flag set to '0' in the AVC timing and HRD descriptor. If the transfer rate into buffer EB_n can be determined from HRD parameters contained in an AVC video stream, and if this transfer rate is used in the T-STD for the transfer between MB_n to EB_n, then the AVC timing and HRD descriptor with the hrd_management_valid_flag set to '1' shall be included in the PMT for that AVC video stream. (See Table 2-90.)

Table 2-90 – AVC timing and HRD descriptor

Syntax	No. of bits	Mnemonic
AVC timing and HRD descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
hrd_management_valid_flag	**1**	**bslbf**
reserved	**6**	**bslbf**
picture_and_timing_info_present	**1**	**bslbf**
if (picture_and_timing_info_present) {		
90kHz_flag	**1**	**bslbf**
reserved	**7**	**bslbf**
if (90kHz_flag = = '0') {		
N	**32**	**uimsbf**
K	**32**	**uimsbf**
}		
num_units_in_tick	**32**	**uimsbf**
}		
fixed_frame_rate_flag	**1**	**bslbf**
temporal_poc_flag	**1**	**bslbf**
picture_to_display_conversion_flag	**1**	**bslbf**
reserved	**5**	**bslbf**
}		

2.6.67 Semantic definition of fields in AVC timing and HRD descriptor

hrd_management_valid_flag – This 1-bit field is only defined for use in transport streams.

When the AVC timing and HRD descriptor is associated to an AVC video stream carried in a transport stream, then the following applies. If the hrd_management_valid_flag is set to '1', then Buffering Period SEI and Picture Timing SEI messages, as defined in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10, shall be present in the associated AVC video stream. These Buffering Period SEI messages shall carry coded initial_cpb_removal_delay and initial_cpb_removal_delay_offset values for the NAL HRD. If the hrd_management_valid_flag is set to '1', then the transfer of each byte from MB_n to EB_n in the T-STD shall be according to the delivery schedule for that byte into the CPB in the NAL HRD, as determined from the coded initial_cpb_removal_delay and initial_cpb_removal_delay_offset values for SchedSelIdx = cpb_cnt_minus1. When the hrd_management_valid_flag is set to '0', the leak method as defined in 2.14.3.1 shall be used for the transfer from MB_n to EB_n in the T-STD.

When the AVC timing and HRD descriptor is associated to an AVC video stream carried in a program stream, then the meaning of the hrd_management_valid_flag is not defined.

picture_and_timing_info_present – This 1-bit field when set to '1' indicates that the 90kHz_flag and parameters for accurate mapping to 90-kHz system clock are included in this descriptor.

90kHz_flag, N, K – The 90kHz_flag when set to '1' indicates that the frequency of the AVC time base is 90 kHz. For an AVC video stream the frequency of the AVC time base is defined by the AVC parameter time_scale in VUI parameters, as defined in Annex E of ITU-T Rec. H.264 | ISO/IEC 14496-10. The relationship between the AVC time_scale and the STC shall be defined by the parameters N and K in this descriptor as follows.

$$time\_scale = \frac{(N \times system\_clock\_frequency)}{K}$$

where time_scale denotes the exact frequency of the AVC time base, with K larger than or equal to N.

If the 90kHz_flag is set to '1', then N equals 1 and K equals 300. If the 90kHz_flag is set to '0', then the values of N and K are provided by the coded values of the N and K fields.

NOTE 1 – This allows mapping of time expressed in units of time_scale to 90-kHz units, as needed for the calculation of PTS and DTS timestamps, for example in decoders for AVC access units for which no PTS or DTS is encoded in the PES header.

num_units_in_tick – Coded exactly in the same way as the num_units_in_tick field in VUI parameters in Annex E of ITU-T Rec. H.264 | ISO/IEC 14496-10. The information provided by this field shall apply to the entire AVC video stream to which the AVC timing and HRD descriptor is associated.

fixed_frame_rate_flag – Coded exactly in the same way as the fixed_frame_rate_flag in VUI parameters in Annex E of ITU-T Rec. H.264 | ISO/IEC 14496-10. When this flag is set to '1', it indicates that the coded frame rate is constant within the associated AVC video stream. When this flag is set to '0', no information about the frame rate of the associated AVC video stream is provided in this descriptor.

temporal_poc_flag – When the temporal_poc_flag is set to '1' and the fixed_frame_rate_flag is set to '1', then the associated AVC video stream shall carry Picture Order Count (POC) information (PicOrderCnt) whereby pictures are counted in units of $\Delta t_{fi,dpb}(n)$, where $\Delta t_{fi,dpb}(n)$ is specified in equation E-10 of ITU-T Rec. H.264 | ISO/IEC 14496-10. When the temporal_poc_flag is set to '0', no information is conveyed regarding any potential relationship between the POC information in the AVC video stream and time.

NOTE 2 – This reduces the overhead necessary to signal timing for each access unit. An effective PTS and DTS can be calculated for access units for which no explicit PTS/DTS is carried. Repetition of most recently presented field of the appropriate parity (or frame) is implied when the difference between the PTSs of the current and the next picture is greater than $2 \times \Delta t_{fi,dpb}$ (or greater than $\Delta t_{fi,dpb}$ when *frame_mbs_only_flag* is equal to 1).

picture_to_display_conversion_flag – This 1-bit field when set to '1' indicates that the associated AVC video stream may carry display information on coded pictures by providing the pic_struct field in picture_timing SEI messages (see Annex D of ITU-T Rec. H.264 | ISO/IEC 14496-10) and/or by providing the Picture Order Count (POC) information (PicOrderCnt), whereby pictures are counted in units of $\Delta t_{fi,dpb}(n)$ (see also the semantics of temporal_poc_flag), so that timing information for a successive AVC access unit can be derived from the previous picture in decoding or presentation order.

When the picture_to_display_conversion_mode_flag is set to '0', then picture timing SEI messages in the AVC video stream, if present, shall not contain the pic_struct field, and hence the pic_struct_present_flag shall be set to '0' in the VUI parameters in the AVC video stream.

2.6.68 MPEG-2 AAC audio descriptor

For individual ISO/IEC 13818-7 streams directly carried in PES packets, the MPEG-2 AAC audio descriptor defined in Table 2-91 provides basic information for identifying the coding parameters of such audio elementary streams.

Table 2-91 – MPEG-2 AAC_audio_descriptor

Syntax	No. of bits	Mnemonic
MPEG-2_AAC_audio_descriptor () {		
descriptor_tag	**8**	**uimsbf**
descriptor_length	**8**	**uimsbf**
MPEG-2_AAC_profile	**8**	**uimsbf**
MPEG-2_AAC_channel_configuration	**8**	**uimsbf**
MPEG-2_AAC_additional_information	**8**	**uimsbf**
}		

2.6.69 Semantic definition of fields in MPEG-2 AAC audio descriptor

MPEG-2_AAC_profile – This 8-bit field indicates the AAC profile according to the index in Table 31 of ISO/IEC 13818-7:2006.

MPEG-2_AAC_channel_configuration – This 8-bit field indicates the number and configuration of audio channels presented to the listener by the AAC decoder for the specified program. Values in the range from 1 to 6 indicate number and configuration of audio channels as given for "Default bitstream index number" in Table 42 of ISO/IEC 13818-7:2006. All other values indicate that the number and configuration of audio channels is undefined.

MPEG-2_AAC_additional_information – This 8-bit field indicates whether or not bandwidth extension data as defined in ISO/IEC 13818-7:2006 is embedded in the AAC bitstream according to Table 2-92.

Table 2-92 – MPEG-2_AAC_additional_information field values

Value	Description
0x00	AAC data according to ISO/IEC 13818-7:2006
0x01	AAC data with Bandwidth Extension data present according to ISO/IEC 13818-7:2006
0x02-0xFF	Reserved

2.7 Restrictions on the multiplexed stream semantics

2.7.1 Frequency of coding the system clock reference

The Program Stream shall be constructed such that the time interval between the bytes containing the last bit of system_clock_reference_base fields in successive packs shall be less than or equal to 0.7 s. Thus:

$$|t(i) - t(i')| \leq 0.7\,\text{s}$$

for all i and i′ where i and i′ are the indexes of the bytes containing the last bit of consecutive system_clock_reference_base fields.

2.7.2 Frequency of coding the program clock reference

The Transport Stream shall be constructed such that the time interval between the bytes containing the last bit of program_clock_reference_base fields in successive occurrences of the PCRs in Transport Stream packets of the PCR_PID for each program shall be less than or equal to 0.1 s. Thus:

$$|t(i) - t(i')| \leq 0.1\,\text{s}$$

for all i and i′ where i and i′ are the indexes of the bytes containing the last bit of consecutive program_clock_reference_base fields in the Transport Stream packets of the PCR_PID for each program.

There shall be at least two (2) PCRs, from the specified PCR_PID within a Transport Stream, between consecutive PCR discontinuities (refer to 2.4.3.4) to facilitate phase locking and extrapolation of byte delivery times.

2.7.3 Frequency of coding the elementary stream clock reference

The Program Stream and Transport Stream shall be constructed such that if the elementary stream clock reference field is coded in any PES packets containing data of a given elementary stream the time interval in the PES_STD between the bytes containing the last bit of successive ESCR_base fields shall be less than or equal to 0.7 s. In PES Streams the ESCR encoding is required with the same interval. Thus:

$$|t(i) - t(i')| \leq 0.7\,\text{s}$$

for all i and i′ where i and i′ are the indexes of the bytes containing the last bits of consecutive ESCR_base fields.

NOTE – The coding of elementary stream clock reference fields is optional; they need not be coded. However, if they are coded, this constraint applies.

2.7.4 Frequency of presentation timestamp coding

The Program Stream and Transport Stream shall be constructed so that the maximum difference between coded presentation timestamps referring to each elementary video or audio stream is 0.7 s. Thus:

$$|tp_n(k) - tp_n(k'')| \leq 0.7\,\text{s}$$

for all n, k, and k″ satisfying:

- $P_n(k)$ and $P_n(k'')$ are presentation units for which presentation timestamps are coded;
- k and k″ are chosen so that there is no presentation unit, $P_n(k')$ with a coded presentation timestamp and with $k < k' < k''$; and
- No decoding discontinuity exists in elementary stream n between $P_n(k)$ and $P_n(k'')$.

The 0.7-s constraint does not apply in the case of:

- still pictures as defined in 2.1;
- AVC still pictures;
- AVC access units with a very low frame rate, where the presentation time of subsequent access units differs by more than 0.7 s. In this particular case, the VUI parameters num_units_in_tick and time_scale shall be present either in the AVC video stream or in an AVC-timing and HRD descriptor associated to the AVC video stream.

NOTE – The presentation time of an AVC access unit is equivalent to the DPB output time $t_{o,dpb}(n)$ defined in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10.

2.7.5 Conditional coding of timestamps

For each elementary stream of a Program Stream or Transport Stream, a presentation timestamp (PTS) shall be encoded for the first access unit.

A decoding discontinuity exists at the start of an access unit $A_n(j)$ in an elementary stream n if the decoding time $td_n(j)$ of that access unit is greater than the largest value permissible given the specified tolerance on the system_clock_frequency. For video, except when trick mode status is true or when low_delay flag is '1', this is allowed only at the start of a video sequence. If a decoding discontinuity exists in any elementary video or audio stream in the Transport Stream or Program Stream, then a PTS shall be encoded referring to the first access unit after each decoding discontinuity except when trick mode status is true.

When low_delay is '1' a PTS shall be encoded for the first access unit after an EB_n or B_n underflow.

A PTS may only be present in a ITU-T Rec. H.222.0 | ISO/IEC 13818-1 video or audio elementary stream PES packet header if the first byte of a picture start code or the first byte of an audio access unit is contained in the PES packet.

A decoding_timestamp (DTS) shall appear in a PES packet header if and only if the following two conditions are met:

- a PTS is present in the PES packet header;
- the decoding time differs from the presentation time.

For each AVC 24-hour picture, no explicit PTS and DTS value shall be encoded in the PES header. For such AVC access unit, decoders shall infer the presentation time from the parameters within the AVC video stream. Therefore, each AVC video stream that contains one or more AVC 24-hour picture(s):

- shall either carry picture timing SEI messages with coded values of cpb_removal_delay and dpb_output_delay; or
- shall carry VUI parameters with the fixed_frame_rate_flag set to '1' and shall carry Picture Order Count (POC) information (PicOrderCnt) whereby pictures are counted in units of $\Delta t_{fi,dpb}(n)$, where $\Delta t_{fi,dpb}(n)$ is specified in equation E-10 of ITU-T Rec. H.264 | ISO/IEC 14496-10.

  NOTE 1 – The requirements in the second bullet are met if an AVC timing and HRD descriptor is associated with the AVC video stream with the fixed_frame_rate_flag set to '1' and the temporal_poc_flag set to '1'.

The following applies to AVC access units in an AVC video stream carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. For each AVC access unit that does not represent an AVC 24-hour picture, a PES header with a coded PTS and, if applicable, DTS value shall be provided, unless all conditions expressed under one of the following four bullets are true:

- In the AVC video sequence the following SEI messages are present, as signalled by VUI parameters:
  a) picture timing SEI messages providing the cpb_removal_delay and the dpb_output_delay parameters; and
  b) buffering period SEI messages providing the initial_cpb_removal_delay and the initial_cpb_removal_delay_offset parameters.

  NOTE 2 – When picture timing SEI messages are present in the AVC video sequence, then these messages are present for each AVC access unit, as required by ITU-T Rec. H.264 | ISO/IEC 14496-10. When buffering period SEI messages are present in the AVC video sequence, then these messages shall be present for each IDR access unit and for each access unit that is associated with a recovery point SEI message, as required by ITU-T Rec. H.264 | ISO/IEC 14496-10.
- An AVC timing and HRD descriptor is associated with the AVC video stream and in this descriptor the fixed_frame_rate_flag is set to '1' and the temporal_poc_flag is set to '1'.
- An AVC timing and HRD descriptor is associated with the AVC video stream and in this descriptor the fixed_frame_rate_flag is set to '1', the picture_to_display_conversion_flag is set to '1', the temporal_poc_flag is set to '0' and in the AVC video sequence picture timing SEI messages with the pic_struct field are present.

  NOTE 3 – In this specific case the pic_struct field is used to determine subsequent PTS values.
- An AVC timing and HRD descriptor is associated with the AVC video stream and in this descriptor the fixed_frame_rate_flag is set to '1' and the temporal_poc_flag is set to '0' and the picture_to_display_conversion_flag is set to '0'.

  NOTE 4 – In this case the POC information in the AVC video stream is used to determine the subsequent PTS values.

2.7.6 Timing constraints for scalable coding

If an audio sequence is coded using an extension bitstream, such as specified in ISO/IEC 13818-3, then corresponding decoding/presentation units in the two layers shall have identical PTS values.

If a video sequence is coded as an SNR enhancement of another sequence, such as specified in 7.8 of ITU-T Rec. H.262 | ISO/IEC 13818-2, then the set of presentation times for both sequences shall be the same.

If a video sequence is coded as two partitions, such as specified in 7.10 of ITU-T Rec. H.262 | ISO/IEC 13818-2, then the set of presentation times for both partitions shall be the same.

If a video sequence is coded as a spatial scalable enhancement of another sequence, such as specified in 7.7 of ITU-T Rec. H.262 | ISO/IEC 13818-2, then the following shall apply:

- If both sequences have the same frame rate, the set of presentation times for both sequences shall be the same.

  NOTE – This does not imply that the picture coding type is the same in both layers.
- If the sequences have different frame rates, the set of presentation times shall be such that as many presentation times as possible shall be common to both sequences.
- The picture from which the spatial prediction is made shall be one of the following:
  - the coincident or most recently decoded lower layer picture;
  - the coincident or most recently decoded lower layer picture that is an I- or P-picture;
  - the second most recently decoded lower layer picture that is an I- or P-picture, and provided that the lower layer does not have the low_delay flag set to '1'.

If a video sequence is coded as a temporally scalable enhancement of another sequence, such as specified in 7.9 of ITU-T Rec. H.262 | ISO/IEC 13818-2, then the following lower layer pictures may be used as the reference. Times are relative to presentation times of:

- the coincident or most recently presented lower layer picture;
- the next lower layer picture to be presented.

2.7.7 Frequency of coding P-STD_buffer_size in PES packet headers

In a Program Stream, the P-STD_buffer_scale and P-STD_buffer_size fields shall occur in the first PES packet of each elementary stream and again whenever the value changes. They may also occur in any other PES packet.

2.7.8 Coding of system header in the Program Stream

In a Program Stream, the system header may be present in any pack, immediately following the pack header. The system header shall be present in the first pack of an Program Stream. The values encoded in all the system headers in the Program Stream shall be identical.

2.7.9 Constrained system parameter Program Stream

A Program Stream is a "Constrained System Parameters Stream" (CSPS) if it conforms to the bounds specified in this subclause. Program Streams are not limited to the bounds specified by the CSPS. A CSPS may be identified by means of the CSPS_flag defined in the system header in 2.5.3.5. The CSPS is a subset of all possible Program Streams.

Packet rate

In the CSPS, the maximum rate at which packets shall arrive at the input to the P-STD is 300 packets per second if the value encoded in the rate_bound field (refer to 2.5.3.6) is less than or equal to 4 500 000 bits/s if the packet_rate_restriction_flag is set to '1', and less than or equal to 2 000 000 bits/s if the packet_rate_restriction_flag is set to '0'. For higher bit rates the CSPS packet rate is bounded by a linear relation to the value encoded in the rate_bound field.

Specifically, for all packs p in the Program Stream when the packet_rate_restriction_flag (refer to 2.5.3.5) is set to a value of '1',

$$NP \le (t(i') - t(i')) \times 300 \times \max\left[1, \frac{R_{\max}}{4.5 \times 10^6}\right] \qquad (2\text{-}27)$$

and if the packet_rate_restriction_flag is set to a value of '0'

$$NP \le (t(i') - t(i')) \times 300 \times \max\left[1, \frac{R_{\max}}{2.5 \times 10^6}\right] \quad (2\text{-}28)$$

where:

$$R_{\max} = 8 \times 50 \times rate\_bound \quad \text{bit/s} \quad (2\text{-}29)$$

NP is the number of packet_start_code_prefixes and system_header_start_codes between adjacent pack_start_codes or between the last pack_start_code and the MPEG_program_end_code as defined in Table 2-37 and semantics in 2.5.3.2.

t(i) is the time, measured in seconds, encoded in the SCR of pack p.

t(i′) is the time, measured in seconds, encoded in the SCR for pack p + 1, immediately following pack p, or in the case of the final pack in the Program Stream, the time of arrival of the byte containing the last bit of the MPEG_program_end_code.

Decoder buffer size

In the case of a CSPS the maximum size of each input buffer in the system target decoder is bounded. Different bounds apply for video elementary streams and audio elementary streams.

In the case of an ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 video elementary stream in a CSPS, the following applies:

BS_n has a size which is equal to the sum of the size of the Video Buffer Verifier (VBV) as specified in the ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 stream, respectively, and an additional amount of buffering BS_{add}. BS_{add} is specified as:

$$BS_{add} \le MAX\ [6 \times 1024, R_{vmax} \times 0.001] \text{ bytes}$$

where R_{vmax} is the maximum bit rate of the ITU-T Rec. H.262 | ISO/IEC 13818-2 or ISO/IEC 11172-2 video elementary stream.

In the case of an ITU-T Rec. H.264 | ISO/IEC 14496-10 video elementary stream in a CSPS, the following applies:

BS_n has a size which is equal to the sum of cpb_size and an additional amount of buffering BS_{add}. BS_{add} is specified as:

$$BS_{add} \le MAX\ [6 \times 1024, R_{vmax} \times 0.001] \text{ bytes}$$

where R_{vmax} is the maximum video bit rate of the AVC video stream, and

where cpb_size is the CpbSize[cpt_cnt_minus1] size of the CPB for the byte stream format signalled in the NAL hrd_parameters() in the AVC video stream. If the NAL hrd_parameters() are not present in the AVC video stream, then the cpb_size shall be the size defined as 1200 × MaxCPB in Annex A of ITU-T Rec. H.264 | ISO/IEC 14496-10 for the applied level.

In the case of an audio elementary stream in a CSPS, the following applies:

$$BS_n \le 4096 \text{ bytes}$$

In the case of ISO/IEC 13818-7 ADTS audio elementary stream in a CSPS the following applies to support 8 channels:

$$BS_n \le 8976 \text{ bytes}$$

2.7.10 Transport Stream

Sample rate locking in Transport Streams

In the Transport Stream there shall be a specified constant rational relationship between the audio sampling rate and the system clock frequency in the system target decoder, and likewise a specified rational relationship between the video frame rate and the system clock frequency. The system_clock_frequency is defined in 2.4.2. The video frame rate is

specified in ITU-T Rec. H.262 | ISO/IEC 13818-2 or in ISO/IEC 11172-2. The audio sampling rate is specified in ISO/IEC 13818-3 or in ISO/IEC 11172-3. For all presentation units in all audio elementary streams in the Transport Stream, the ratio of system_clock_frequency to the actual audio sampling rate, SCASR, is constant and equal to the value indicated in the following table at the nominal sampling rate indicated in the audio stream.

$$SCASR = \frac{system\_clock\_frequency}{audio\_sample\_rate\_in\_the\_T-STD} \tag{2-30}$$

The notation $\frac{X}{Y}$ denotes real division.

Nominal audio sampling frequency (kHz)	16	32	22.05	44.1	24	48
SCASR	$\frac{27\,000\,000}{16\,000}$	$\frac{27\,000\,000}{32\,000}$	$\frac{27\,000\,000}{22\,050}$	$\frac{27\,000\,000}{44\,100}$	$\frac{27\,000\,000}{24\,000}$	$\frac{27\,000\,000}{48\,000}$

For all presentation units in each ISO/IEC 11172-2 video and ITU-T Rec. H.262 | ISO/IEC 13818-2 video stream in the Transport Stream, the ratio of system_clock_frequency to the actual video frame rate, SCFR, is constant and equal to the value indicated in the following table at the nominal frame rate indicated in the video stream.

$$SCFR = \frac{system\_clock\_frequency}{frame\_rate\_in\_the\_T-STD} \tag{2-31}$$

Nominal frame rate (Hz)	23.976	24	25	29.97	30	50	59.94	60
SCFR	1 126 125	1 125 000	1 080 000	900 900	900 000	540 000	450 450	450 000

The values of the SCFR are exact. The actual frame rate differs slightly from the nominal rate in cases where the nominal rate is 23.976, 29.97, or 59.94 frames per second.

For ISO/IEC 14496-2 video streams carried in a Transport Stream, the time base of the ISO/IEC 14496-2 video stream, as defined by vop_time_increment_resolution, shall be locked to the STC and shall be exactly equal to N times system_clock_frequency divided by K, with N and K integers that have a fixed value within each visual object sequence, with K greater than or equal to N.

For ITU-T Rec. H.264 | ISO/IEC 14496-10 video streams, the time base of the ITU-T Rec. H.264 | ISO/IEC 14496-10 video stream shall be locked to the system clock frequency. The frequency of the AVC time base is defined by the AVC parameter time_scale, and this frequency shall be exactly equal to N times system_clock_frequency divided by K, with N and K integers that have a fixed value within each AVC video sequence and K greater than or equal to N. For example, if the time_scale is set to 90 000, then the frequency of the AVC time base is exactly equal to system_clock_frequency divided by 300.

2.8 Compatibility with ISO/IEC 11172

The Program Stream of this Recommendation | International Standard is defined to be forward compatible with ISO/IEC 11172-1. Decoders of the Program Stream as defined in this Recommendation | International Standard shall also support decoding of ISO/IEC 11172-1.

2.9 Registration of copyright identifiers

2.9.1 General

Parts 1, 2 and 3 of ISO/IEC 13818 provide support for the management of audiovisual works copyrighting. In ITU-T Rec. H.222.0 | ISO/IEC 13818-1 this is by means of a copyright descriptor, while ITU-T Rec. H.262 | ISO/IEC 13818-2 and ISO/IEC 13818-3 contain fields for identifying copyright holders through syntax fields in the elementary stream syntax. This Recommendation | International Standard presents the method of obtaining and registering copyright identifiers in ITU-T Rec. H.222.0 | ISO/IEC 13818-1.

ITU-T Rec. H.222.0 | ISO/IEC 13818-1 specifies a unique 32-bit copyright_identifier which is a work type code identifier (such as ISBN, ISSN, ISRC, etc.) carried in the copyright descriptor. The copyright_identifier enables

identification of a wide number of Copyright Registration Authorities. Each Copyright Registration Authority may specify a syntax and semantic for identifying the audiovisual works or other copyrighted works within that particular copyright organization through appropriate use of the variable length additional_copyright_info field which contains the copyright number.

In the following subclause and Annexes L, M and N, the benefits and responsibilities of all parties to the registration of copyright_identifier are outlined.

2.9.2 Implementation of a Registration Authority (RA)

ISO/IEC JTC 1 shall call for nominations for an international organization which will serve as the Registration Authority for the **copyright_identifier** as defined in 2.6.24. The selected organization shall serve as the Registration Authority. The so-named Registration Authority shall execute its duties in compliance with Annex H/JTC 1 Directives. The registered copyright_identifier is hereafter referred to as the Registered Identifier (RID).

Upon selection of the Registration Authority, JTC 1 shall require the creation of a Registration Management Group (RMG) which will review appeals filed by organizations whose request for a RID to be used in conjunction with ITU-T Rec. H.222.0 | ISO/IEC 13818-1 has been denied by the Registration Authority.

Annexes L, M and N provide information on the procedure for registering a unique copyright identifier.

2.10 Registration of private data format

The registration descriptor of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 is provided by this text in order to enable users of this Specification to unambiguously carry data when its format is not recognized by this Specification. This provision will permit this Specification to carry all types of data while providing for a method of unambiguous identification of the characteristics of the underlying private data.

2.10.1 General

In the following subclause and Annexes O and P, the benefits and responsibilities of all parties to the registration of private data format are outlined.

2.10.2 Implementation of a Registration Authority (RA)

ISO/IEC JTC 1/SC 29 shall call for nominations from member bodies of ISO or National Committees of IEC which will serve as the Registration Authority for the **format_identifier** as defined in 2.6.8 and 2.6.9. The selected organization shall serve as the Registration Authority. The so-named Registration Authority shall execute its duties in compliance with Annex H/JTC 1 Directives. The registered private data format_identifier is hereafter referred to as the Registered Identifier (RID).

Upon selection of the Registration Authority, JTC 1 shall require the creation of a Registration Management Group (RMG) which will review appeals filed by organizations whose request for an RID to be used in conjunction with this Specification has been denied by the Registration Authority.

Annexes O and P provide information on the procedures for registering a unique format identifier.

2.11 Carriage of ISO/IEC 14496 data

2.11.1 Introduction

An ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream may carry individual ISO/IEC 14496-2 and 14496-3 elementary streams as well as ISO/IEC 14496-1 audiovisual scenes with its associated streams. Typically, the ISO/IEC 14496 streams will be elements of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program, as defined by the PMT in a Transport Stream and the PSM in a Program Stream.

For the carriage of ISO/IEC 14496 data in Transport Streams and Program Streams, distinction is made between individual elementary streams and an ISO/IEC 14496-1 audiovisual scene with its associated streams. For carriage of individual ISO/IEC 14496-2 and 14496-3 elementary streams, only system tools from ITU-T Rec. H.222.0 | ISO/IEC 13818-1 are used, as defined in 2.11.2. For carriage of an audiovisual ISO/IEC 14496-1 scene and associated ISO/IEC 14496 elementary streams, contained in ISO/IEC 14496-1 SL_packetized streams or FlexMux streams, tools from both ITU-T Rec. H.222.0 | ISO/IEC 13818-1 and from ISO/IEC 14496-1 are used, as defined in 2.11.3.

Carriage of ITU-T Rec. H.264 | ISO/IEC 14496-10 video over ITU-T Rec. H.222.0 | ISO/IEC 13818-1 streams is specified in 2.14.

2.11.2 Carriage of individual ISO/IEC 14496-2 and 14496-3 Elementary Streams in PES packets

2.11.2.1 Introduction

Individual ISO/IEC 14496-2 and 14496-3 elementary streams may be carried in PES packets as PES_packet_data_bytes. For PES packetization no specific data alignment constraints apply. For synchronization PTSs and, when appropriate, DTSs are encoded in the header of the PES packet that carries the ISO/IEC 14496 elementary stream data; for PTS and DTS encoding the same constraints apply as for ISO/IEC 13818 elementary streams. See Table 2-93 for an overview of how to carry individual ISO/IEC 14496 streams within an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream.

Table 2-93 – Carriage of individual ISO/IEC 14496 streams in ITU-T Rec. H.222.0 | ISO/IEC 13818-1

ISO/IEC 14496-2 visual	Carriage in PES packets	Stream_type = 0x10	Stream_id = '1110 xxxx'
ISO/IEC 14496-3 audio	Carriage in PES packets	Stream_type = 0x11	Stream_id = '110x xxxx'

If a PTS or DTS is present in the PES packet header it shall refer to the visual object that follows either the first VOP start code or the first still texture object startcode that commences in the PES packet. Each ISO/IEC 14496-2 video stream carried by ITU-T Rec. H.222.0 | ISO/IEC 13818-1 shall contain the information required to decode the ISO/IEC 14496-2 video stream; consequently the stream shall contain Visual Object Sequence Headers, Visual Object Headers and Video Object Layer Headers.

In case of an ISO/IEC 14496-3 elementary stream, before PES packetization the elementary stream data shall be first encapsulated in the LATM/LOAS AudioSyncStream() transport syntax defined in ISO/IEC 14496-3. If a PTS is present in the PES packet header, it shall refer to the first audio frame that follows the first syncword that commences in the payload of the PES packet.

Carriage of individual ISO/IEC 14496-2 and ISO/IEC 14496-3 elementary streams in PES packets shall be identified by appropriate stream_id and stream_type values, indicating the use of ISO/IEC 14496-2 Visual or 14496-3 Audio. In addition, such carriage shall be signalled by the MPEG-4_video descriptor or MPEG-4_audio descriptor, respectively. These descriptors shall be conveyed in the descriptor loop for the respective elementary stream entry in the Program Map Table in case of a Transport Stream or in the Program Stream Map, when present, in case of a Program Stream. ITU-T Rec. H.222.0 | ISO/IEC 13818-1 does not specify presentation of ISO/IEC 14496-2 and ISO/IEC 14496-3 elementary streams in the context of a program.

2.11.2.2 STD extensions for individual ISO/IEC 14496 elementary streams

The T-STD model includes a transport buffer TB_n and a multiplex buffer B_n prior to decoding of each individual ISO/IEC 14496 elementary stream n. Note that in the T-STD the single multiplex buffer B_n is also applied for ISO/IEC 14496-2 video, as indicated in Figure 2-4, instead of the approach with two buffers MB_n and EB_n used for ISO/IEC 13818-2 video in the T-STD. For buffers TB_n and B_n and the rate Rx_n between TB_n and B_n the following constraints apply.

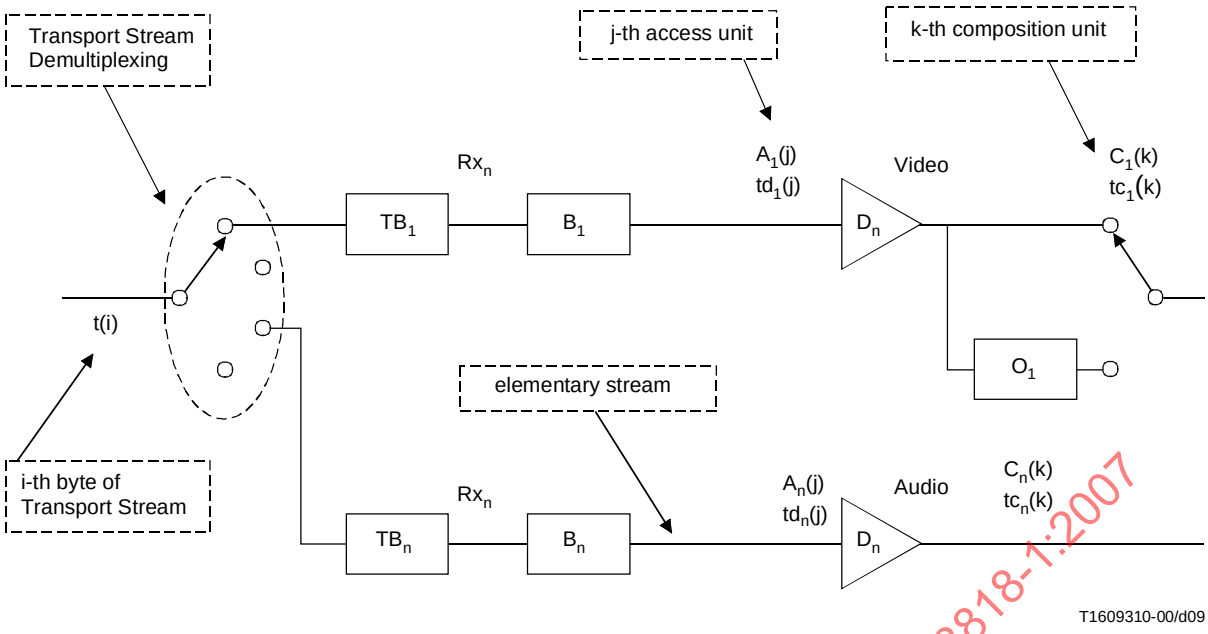

Figure 2-4 – T-STD model extensions for individual ISO/IEC 14496 elementary streams

In case of carriage of an ISO/IEC 14496-2 stream:

Size BS_n of Buffer B_n:

$$BS_n = BS_{mux} + BS_{oh} + VBV_{max}[profile,level]$$

where:

BS_{oh}, packet overhead buffering, is defined as:

$BS_{oh} = (1/750)$ seconds $\times$ max{R_{max}[profile,level], 2 000 000 bit/s}

and:

BS_{mux}, additional multiplex buffering, is defined as:

$BS_{mux} = 0.004$ seconds $\times$ max{R_{max}[profile,level], 2 000 000 bit/s}

Rate Rx_n:

$$Rx_n = 1.2 \times R_{max}[profile,level]$$

where:

VBV_{max}[profile,level] and R_{max}[profile,level] are defined in ISO/IEC 14496-2 for each profile and level. For profiles and levels for which no VBV_{max} value is specified, the size of B_n and the rate Rx_n are user defined.

In case of carriage of an ISO/IEC 14496-3 stream:

Size BS_n of Buffer B_n for ISO/IEC 14496-3 AAC audio.

$$\text{else } BS_n = BS_{mux} + BS_{dec} + BS_{oh} = 3584 \text{ bytes}$$

In this case the size of the access unit decoding buffer BS_{dec}, and the PES packet overhead buffer BS_{oh} are constrained by:

$$BS_{dec} + BS_{oh} \leq 2848 \text{ bytes}$$

A portion (736 bytes) of the 3584 byte buffer is allocated for buffering to allow multiplexing. The rest, 2848 bytes, are shared for access unit buffering BS_{dec}, BS_{oh} and additional multiplexing.

Rate Rx_n for ISO/IEC 14496-3 AAC audio same as defined for ISO/IEC 13818-7 ADTS audio in 2.4.2.3:

$$\text{else } Rx_n = 2\ 000\ 000 \text{ bit/s}$$

The P-STD model includes a multiplex buffer B_n prior to decoding of each individual ISO/IEC 14496 elementary stream n. The size BS_n of buffer B_n in the P-STD is defined by the P-STD_buffer_size field in the PES packet header.

2.11.3 Carriage of audiovisual ISO/IEC 14496-1 scenes and associated ISO/IEC 14496 streams

2.11.3.1 Introduction

This clause describes the encapsulation and signaling when an audiovisual scene represented by ISO/IEC 14496 data is carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Stream or Transport Stream. ISO/IEC 14496 content consists of the initial object descriptor and a variable number of streams such as object descriptor streams, scene description streams (carrying either BIFS-Command or BIFS-Anim access units), IPMP streams, OCI streams and audiovisual streams. Each of the ISO/IEC 14496 streams shall be contained in an SL-packetized stream and may optionally be multiplexed into a FlexMux stream, both defined in ISO/IEC 14496-1. For carriage in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Stream or Transport Stream, these SL-packetized streams and FlexMux streams shall contain encoded Object Clock Reference (OCR) and FlexMux Clock Reference (FCR) fields as specified in 2.11.3.4 and in 2.11.3.5, respectively. The SL-packetized streams or FlexMux streams are then encapsulated either in PES packets or in ISO_IEC_14496_sections prior to Transport Stream packetization and multiplexing or Program Stream multiplexing. ISO_IEC_14496_sections are built on the long format of H.222.0 | ISO/IEC 13818-1 sections.

2.11.3.2 Assignment of ES_ID values

An ISO/IEC 14496-1 scene carried over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream may associate a number of ISO/IEC 14496, ISO/IEC 13818 and other streams by the use of the ES_ID parameter. The scene and the associated streams may be carried over the same ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, but a scene may also reference streams carried elsewhere, for example over an IP network. How to identify such other means is not defined in this Specification.

ISO/IEC 14496-1 defines name scoping rules for identifiers. These rules allow the same ES_ID value to be used for two different streams within ISO/IEC 14496 content. When one or multiple ISO/IEC 14496-1 scenes are carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program, duplicate ES_ID values shall not occur within the program such that each ISO/IEC 14496 SL-packetized stream or ISO/IEC 14496-1 FlexMux channel has a unique ES_ID value in the program.

2.11.3.3 Timing of ISO/IEC 14496 scenes and associated streams

When carried over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, the object time base of each ISO/IEC 14496 stream shall be locked to the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 STC, that is:

$$\text{If } X(t) = f_{stc}(t)/f_{object}(t)$$

then the value of X(t) shall be constant at any time t.

where:

$f_{stc}(t)$ denotes the intended frequency of the STC at time t, i.e., 27 000 000 Hz

$f_{object}(t)$ denotes the frequency of the object time base at time t

The object time base of ISO/IEC 14496 streams carried over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream is conveyed as follows:

- The object time base of an SL-packetized stream carried in PES packets without the use of the FlexMux shall be conveyed by coded OCRs in the SL packet header of that stream. See 2.11.3.4.
- The object time base of SL-packetized streams carried in PES packets within a FlexMux stream shall be conveyed by FCRs in that FlexMux stream. See 2.11.3.5. Consequently, all ISO/IEC 14496 streams contained within the same FlexMux stream share the same object time base.
- The object time base of an SL-packetized stream carried in sections shall be conveyed by another ISO/IEC 14496 stream within the Transport Stream or Program Stream as indicated by the OCR_ES_ID field in the ES descriptor for that stream.

The following constraints shall apply for encoding of OCRs and FCRs in SL-packetized streams and FlexMux streams carried over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream:

- The OCRs and FCRs in each SL-packetized stream and each FlexMux stream associated to the same scene shall have the same resolution.
- The resolution of OCRs and FCRs for a scene, f_{cr}, shall have a value smaller than or equal to 90 000 Hz.
- The ratio $(f_{stc}(t)/300)/f_{cr}$, shall be an integer value larger than or equal to one. Consequently the resolution of the OCR and FCR syntax elements may only take values such as 90 000 Hz, 45 000 Hz, 30 000 Hz, 22 500 Hz, 18 000 Hz, etc.

Within the above constraints and the ISO/IEC 14496-1 constraint that the resolution f_{cr} shall represent an integer number of cycles per second, f_{cr} can be selected as appropriate for the scene.

The ISO/IEC 14496 time stamps coded in the SL packet header shall refer to instants of the object time base of the stream carried in the SL packet. The resolution of each such time stamp shall be of a factor 2^k smaller than the resolution of the OCRs or FCRs associated to the stream, with k a positive integer larger than or equal to zero. To achieve the same wrap around, the length of the time stamp fields, TimeStampLength, shall be k bit smaller than the length of the OCR or FCR field, OCRLength and FCRLength, respectively. Hence for each stream the following conditions shall apply for encoding of time stamps:

- TimeStampResolution = (OCRResolution or FCRResolution respectively)/2^k, with k a positive integer larger than or equal to zero. ISO/IEC 14496-1 requires TimeStampResolution to represent an integer number of cycles per second.
- TimeStampLength = OCRLength or FCRLength respectively – k.

The relationship between a value of the STC and the corresponding value of the object time base of a stream is established by associating PTS fields in PES packet headers with the OCR or FCR in SL packet headers and FlexMux Stream packets, respectively, as specified in 2.11.3.6 and 2.11.3.7.

2.11.3.4 Delivery timing of SL-packetized streams

To carry ISO/IEC 14496 content in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, ISO/IEC 14496-1 SL-packetized streams are used. In each SL-packetized stream carried in a PES packet without the use of FlexMux, the objectClockReference field shall be encoded as follows:

1) An objectClockReference (OCR) field shall be present in the first SL packet header of a SL-packetized stream.
2) The SL-packetized stream shall be constructed such that the time interval between the bytes containing the last bit of successive OCR fields shall be less than or equal to 0.7 s. Thus:

$$|t(i'') - t(i')| <= 0.7\,\text{s}$$

for all i′ and i″ where i' and i″ are the indexes of the bytes containing the last bit of consecutive OCR fields in the FlexMux stream.

If an objectClockReference is encoded in an SL packet header, also the instantBitrate field shall be coded.

2.11.3.5 Delivery timing of FlexMux streams

Next to SL-packetized streams also the ISO/IEC 14496-1 FlexMux tool may be used to carry ISO/IEC 14496 content in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 streams. The payload of FlexMux packets shall consist of SL packets as specified in ISO/IEC 14496-1. In each FlexMux stream carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream the fmxClockReference field shall be encoded as follows:

1) An fmxClockReference (FCR) field shall be present in the first FlexMux packet of a FlexMux stream.
2) The FlexMux stream shall be constructed such that the time interval between the bytes containing the last bit of successive FCR fields shall be less than or equal to 0.7 s. Thus:

$$|t(i'') - t(i')| <= 0.7\,\text{s}$$

for all i′ and i″ where i′ and i″ are the indexes of the bytes containing the last bit of consecutive FCR fields in the FlexMux stream.

3) All ISO/IEC 14496 time stamps within the SL-packetized streams carried within a FlexMux stream shall refer to instants of the object time base conveyed by the FCR fields in the FlexMux stream. The SL-packetized streams carried in FlexMux packets need not carry OCR fields. If OCR fields are present, they may be ignored.

2.11.3.6 Carriage of SL-packetized streams in PES packets

A single ISO/IEC 14496-1 SL-packetized stream may be mapped into a single PES stream. One and only one SL packet from an SL-packetized stream shall constitute the payload of one PES packet. PES packets that carry an SL-packetized stream shall be identified by stream_id = 0xFA in the PES packet header.

When an OCR field is coded in the SL packet header, a PTS shall be encoded in the header of the PES packet that carries such SL packet header. This PTS shall be encoded with the 33-bit value of the 90-kHz portion of the STC that corresponds to the value of the object time base at the instant in time indicated by the OCR.

The ES_ID associated to the SL-packetized stream shall be signalled by an SL descriptor as specified in 2.6.46.

2.11.3.7 Carriage of FlexMux streams in PES packets

PES packets with a payload consisting of FlexMux packets shall be identified by stream_id = 0xFB in the PES packet header. An integer number of FlexMux packets shall constitute the payload of one PES packet, i.e., the payload of a PES packet carrying a FlexMux stream shall start with a FlexMux packet header and shall end with the last byte of a FlexMux packet.

If an fmxClockReference (FCR) field is encoded in one of the FlexMux packets contained in a PES packet, then a PTS shall be encoded in the header of the PES packet that contains such FlexMux packet. This PTS shall be encoded with the 33-bit value of the 90-kHz portion of the STC that corresponds to the value of the object time base of the FlexMux stream at the instant in time indicated by the FCR. In case multiple FlexMux packets with an encoded FCR field are contained in a PES packet, the PTS shall correspond to the time indicated by the FCR in the first such FlexMux packet encountered in the payload of the PES packet.

The ES_IDs associated to each SL-packetized stream conveyed in the FlexMux stream shall be signalled by an FMC descriptor as specified in 2.6.44.

2.11.3.8 Carriage of SL packets and FlexMux packets in sections

For transport of ISO/IEC 14496 content in sections, ISO_IEC_14496_sections are defined. Only SL-packetized object descriptor streams and scene description streams shall use ISO_IEC_14496_sections. A single ISO_IEC_14496_section shall contain either an entire SL packet of an SL-packetized stream or an integer number of FlexMux packets each carrying an SL packet of the same ISO/IEC 14496-1 elementary stream.

Table 2-94 shows the syntax of ISO_IEC_14496_sections defined to convey ISO/IEC 14496-1 elementary streams, qualified by the table_id as either object descriptor or scene description stream data. Object descriptor stream data consists of an Object Descriptor Table that comprises a number of object descriptors. The Object Descriptor Table may be transmitted in multiple ISO_IEC_14496_sections. Scene description data consists of a Scene Description Table that may comprise a number of BIFS commands. The Scene Description Table may be transmitted in multiple ISO_IEC_14496_sections. It is not required that a complete table be received in order to process its payload. However, the payload of sections shall be processed in the correct order, as indicated by the value of the section_number field in the ISO_IEC_14496_section header bytes.

Table 2-94 – Section syntax for transport of ISO/IEC 14496 stream

Syntax	No. of bits	Mnemonic
ISO_IEC_14496_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
private_indicator	**1**	**bslbf**
reserved	**2**	**bslbf**
ISO_IEC_14496_section_length	**12**	**uimsbf**
table_id_extension	**16**	**uimsbf**
reserved	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
if (PMT_has_SL_descriptor(current_PID)) {		
SL_Packet()		
}		
else if (PMT_has_FMC_descriptor(current_PID)) {		
for (i = 1; i < N; i++)		
FlexMuxPacket()		
}		
else {		
for (i = 1; i < N; i++)		
reserved	**8**	**bslbf**
}		
CRC_32	**32**	**rpchof**
}		

table_id – This 8-bit field shall be set to '0x04' or '0x05' in case of an ISO_IEC_14496_section. A value of '0x04' indicates an ISO_IEC_14496_scene_description_section that carries an ISO/IEC 14496-1 scene description stream. A value of '0x05' indicates an ISO_IEC_14496_object_descriptor_section that carries an ISO/IEC 14496-1 object descriptor stream.

section_syntax_indicator – This 1-bit field shall be set to '1'.

private_indicator – This 1-bit field shall not be specified by this Specification.

ISO_IEC_14496_section_length – This 12-bit field shall specify the number of remaining bytes in the section immediately following the ISO_IEC_14496_section_length field up to the end of the ISO_IEC_14496_section. The value of this field shall not exceed 4093 (0xFFD).

table_id_extension – This 16-bit field shall not be specified by this Specification; its use and value are defined by the user.

version_number – This 5-bit field shall represent the version number of the Object Descriptor Table or Scene Description Table respectively. The version number shall be incremented by 1 modulo 32 with each new version of the table. Version control is at the discretion of the application.

current_next_indicator – This 1-bit field shall be set to 1.

section_number – This 8-bit field shall represent the number of the ISO_IEC_14496_section. The section_number field of the first ISO_IEC_14496_section of the Object Descriptor Table or the Scene Description Table shall have a value equal to 0x00. The value of section_number shall be incremented by 1 with each additional section in the table.

last_section_number – This 8-bit field shall specify the number of the last section of the Object Descriptor Table or Scene Description Table of which this section is a part.

PMT_has_SL_descriptor(current_PID) – A pseudo function that shall be true if an SL descriptor is contained in the descriptor loop in the Program Map Table for the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program element that conveys this ISO_IEC_14496_section.

SL_Packet() – A sync layer packet as specified in 10.2.2 of ISO/IEC 14496-1.

PMT_has_FMC_descriptor(current_PID) – A pseudo function that shall be true if an FMC descriptor is contained in the descriptor loop in the Program Map Table for the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program element that conveys this ISO_IEC_14496_section.

FlexMuxPacket() – A FlexMux packet as specified in 11.2.4 of ISO/IEC 14496-1.

CRC_32 – This 32-bit field shall contain the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire ISO_IEC_14496_section.

2.11.3.9 T-STD extensions

2.11.3.9.1 T-STD Model for 14496 content

Figure 2-5 shows extensions of the Transport System Target Decoder for delivery of ISO/IEC 14496 program elements encapsulated in ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Transport Streams.

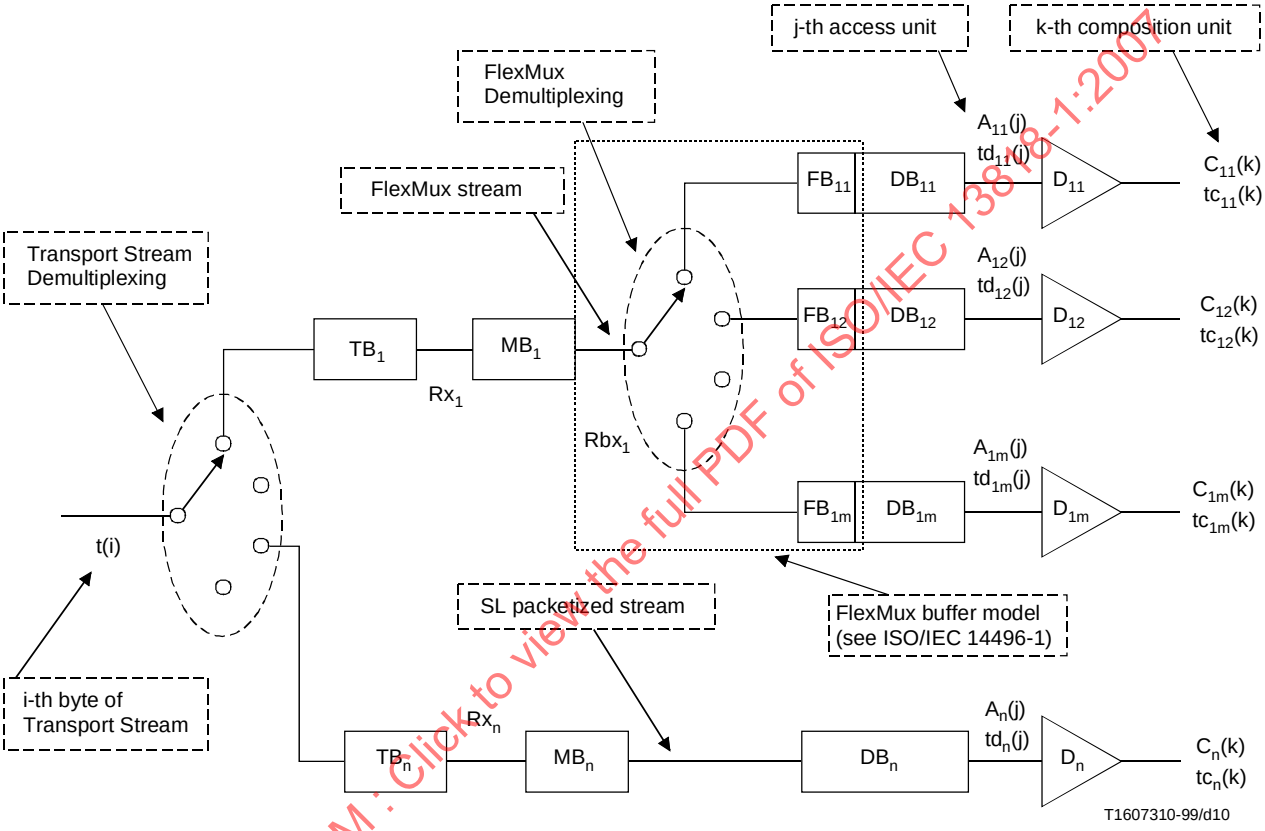

Figure 2-5 – T-STD model for ISO/IEC 14496 content

The following notation is used in Figure 2-5 and its description:

TB_n is the transport buffer.

MB_n is the multiplex buffer for FlexMux stream n or for SL-packetized stream n.

FB_{np} is the FlexMux buffer for the elementary stream in FlexMux channel p of FlexMux stream n.

DB_{np} is the decoder buffer for the elementary stream in FlexMux channel p of FlexMux stream n.

DB_n is the decoder buffer for elementary stream n.

D_{np} is the decoder for the elementary stream in FlexMux channel p of FlexMux stream n.

D_n is the decoder for elementary stream n.

Rx_n is the rate at which data are removed from TB_n.

Rbx_n is the rate at which data are removed from MB_n.

$A_{np}(j)$ is the jth access unit in elementary stream in FlexMux channel p of FlexMux stream n. $A_{np}(j)$ is indexed in decoding order.

$A_n(j)$ is the jth access unit in elementary stream n. $A_n(j)$ is indexed in decoding order.

$Td_{np}(j)$ is the decoding time, measured in seconds, in the system target decoder of the jth access unit in elementary stream in FlexMux channel p of FlexMux stream n.

$Td_n(j)$ is the decoding time, measured in seconds, in the system target decoder of the jth access unit in elementary stream n.

$C_{np}(k)$ is the kth composition unit in elementary stream in FlexMux channel p of FlexMux stream n. $C_{np}(k)$ results from decoding $A_{np}(j)$. $C_{np}(k)$ is indexed in composition order.

$C_n(k)$ is the kth composition unit in elementary stream n. $C_n(k)$ results from decoding $A_n(j)$. $C_n(k)$ is indexed in composition order.

$tc_{np}(k)$ is the composition time, measured in seconds, in the system target decoder of the kth composition unit in elementary stream in FlexMux channel p of FlexMux stream n.

$tc_n(k)$ is the composition time, measured in seconds, in the system target decoder of the kth composition unit in elementary stream n.

t(i) indicates the time in seconds at which the ith byte of the Transport Stream enters the system target decoder.

2.11.3.9.2 Processing of FlexMux streams

Complete Transport Stream packets containing data from FlexMux stream n are passed to the transport buffer for FlexMux stream n, TB_n. The size of TB_n is fixed at 512 bytes. All bytes that enter TB_n are removed from TB_n at a rate Rx_n, specified by the TB_leak_rate field in the MultiplexBuffer descriptor associated with FlexMux stream n. When there is no data in buffer TB_n, rate Rx_n is equal to zero. Duplicate Transport Stream packets are not delivered to MB_n.

In case of carriage in PES packets, the PES packet header and payload data bytes are delivered to buffer MB_n; all other bytes leaving TB_n do not enter MB_n, and may be used to control the system. In case of carriage in ISO_IEC_14496_sections, the section header, payload and CRC-32 data bytes are delivered to buffer MB_n; all other bytes do not enter MB_n and may be used to control the system. In either case, the size of MB_n shall be specified by the MB_buffer_size field in the MultiplexBuffer descriptor.

The FlexMux Stream packet bytes in buffer MB_n are all delivered to their associated FlexMux buffer at the rate specified by the field fmxRate encoded in the FlexMux stream and in compliance with the FlexMux buffer model defined in 11.2.9 of ISO/IEC 14496-1. Only FlexMux packet payload data bytes in FlexMux channel p of FlexMux stream n enter buffer FB_{np}. FlexMux packet header bytes in FlexMux channel p of FlexMux stream n are discarded and may be used to control the system. The rate specified by the fmxRate field shall be applicable for all FlexMux packets in the stream immediately following the FlexMux Clock Reference channel packet up to the next encountered FlexMux Clock Reference channel packet. When there is no FlexMux stream data present in MB_n, no data is removed from MB_n. Bytes from the PES packet header or from the ISO_IEC_14496_section header that immediately precede a FlexMux header are instantaneously removed and discarded and may be used to control the system. Bytes from the ISO_IEC_14496_section CRC-32 fields that immediately follow the last FlexMux Stream packet in the section payload are removed instantaneously and discarded and may be used to verify the integrity of the data. Bytes from the FlexMux Clock Reference channel are instantaneously removed and discarded and may be used to lock the ISO/IEC 14496 object time base to the STC. When there is no PES packet or section payload data bytes, respectively present in MB_n, no data is removed from MB_n. All data that enters MB_n leaves it. All PES packet payload bytes of stream n enter the FlexMux demultiplexer instantaneously upon leaving MB_n.

2.11.3.9.3 Definition of FlexMux Buffer, FB_{np}

For each channel p of a FlexMux stream n, the size of FlexMux buffer FB_{np} is defined using the FmxBufferSize descriptor. FlexMux packet payload bytes are transferred from buffer FB_{np} to decoder buffer DB_{np} in compliance with the FlexMux buffer model defined in 11.2.9 of ISO/IEC 14496-1. Only SL packet payload bytes in FlexMux channel p of FlexMux stream n enter buffer DB_{np}. The SL packet header bytes in FlexMux channel p of FlexMux stream n are discarded and may be used to control the system.

2.11.3.9.4 Processing of SL-packetized streams

Complete Transport Stream packets containing data from SL-packetized stream n are passed to the transport buffer for SL-packetized stream n, TB_n. All bytes that enter TB_n are removed at a rate Rx_n, specified by the TB_leak_rate field in the MultiplexBuffer descriptor. When there is no data in buffer TB_n, rate Rx_n is equal to zero. Duplicate Transport Stream packets are not delivered to MB_n.

In case of carriage in PES packets, the PES packet header and payload data bytes are delivered to buffer MB_n; all other bytes leaving TB_n do not enter MB_n, and may be used to control the system. In case of carriage in ISO_IEC_14496_sections, the section header, payload and CRC-32 data bytes are delivered to buffer MB_n; all other bytes do not enter MB_n and may be used to control the system. In either case the size of MB_n is specified by the MB_buffer_size field in the MultiplexBuffer descriptor.

The SL-packetized stream bytes in buffer MB_n are all delivered to the decoder buffer DB_n at the rate specified by the field instantBitRate encoded in the SL-packetized stream and in compliance with the System Decoder Model defined in 7.4 of ISO/IEC 14496-1. The rate specified by the instantBitRate field shall be applicable for all data bytes in the SL-packetized stream immediately following the instantBitRate field in the SL packet header up to the next encountered instantBitRate field. If there are no SL-packetized stream bytes in MB_n, no bytes are removed from MB_n. Bytes from the PES packet header or from the ISO_IEC_14496_section header that immediately precede a SL packet header are instantaneously removed and discarded and may be used to control the system. Bytes from the ISO_IEC_14496_section CRC-32 fields that immediately follow the last SL packet payload byte in the section are removed instantaneously and discarded and may be used to verify the integrity of the data. When there are no PES packet or section payload data bytes, respectively present in MB_n, no data is removed from MB_n. All data that enters MB_n leaves it. All PES packet payload bytes of stream n enter buffer DB_n instantaneously upon leaving MB_n, with the exception of the SL packet headers. Bytes from the SL packet headers do not enter DB_n and may be used to control the system. The size of decoder buffer DB_n is given by the bufferSizeDB of the DecoderConfigDescriptor defined in ISO/IEC 14496-1.

2.11.3.9.5 Buffer management

Transport streams shall be constructed so that conditions defined in this subclause are satisfied.

TB_n shall not overflow and shall be empty at least once every second. MB_n shall not overflow. FB_{np} shall not overflow. DB_{np} and DB_n shall neither underflow nor overflow. Underflow of DB_{np} occurs when one or more bytes of an access unit are not present in DB_{np} at the decoding time associated with this access unit. Underflow of DB_n occurs when one or more bytes of an access unit are not present in DB_n at the decoding time associated with this access unit.

2.11.3.10 Carriage within a Transport Stream

2.11.3.10.1 Overview

A Transport Stream may contain one or more programs, each described by a Program Map Table. ISO/IEC 14496 content can be conveyed in addition to the already defined stream types for such a program. Elements of the ISO/IEC 14496 content may be conveyed in one or more ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program elements referenced by a unique PID value within a Transport Stream. As a special case, it is possible that a program within a Transport Stream consists only of ISO/IEC 14496 program elements. ISO/IEC 14496 content associated to a program and carried in the Transport Stream shall be referenced in the Program Map Table of that program. An initial object descriptor shall be used to define an ISO/IEC 14496-1 scene; the use of this descriptor is specified in 2.11.3.10.2.

Carriage of ISO/IEC 14496 content in a PID is signalled by a stream_type value of 0x12 or 0x13 in the Program Map Table in association with that PID value. A value of 0x12 indicates carriage in PES packets. The stream_id field in the PES packet header signals whether the PES packet contains a single SL packet or a number of FlexMux packets. A stream_type value of 0x13 in the Program Map Table indicates that the program element carries an object descriptor stream or a BIFS-Command stream contained in sections. In this case the table_id in the section header indicates whether an object descriptor stream is carried in the sections or a BIFS-Command stream. See also Table 2-95. The section contains either a single SL packet or a number of FlexMux packets, as indicated by the presence of an SL descriptor or a FMC descriptor respectively in the descriptor loop of the Program Map Table for the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program element that carries the sections. When ISO/IEC 14496 content is carried, the SL descriptor and the FMC descriptor shall specify the ES_ID for each encapsulated ISO/IEC 14496 stream. When the assignment of ES_ID values changes, the Program Map Table shall be updated and the version_number of the PMT shall be incremented by 1 modulo 32. An example of a content access procedure for ISO/IEC 14496 program components within a Transport Stream is given in Annex R.

Table 2-95 – ISO/IEC defined options for carriage of an ISO/IEC 14496 scene and associated streams in ITU-T Rec. H.222.0 | ISO/IEC 13818-1

ISO/IEC 14496-1 object descriptor streams	Encapsulation in SL packets	Carriage in PES packets	Stream_type = 0x12	Stream_id = '1111 1010'
		Carriage in ISO_IEC_14496_sections	Stream_type = 0x13	Table_id = 0x05
	Encapsulation in SL packets followed by Multiplex into FlexMux packets	Carriage in PES packets	Stream_type = 0x12	Stream_id = '1111 1011'
		Carriage in ISO_IEC_14496_sections	Stream_type = 0x13	Table_id = 0x05
ISO/IEC 14496-1 scene description streams	Encapsulation in SL packets	Carriage in PES packets	Stream_type = 0x12	Stream_id = '1111 1010'
		Carriage in ISO_IEC_14496_sections	Stream_type = 0x13	Table_id = 0x04
	Encapsulation in SL packets followed by Multiplex into FlexMux packets	Carriage in PES packets	Stream_type = 0x12	Stream_id = '1111 1011'
		Carriage in ISO_IEC_14496_sections	Stream_type = 0x13	Table_id = 0x04
All other ISO/IEC 14496 streams	Encapsulation in SL packets	Carriage in PES packets	Stream_type = 0x12	Stream_id = '1111 1010'
	Encapsulation in SL packets followed by Multiplex into FlexMux packets	Carriage in PES packets	Stream_type = 0x12	Stream_id = '1111 1011'

2.11.3.10.2 Initial Object Descriptor

In case of carriage of an ISO/IEC 14496-1 scene, the ISO/IEC 14496-1 initial object descriptor serves as the initial access point to all associated streams. The initial object descriptor shall be conveyed in the IOD descriptor located in the descriptor loop immediately following the program_info_length field in the Program Map Table of the program to which the scene is associated. It contains ES_Descriptors identifying the scene description and object descriptor streams that form part of this program. It may also contain ES_Descriptors identifying one or more associated IPMP or OCI streams. Identification of streams is done by means of ES_IDs as specified in clause 8 of ISO/IEC 14496-1.

2.11.3.11 P-STD Model for 14496 content

Figure 2-6 shows the STD model when ISO/IEC 14496 systems data are carried in a Program Stream.

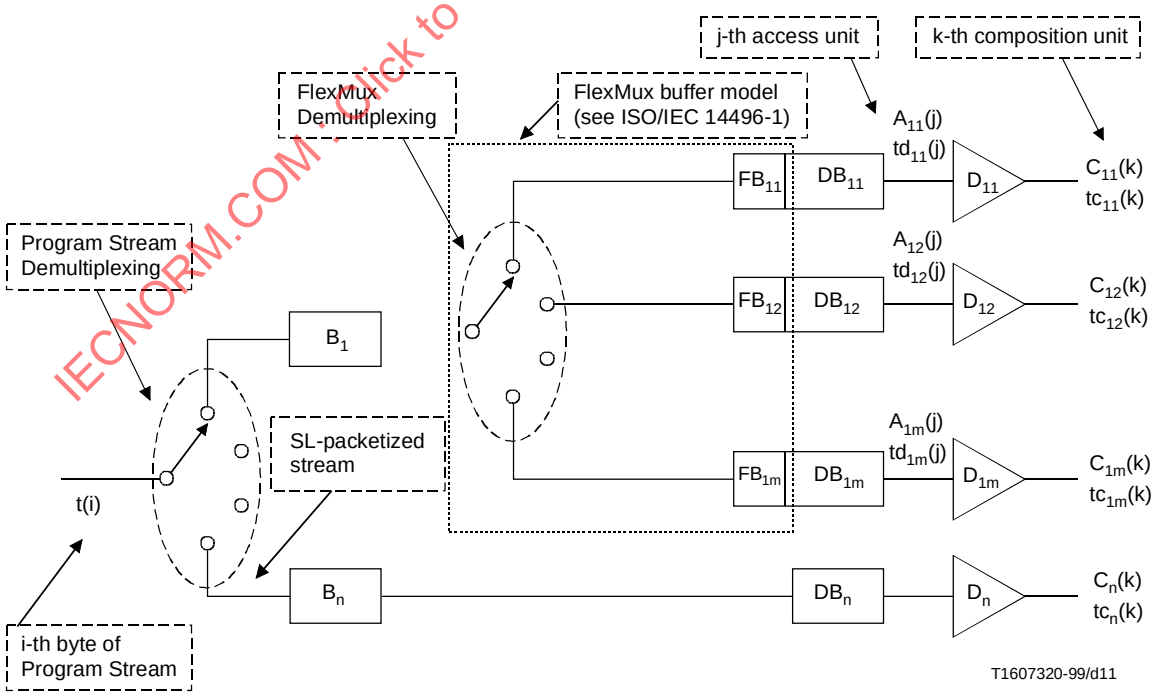

Figure 2-6 – P-STD model for ISO/IEC 14496 Systems stream

The following notation is used in Figure 2-6 and its description:

B_n is the input buffer for FlexMux stream n or for SL-packetized stream n.

FB_{np} is the FlexMux buffer for the elementary stream in FlexMux channel p of FlexMux stream n.

DB_{np} is the decoder buffer for the elementary stream in FlexMux channel p of FlexMux stream n.

DB_n is the decoder buffer for elementary stream n.

D_{np} is the decoder for elementary stream in FlexMux channel p of FlexMux stream n.

D_n is the decoder for elementary stream n.

$A_{np}(j)$ is the jth access unit in elementary stream in FlexMux channel p of FlexMux stream n. $A_{np}(j)$ is indexed in decoding order.

$A_n(j)$ is the jth access unit in elementary stream n. $A_n(j)$ is indexed in decoding order.

$Td_{np}(j)$ is the decoding time, measured in seconds, in the system target decoder of the jth access unit in elementary stream in FlexMux channel p of FlexMux stream n.

$Td_n(j)$ is the decoding time, measured in seconds, in the system target decoder of the jth access unit in elementary stream n.

$C_{np}(k)$ is the kth composition unit in elementary stream in FlexMux channel p of FlexMux stream n. $C_{np}(k)$ results from decoding $A_{np}(j)$. $C_{np}(k)$ is indexed in composition order.

$C_n(k)$ is the kth composition unit in elementary stream n. $C_n(k)$ results from decoding $A_n(j)$. $C_n(k)$ is indexed in composition order.

$tc_{np}(k)$ is the composition time, measured in seconds, in the system target decoder of the kth composition unit in elementary stream in FlexMux channel p of FlexMux stream n.

$tc_n(k)$ is the composition time, measured in seconds, in the system target decoder of the kth composition unit in elementary stream n.

t(i) indicates the time in seconds at which the ith byte of the Program Stream enters the system target decoder.

2.11.3.11.1 Processing of FlexMux streams

At the input of the STD each byte in the payload of PES packets carrying a FlexMux stream n is transferred instantaneously to buffer B_n. The i-th byte enters B_n at time t(i). PES packet header bytes do not enter buffer B_n and may be used to control the system. The size of B_n is specified by the P-STD_buffer_size field in the header of the PES packet that carries stream n.

The FlexMux stream packet bytes in buffer B_n are all delivered to their associated FlexMux buffer at the rate specified by the field fmxRate encoded in the FlexMux stream and in compliance with the FlexMux buffer model defined in 11.2.9 of ISO/IEC 14496-1. Only FlexMux packet payload data bytes in FlexMux channel p of FlexMux stream n enter buffer FB_{np}. FlexMux packet header bytes in FlexMux channel p of FlexMux stream n are discarded and may be used to control the system. The rate specified by the fmxRate field shall be applicable for all FlexMux packets in the stream up to the next encountered FlexMux Clock Reference channel packet. Bytes from the FlexMux Clock Reference channel are instantaneously removed and discarded and may be used to lock the ISO/IEC 14496 object time base to the STC. When there is no PES packet payload data present in B_n, no data is removed from B_n. All data that enters B_n leaves it. All PES packet payload bytes of stream n enter the FlexMux demultiplexer instantaneously upon leaving B_n.

2.11.3.11.2 Definition of FlexMux Buffer, FB_{np}

For each channel p of a FlexMux stream n, the size of FlexMux buffer FB_{np} is defined using the FmxBufferSize descriptor if a Program Stream Map is present in the Program Stream. FlexMux packet payload bytes are transferred from buffer FB_{np} to decoder buffer DB_{np} in compliance with the FlexMux buffer model defined in 11.2.9 of ISO/IEC 14496-1. Only SL packet payload bytes in FlexMux channel p of FlexMux stream n enter buffer DB_{np}. The SL packet header bytes in FlexMux channel p of FlexMux stream n are discarded and may be used to control the system

2.11.3.11.3 Processing of SL-packetized streams

At the input of the STD each byte in the payload of PES packets carrying an SL-packetized stream n is transferred instantaneously to buffer B_n. The i-th byte enters B_n at time t(i). PES packet header bytes do not enter buffer B_n and may be used to control the system. The size of B_n is specified by the P-STD_buffer_size field in the header of the PES packet that carries stream n. The SL-packetized stream bytes in buffer B_n are delivered to the decoder buffer DB_n at the rate specified by the field instantBitRate encoded in the SL-packetized stream and in compliance with the System Decoder Model defined in 7.4 of ISO/IEC 14496-1. The rate specified by the instantBitRate field shall be applicable for all data bytes in the SL-packetized stream up to the next encountered instantBitRate field. When there is no PES packet

payload data present in B_n, no data is removed from B_n. All data that enters B_n leaves it. All bytes of stream n enter buffer DB_n instantaneously upon leaving B_n, with the exception of the SL packet headers. Bytes from the SL packet headers do not enter DB_n and may be used to control the system. The size of decoder buffer DB_n is given by the bufferSizeDB of the DecoderConfigDescriptor defined in ISO/IEC 14496-1.

2.11.3.11.4 Buffer management

Program Streams shall be constructed so that B_n does not overflow. FB_{np} shall not overflow. DB_{np} and DB_n shall neither underflow nor overflow. Underflow of DB_{np} occurs when one or more bytes of an access unit are not present in DB_{np} at the decoding time associated with this access unit. Underflow of DB_n occurs when one or more bytes of an access unit are not present in DB_n at the decoding time associated with this access unit.

2.11.3.12 Carriage within a Program Stream

2.11.3.12.1 Overview

A Program Stream contains only one program. ISO/IEC 14496 data can be conveyed in addition to the already defined stream types for such a program. As a special case, it is also possible that a Program Stream carries only ISO/IEC 14496 data. If a Program Stream Map is present, ISO/IEC 14496 content carried in the Program Stream shall be referenced as follows. Carriage of ISO/IEC 14496-1 scenes and associated ISO/IEC 14496 streams in SL and FlexMux packets is indicated by the appropriate stream_id and by an initial object descriptor; the use of this descriptor is specified in 2.11.3.12.2. For each carried ISO/IEC 14496 stream the SL descriptor and the FMC descriptor shall specify the ES_ID. When the assignment of ES_ID values changes, the Program Stream Map, if present, shall be updated and the program_stream_map_version shall be incremented by 1 modulo 32. Note that in a Program Stream the ISO/IEC 14496 content may also be referenced by private means.

For an example of a content access procedure for ISO/IEC 14496 program components within a Program Stream, see Annex R.

2.11.3.12.2 Initial object descriptor

In case of carriage of an ISO/IEC 14496-1 scene, the ISO/IEC 14496 initial object descriptor serves as the initial access point to all associated streams. If a Program Stream Map is present in the Program Stream, the initial object descriptor shall be conveyed in the IOD descriptor that is located in the descriptor loop immediately following the program_stream_info_length field. It contains ES_Descriptors identifying the scene description and object descriptor streams of the scene that form part of this program. It may also contain ES_Descriptors identifying one or more associated IPMP or OCI streams. Identification of streams is done by means of ES_IDs as specified in clause 8 of ISO/IEC 14496-1. In a Program Stream, the initial object descriptor may also be conveyed by private means.

2.12 Carriage of metadata

2.12.1 Introduction

An ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream can carry metadata. The format of the metadata may be defined by ISO or by any other authority. This subclause defines how to carry the metadata; transport mechanisms are defined as well as metadata related-signalling, the applied metadata timing model and extensions of the STD model for decoding of metadata.

A metadata service is defined to be a coherent set of metadata of the same format delivered to a receiver for a specific purpose. Metadata services are contained in metadata streams; each metadata stream carries one or more metadata services. This Specification assumes the notion of metadata Access Units within a metadata service. The definition of a Metadata Access Unit is metadata format specific, but each metadata service is assumed to represent a concatenation (or a collection) of metadata Access Units.

When transporting a metadata service over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, a unique metadata service id is assigned to each such service. A metadata service id references uniquely a metadata service among all the metadata services available on the same Transport or Program Stream, and *not* unique *solely* within a metadata stream. The metadata service identifier is used to retrieve the metadata service and all the information needed to decode it.

Decoding of metadata may require the availability of decoder configuration data. If a metadata service carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream requires decoder configuration data for decoding, then this metadata decoder configuration data shall be carried within the same program of the same ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream.

Subclause 2.12.2 discusses metadata timing, while 2.12.3 provides an overview of tools that are defined for transport of metadata over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. The use of available transport tools is specified in

2.12.4 up to 2.12.8, and 2.12.9 specifies metadata related signalling. Finally, the STD model for metadata decoding is specified in 2.12.10.

Since many forms of metadata may be carried, it is essential to signal both the precise format and encoding of the metadata, and the semantic meaning the metadata conveys. The former is signalled by the metadata format, while the latter is signalled by the metadata application format. In other words, the metadata format conveys how the metadata shall be decoded, while the metadata application format conveys how to use the metadata, essentially which application uses the metadata. This division is important since it separates the encoding or representation of the metadata from its meaning, thereby allowing an application to be agnostic of the means by which its metadata is conveyed.

2.12.2 Metadata time-line model

Metadata may refer to time codes associated to the content, for example to indicate the beginning of a content segment. Each time indication made in the metadata refers to a certain metadata content time line specific to the actual metadata format and/or metadata application format. For example, one metadata (application) format may use UTC, while another metadata application format may use SMPTE time codes. To allow for transport of the content at any time over any media, the metadata content time line is expected but not required to be transport agnostic.

When transporting content and the associated metadata over ITU-T Rec. H.222.0 | ISO/IEC 13818-1 streams, accurate time references from the metadata to the content are to be maintained. The same is needed if the metadata is delivered over other means. To achieve this, the time line model of Figure 2-7 is assumed in this Specification.

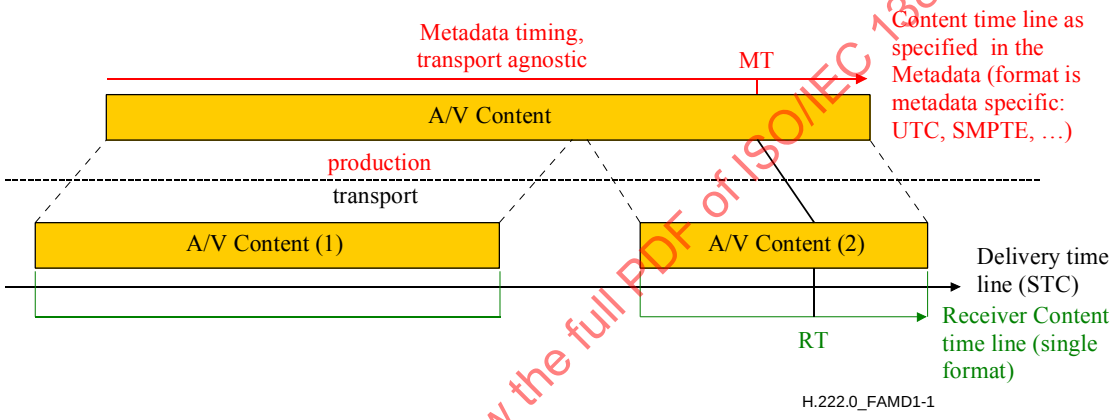

Figure 2-7 – Timing model for delivery of content and metadata

Metadata is associated to the audiovisual content, usually in a transport agnostic way, at production or any other stage prior to transport. Where needed, time information is embedded in the metadata to indicate for example specific segments within the content, using the metadata content time line used in the metadata. For example UTC or SMPTE time codes may be used. The time line format is independent of any time code that may or may not be embedded in the audiovisual stream itself. For example, the metadata time line may utilize UTC, while SMPTE time codes are embedded in the video stream.

The following requirements shall be met for each metadata stream:

- no time discontinuities shall occur in the metadata content time line;
- the metadata content time line shall be locked to the sampling clock of the content;
- each time reference in the metadata stream refers to the same metadata content time line.

At transport, a transport-specific timing is associated with the content; this is the delivery time line. In the case of transport over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, the delivery time line is provided by the System Time Clock, the STC. The content may be delivered as a contiguous piece of information, but it is also possible to interrupt the delivery of the content, for example in the case of news-flash interruptions of a program; in such and other cases time line discontinuities may occur.

When time references are used in the metadata, in the System Target Decoder (STD) these time references are to be associated unambiguously with time values in the received content. To achieve this, a receiver content time line is required. The STC can be used as the receiver content time line, but due to STC discontinuities that may occur, the STC does not necessarily offer an unambiguous time association. Therefore the NPT (Normal Play Time) concept from ISO/IEC 13818-6 DSM-CC is also available for use as the receiver content time line. In any playback mode, such as normal, reverse, slow motion, fast forward, fast backward and still picture, the NPT provides an unambiguous time

association, independent of STC discontinuities, and independent of insertions of other content. Note that a new NPT_reference_descriptor needs to be transmitted when the STC rolls over.

To maintain the accurate time references from metadata to the content, information is needed how to map a metadata time, MT, defined on the metadata content time line to the corresponding receiver time, RT, of the receiver content time line. This is achieved by providing the offset in time (in 90-kHz units) between the metadata content time line and the receiver content time line. The offset is provided in the content labelling descriptor. The offset conveys the value of the metadata time base at the instant in time at which the receiver content time base reaches a specified value. See also Figure 2-7.

The timing in metadata systems may refer to a specific picture or audio frame, for example using SMPTE time codes. The offset in time between the metadata content time line and the receiver content time line is expressed in units of 90 kHz, and consequently the metadata time reference will translate into a 90-kHz value in receivers. To accommodate for inaccuracies, receivers shall assume that when reference is made to a picture or audio frame the closest match shall be used. For example, the translated 90-kHz metadata time reference shall be matched with the picture or frame whose PTS value is closest to the translated value.

When using NPT, during playback in any mode at any point in time the offset remains constant between the metadata time base and the NPT time base. As long as neither STC discontinuities nor insertions with other content occur, the same is true for the offset in time between the metadata time base and the STC time base, but only in normal playback mode. For privately defined time lines the offset is also required to be constant, but possibly within constraints not defined in this Specification.

When synchronous transport of metadata is applied in PES packets or by using the synchronized DSM-CC download protocol, PTSs are assigned to the metadata. Such PTS may for example indicate the point in time at which the metadata becomes valid. This implies *a priori* knowledge of how to associate the metadata to the delivery timing. However, synchronously transported metadata may also contain time references, which are to be mapped from the metadata content time line to the receiver content time line using the specified offset between both time lines. See also Figure 2-8.

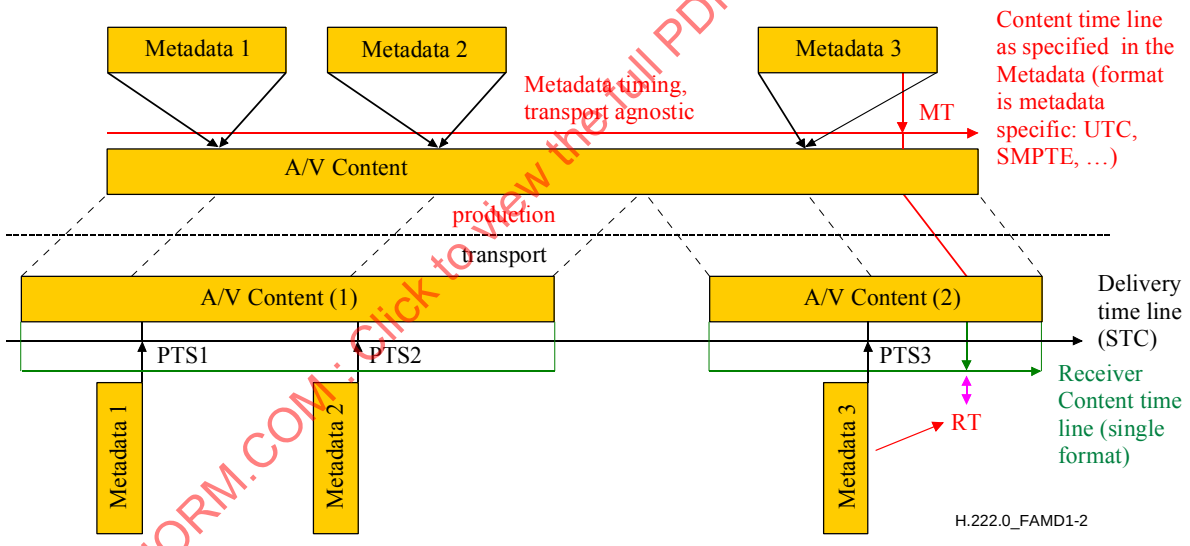

Figure 2-8 – Delivery of metadata in PES packets

2.12.3 Options for transport of metadata

To acknowledge the very diverse characteristics of metadata, a variety of tools is defined to transport the metadata over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream.

This Specification defines two tools for synchronous delivery of the metadata:

- carriage in PES packets;
- use of DSM-CC synchronized download protocol.

In addition, this Specification defines three tools for asynchronous delivery of metadata:

- carriage in metadata sections;
- use of DSM-CC data carousels;
- use of DSM-CC object carousels.

Note that some of the asynchronous transport options support carousels and file structures. The choice of transport tool depends on the requirements that apply to the delivery of the metadata, and the requirements of the tools, as described in the following subclauses.

Metadata may also be carried by private means such as PES packets with stream id value 0xBD or 0xBF (private_stream_id_1 or private_stream_id_2) or private sections. This Specification does not specify how to use private means for carriage of metadata, but allows for signalling of such metadata using the descriptors defined in 2.6.56 up to 2.6.63.

The basic referencing of metadata services is the same for all tools, using the metadata service id. However, there are differences per tool. When PES packets, metadata sections, or synchronized DSM-CC download sections are used, data from each metadata service is explicitly signalled within a metadata stream, using the metadata_service_id field. However, when using DSM-CC carousels, this signalling is left at the discretion of metadata applications. Note that this Specification allows for carriage of a metadata service in a DSM-CC carousel, but does not constrain how many metadata services can be carried in one DSM-CC carousel.

Metadata decoder configuration data is signalled explicitly when carried in a metadata descriptor, in PES packets with stream_type 0x15 and stream_id 0xFC, in metadata sections or in synchronized DSM-CC download sections. When metadata decoder configuration data is carried in a DSM-CC carousel, the signalling of such data is required, but not defined by this Specification; instead, such signalling is left at the discretion of applications.

2.12.4 Use of PES packets to transport metadata

PES packets provide a mechanism for synchronous transport of metadata. By means of the PTS in the PES packet header the metadata access units are associated to a certain instant of the STC, without the need for time references in the metadata. This implies *a priori* knowledge of how to associate the metadata to the delivery timing. Specific stream_id and stream_type values are assigned to signal PES packets carrying metadata; see 2.12.9.

When using PES packets with a stream_type of 0x15 and a stream_id of 0xFC to transport the metadata, a Metadata Access Unit Wrapper shall be used as the tool to align PES packets and the metadata Access Units, using metadata_AU_cells. This allows random access indication, whose meaning depends on the format of the metadata, and a cell sequence counter to identify loss of metadata_AU_cells. Each metadata Access Unit is carried and, if appropriate, fragmented in one or more metadata_AU_cells. In each PES packet that carries metadata, the first PES_packet_data_byte shall be the first byte of a Metadata_AU_cell. For each metadata Access Unit contained in the same PES packet, the PTS in the PES header applies. The PTS signals the time at which the metadata Access Units are decoded instantaneously and removed from buffer B_n in the STD. Note that the relationship between a decoded metadata Access Unit and audiovisual content is beyond the scope of this Specification.

A PES packet may contain a single metadata_AU_cell. This is useful if a metadata Access Unit does not fit into a single PES packet, in which case the fragmentation of the metadata Access Unit is handled by the metadata_AU_cell.

When metadata is carried by PES packets in a Program Stream, and if a Program Stream Map is applied in that Program Stream, then the Program Stream Map shall specify which PES packets contain the associated metadata.

2.12.4.1 Metadata Access Unit Wrapper

The metadata Access Unit Wrapper shall be used when carrying metadata Access Units in PES packets with a stream_type of 0x15 and a stream_id value of 0xFC or in synchronized DSM-CC download sections of stream_type 0x19. The wrapper defines a structure consisting of a concatenated number of Metadata_AU_cells. By coding the size of the contained metadata in each metadata_AU_cell, metadata agnostic parsing is possible in receivers: the parser can retrieve the metadata and provide it to a metadata decoder without *a priori* knowledge on any detail of the metadata. The Metadata_AU_cell shall be aligned with the transport; that is the first byte of the payload of the PES packet or synchronized DSM-CC download section shall be the first byte of a Metadata_AU_cell.

If a metadata Access Unit does not fit entirely into a metadata_AU_cell, then the metadata Access Unit shall be fragmented into multiple metadata_AU_cells, where the fragmentation_indication in each such metadata_AU_cell signals that the metadata_AU_cell contains a fragment.

To each Metadata_AU_cell that is contained in the same PES packet or synchronized download section, the PTS as coded in the header of the PES packet or synchronized download section, respectively, applies.

Table 2-96 – Metadata Access Unit Wrapper

Syntax	No. of bits	Mnemonic

Metadata_AU_wrapper () {
for (i = 0; i < N; i++){
Metadata_AU_cell ()
}
}		

Table 2-97 – Metadata AU cell

Syntax	No. of bits	Mnemonic
Metadata_AU_cell () {		
metadata_service_id	**8**	**uimsbf**
sequence_number	**8**	**uimsbf**
cell_fragment_indication	**2**	**bslbf**
decoder_config_flag	**1**	**bslbf**
random_access_indicator	**1**	**bslbf**
reserved	**4**	**bslbf**
AU_cell_data_length	**16**	**uimsbf**
for (i = 0; I < AU_cell_data_length; i++){		
AU_cell_data_byte	**8**	**bslbf**
}		
}		

metadata_service_id: This 8-bit field identifies the metadata service associated to the metadata Access Unit carried in this metadata AU cell.

sequence_number: This 8-bit field specifies the sequence number of the metadata_AU_cell. This number increments by one for each successive metadata_AU_cell constituting the metadata_AU_wrapper, independent of the coded value of the metadata_service_id.

cell_fragment_indication: This 2-bit field conveys information on the metadata Access Unit carried in this metadata_AU_cell, corresponding to Table 2-98.

Table 2-98 – Cell fragment indication

Value	Description
11	A single cell carrying a complete metadata Access Unit.
10	The first cell from a series of cells with data from one metadata Access Unit.
01	The last cell from a series of cells with data from one metadata Access Unit.
00	A cell from a series of cells with data from one metadata Access Unit, but neither the first nor the last one.

random_access_indicator: This 1-bit field, when coded with the value '1', indicates that the metadata carried in this metadata_AU_cell represents an entry point to the metadata service where decoding is possible without information from previous metadata_AU_cells. The meaning of a random access point is defined by the format of the metadata.

decoder_config_flag: This 1-bit field signals the presence of decoder configuration information in the carried metadata Access Unit. Note that this does not preclude the presence of metadata in the Access Unit next to decoder configuration data.

AU_cell_data_length: This 16-bit field specifies the number of AU_cell_data_bytes immediately following.

AU_cell_data_byte: This 8-bit field contains contiguous bytes from a metadata Access Unit.

2.12.5 Use of the DSM-CC synchronized download protocol to transport metadata

For synchronized transport, in addition to PES packets, the DSM-CC synchronized download protocol can be used. When using synchronized DSM-CC download sections to transport the metadata, the Metadata Access Unit Wrapper defined in 2.12.4.1 shall be used as the tool to encapsulate metadata Access Units. This allows random access indication, whose meaning depends on the format of the metadata, and a cell sequence counter to identify loss of metadata_AU_cells. In each DSM-CC synchronized download section that carries metadata, the first byte of the payload shall be the first byte of a Metadata_AU_cell. For each metadata Access Unit contained in the same DSM-CC synchronized download section, the PTS in the section header applies. The PTS signals the time at which the metadata

Access Units are decoded instantaneously and removed from buffer B_n in the STD. Note that the relationship between a decoded metadata Access Unit and audiovisual content is beyond the scope of this Specification. A specific stream_type value (as detailed in Table 2-34) is assigned to signal carriage of metadata in DSM-CC synchronized download sections.

2.12.6 Use of metadata sections to transport metadata

If asynchronous transport of metadata Access Units without a carousel delivery mechanism is needed, metadata sections can be utilized. The syntax and semantics of metadata sections are defined in this subclause. Each metadata section shall carry either one complete metadata Access Unit or a single part of one metadata Access Unit, as signalled by the section_fragment_indication field.

For transport in metadata sections, the metadata Access Units are structured in one or more Metadata Tables. Each Metadata Table contains one or more complete metadata Access Units from one or more metadata services. Conceptually, the transport mechanism of Metadata Tables is comparable to the transport mechanism of Program Map Tables and Program Association Tables. Each Metadata Table may be made up of multiple metadata sections. Each Metadata Table may contain metadata from multiple metadata services.

Specific stream_type and table_id values are assigned to metadata sections. Metadata decoder configuration data can also be carried in sections, signalled by a metadata description value, as assigned by the metadata decoder configuration descriptor.

Table 2-99 – Section syntax for transport of metadata

Syntax	No. of bits	Mnemonic
Metadata_section() {		
table_id	**8**	**uimsbf**
section_syntax_indicator	**1**	**bslbf**
private_indicator	**1**	**bslbf**
random_access_indicator	**1**	**bslbf**
decoder_config_flag	**1**	**bslbf**
metadata_section_length	**12**	**uimsbf**
metadata_service_id	**8**	**uimsbf**
reserved	**8**	**bslbf**
section_fragment_indication	**2**	**bslbf**
version_number	**5**	**uimsbf**
current_next_indicator	**1**	**bslbf**
section_number	**8**	**uimsbf**
last_section_number	**8**	**uimsbf**
for (i = 1; i < N; i++){		
metadata_byte	**8**	**bslbf**
}		
CRC_32	**32**	**rpchof**
}		

table_id: The table_id is an 8-bit field that shall be set to '0x06' for each metadata section.

section_syntax_indicator: This 1-bit field shall be set to '1'.

private_indicator: This 1-bit field is not specified by this Specification.

random_access_indicator: This 1-bit field, when coded with the value '1', indicates that the metadata carried in this metadata section represents an access point to the metadata service where decoding is possible without information from previous metadata sections. The meaning of a random access point is defined by the format of the metadata.

decoder_config_flag: This 1-bit field, when coded with the value '1', indicates that decoder configuration information is present in the metadata Access Unit carried in this metadata section.

metadata_section_length: This 12-bit field shall specify the number of remaining bytes in the section immediately following the metadata_section_length field, and including the CRC. The value of this field shall not exceed 4093 (0xFFD).

metadata_service_id: This 8-bit field identifies the metadata service associated to the metadata Access Unit carried in this metadata section. Each Metadata Table may contain metadata from multiple metadata services.

section_fragment_indication: This 2-bit field conveys information on the fragmentation of the metadata Access unit carried in this metadata section, corresponding to Table 2-100.

Table 2-100 – Section fragment indication

Value	Description
11	A single metadata section carrying a complete metadata Access Unit.
10	The first metadata section from a series of metadata sections with data from one metadata Access Unit.
01	The last metadata section from a series of metadata sections with data from one metadata Access Unit.
00	A metadata section from a series of metadata sections with data from one metadata Access Unit, but neither the first nor the last one.

version_number: This 5-bit field is the version number of the whole Metadata Table. The version number shall be incremented by 1 modulo 32 whenever the information contained within the Metadata Table changes. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable Metadata Table. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable Metadata Table.

current_next_indicator: A 1-bit field, which when set to '1' indicates that the Metadata Table sent is currently applicable. When the bit is set to '0', it indicates that the Metadata Table sent is not yet applicable and shall be the next Metadata Table to become valid.

section_number: This 8-bit field gives the number of the metadata section. The section_number of the first section in a Metadata Table shall be 0x00. The section_number shall be incremented by 1 with each additional section in this Metadata Table.

last_section_number: This 8-bit field specifies the number of the last section (that is, the section with the highest section_number) of the complete Metadata Table of which this section is a part.

metadata_byte: This 8-bit contains contiguous bytes from a metadata Access Unit.

CRC_32: This 32-bit field shall contain the CRC value that gives a zero output of the registers in the decoder defined in Annex A after processing the entire metadata_section.

2.12.7 Use of the DSM-CC data carousel to transport metadata

The DSM-CC tools as defined in ISO/IEC 13818-6 for Data Carousels can be used if a carousel delivery mechanism is required without the need to express the hierarchical organization of the metadata structure in the transport mechanism. Information on the carousel in which the metadata is contained, is included in the metadata descriptor defined in 2.6.60 and 2.6.62. A specific stream_type value is assigned to signal carriage of metadata in the DSM-CC data carousel. Note that signalling of metadata services within a DSM-CC data carousel is required, but not defined by this Specification.

2.12.8 Use of the DSM-CC object carousel to transport metadata

If a carousel delivery mechanism is required with the capability to express the hierarchical organization of the metadata structure in the transport, then the DSM-CC tools and file structures as defined in ISO/IEC 13818-6 for User to User Object Carousels can be used. These file structures provide the tools to structure the metadata as deemed appropriate for efficient parsing of the metadata and for expressing the hierarchical organization of the metadata. Information needed to identify the carousel in which the metadata is contained, is included in the metadata descriptor defined in 2.6.60 and 2.6.61. This may be the IOP:IOR() as defined in 11.3.1 and 5.7.2.3 of ISO/IEC 13818-6 DSM-CC. A specific stream_type value is assigned to signal carriage of metadata in the DSM-CC object carousel. Note that signalling of metadata services within a DSM-CC object carousel is required, but not defined by this Specification.

2.12.9 Metadata-related signalling

Metadata-related signalling covers four distinct areas:

- signalling of metadata services and streams;
- signalling of content for use by a metadata system;
- association of metadata to content; and
- signalling of decoder configuration data.

2.12.9.1 Signalling of metadata services and streams

Carriage of metadata is signalled by a stream_type value in the inclusive range between 0x15 and 0x19, specifying which of the five methods described in 2.12.4 to 2.12.8 is used to transport the metadata, and if appropriate, by a stream_id value of 0xFC indicating a metadata stream.

To uniquely identify a metadata service a metadata_service_id value is assigned to each such service by the transport; the assigned value shall be unique within the Transport or Program Stream carrying the metadata service. If the metadata is carried in PES packets with a stream_id of 0xFC, or in metadata sections, or in ISO/IEC 13818-6 synchronized download sections, the assigned metadata_service_id value is signalled explicitly in the header of the metadata_AU_cell or the metadata section. If a ISO/IEC 13818-6 carousel is used to carry the metadata, then the signalling of metadata services is left to the application. The metadata descriptor specifies the format of the metadata and provides information on the decoder configuration data, and is linked to the metadata service by carrying information on the metadata service it is associated with.

2.12.9.2 Signalling of content for use by a metadata system

In 2.6.56 and 2.6.57, a content labelling descriptor is defined that can be used to assign a metadata application format specific reference, the content_reference_id_record, to audiovisual or any other content carried over an MPEG-2 Transport Stream or Program Stream. The content_reference_id_record can be used by the metadata system as a label to refer to such content. The content may represent, for example, a program or a stream or segments thereof. The content labelling descriptor also provides information on the content time base used for time referencing from the metadata, including the constant offset in time between the metadata time base and the applied content time base. The descriptor allows carriage of private data. The metadata_application_format may define constraints on the content_reference_record, such as constraints on the time period during which it is valid.

2.12.9.3 Association of metadata to content

In 2.6.58 and 2.6.59 the metadata pointer descriptor is defined to associate a single metadata service to audiovisual or any other content in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. The metadata is associated to the content within the context as defined by the location of the descriptor. In a transport stream, the descriptor may be located in the PMT in the descriptor loop for either the program or an elementary stream, but may also be located in tables not defined in this Specification, such as tables describing bouquets of broadcast services.

The metadata pointer descriptor points from the content's context to the metadata service associated to that content. The descriptor provides the value of the metadata_service_id that is assigned to the associated metadata service, as well as one or more locations of the associated metadata. The location may for example be within the same Transport Stream as the content, or within another Transport Stream, but also at a non-ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream location such as the Internet.

2.12.9.4 Signalling decoder configuration data

Decoding of metadata may require the availability of metadata decoder configuration data. If needed, decoder configuration data shall be contained in one of the metadata services in the same program in the same ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream as the metadata service. If decoder configuration data is needed to decode a metadata service, then the metadata descriptor either carries such data or provides the information on retrieval of the decoder configuration data from the same or another metadata service. In a transport stream such other service can be found by searching in the PMT for a metadata_descriptor with the metadata_service_id as specified in the decoder_config_metadata_service_id field (and with the same metadata_format and the same metadata_application format).

2.12.9.5 Overview of metadata signalling

Figure 2-9 provides an example of metadata signalling, in which a single program carries the content (or essence), the "content program", and the metadata is carried in a separate program, the "metadata program". In this example, the metadata program and the content program exist on the same transport stream.

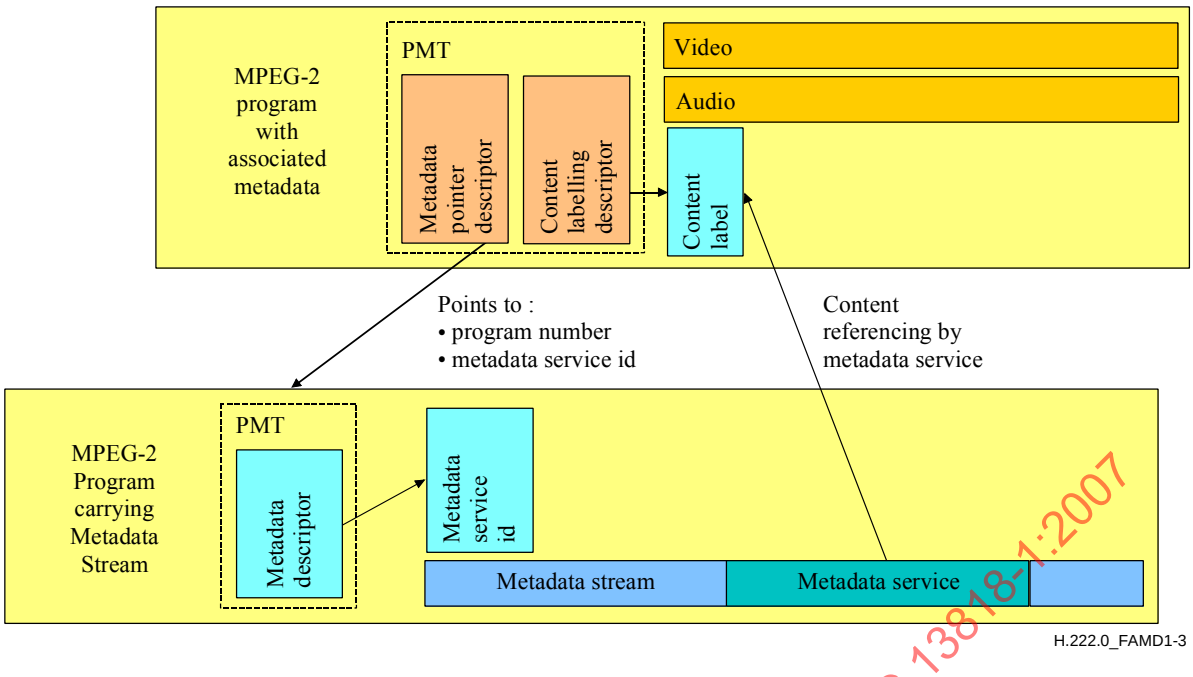

Figure 2-9 – Metadata signalling and referencing

In the content program there are two metadata-related descriptors, the content_labeling descriptor and the metadata_pointer descriptor. The content_labeling descriptor associates a label, illustrated in the diagram by "content label" and encoded in the descriptor in the content_reference_id fields, with the content. The label can than be used by the metadata service to refer to the essence, either in whole, in part, or by a time-described segment. For example, the content_labeling descriptor could provide the label "News of 1/1/02", and the metadata could then refer to a specific story item in the "News of 1/1/02", for example by providing the specific timing of the story item.

The metadata pointer descriptor provides information of where the metadata service can be found for the given content. In this example, the metadata is carried in a separate program, but it would be equally valid to have the metadata carried in the same program as the content, or provided by some means beyond the scope of this Specification, for instance from a URL. This descriptor also provides the metadata service id value that is assigned to the metadata service. This is required since a metadata stream could carry multiple metadata services for many different programs and each program needs to be able to uniquely identify its own metadata service.

In the metadata program, the metadata descriptor signals to which metadata service within a metadata stream it applies. If used, the metadata descriptor provides details of where to find the decoder configuration information.

Upon identifying a metadata pointer descriptor in the PMT by a receiver decoding the content program, the receiver retrieves the metadata descriptor from the metadata program. If needed first the decoder configuration data is retrieved, then the decoder is configured accordingly, after which the metadata service can start being decoded.

2.12.10 STD model for metadata

The STD model specifies normative constraints on ITU-T Rec. H.222.0 | ISO/IEC 13818-1 streams that carry metadata. For decoding of metadata in the STD, the regular T-STD and P-STD models are applicable with buffer B_n, input rate Rx_n of the metadata into B_n and output rate $R_{metadata}$ out of B_n and into $D_{metadata}$, the metadata decoder. See Figure 2-10.

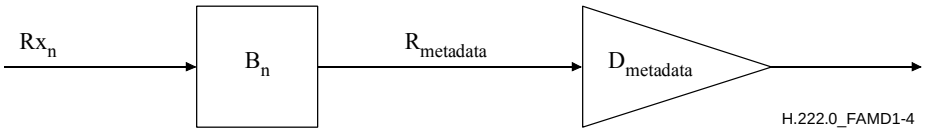

Figure 2-10 – Metadata decoding in the STD

The metadata enters buffer B_n at rate Rx_n. In the P-STD, rate Rx_n equals the rate of the program stream. In the T-STD, rate Rx_n is the rate out of TB_n and equal to the rate defined by the metadata_input_leak_rate field in the metadata STD descriptor. The size BS_n of buffer B_n is equal to the size defined in the metadata_buffer_size field in the metadata STD

descriptor. In case of synchronous delivery, metadata decoding is instantaneous and controlled by PTSs. At decode time, that is when the STC equals the PTS, the associated metadata is removed instantaneously from B_n. In case of asynchronous delivery, the metadata is removed from B_n at a rate $R_{metadata}$ equal to the rate defined by the metadata_output_leak_rate field in the metadata STD descriptor. Buffer B_n shall not overflow.

Note that the STD model defines constraints on the delivery of the metadata, without specifying any constraint on the timing used in the metadata.

2.13 Carriage of ISO 15938 data

2.13.1 Introduction

Carriage of metadata over an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream as defined in 2.12 allows for carriage of ISO 15938 data by appropriate coding of the metadata_format field. In this subclause, for the purpose to transport ISO 15938 data, a specific instance is defined. Carriage of ISO 15938 data shall meet each requirement defined in 2.12, but in addition the requirements defined in this subclause shall apply for transport of ISO 15938 data.

2.13.2 ISO 15938 decoder configuration data

Decoding of ISO 15938 data requires the availability of decoder configuration data. Consequently, when ISO 15938 data is carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, then the metadata descriptor shall signal carriage of associated decoder configuration data in the same ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream by coding a value of the decoder_config_flags of either '001' or '010' or '011' or '100'.

2.14 Carriage of ITU-T Rec. H.264 | ISO/IEC 14496-10 video

2.14.1 Introduction

This Specification defines the carriage of ITU-T Rec. H.264 | ISO/IEC 14496-10 elementary stream within ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems, both for program and transport streams. Typically, an ITU-T Rec. H.264 | ISO/IEC 14496-10 stream will be an element of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 program, as defined by the PMT in a Transport Stream and the PSM in a Program Stream. The carriage and buffer management of AVC video streams is defined using existing parameters from this Recommendation | International Standard such as PTS and DTS, as well as information present within an AVC video stream.

Carriage of AVC video streams in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream defines accurate mapping between STD parameters and HRD parameters that may be present in an AVC video stream. Requirements are defined for the presence of HRD parameters in the AVC video stream, to ensure that it can be verified whether each STD requirement is met for each AVC video stream carried in a transport stream or a program stream.

NOTE 1 – Though the timing information present in the AVC video stream may not use a 90-kHz clock, the PTS and DTS timestamps need to be expressed in units of 90 kHz.

When an ITU-T Rec. H.264 | ISO/IEC 14496-10 stream is carried in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream, the ITU-T Rec. H.264 | ISO/IEC 14496-10 coded data shall be contained in PES packets. The ITU-T Rec. H.264 | ISO/IEC 14496-10 coded data shall comply with the byte stream format defined in Annex B of ITU-T Rec. H.264 | ISO/IEC 14496-10, with the following constraints:

- Each AVC access unit shall contain an access unit delimiter NAL Unit;

  NOTE 2 – ITU-T Rec. H.264 | ISO/IEC 14496-10 requires that an access unit delimiter NAL Unit, if present, is the first NAL Unit within an AVC access unit. Access unit delimiter NAL Units simplify the ability to detect the boundary between pictures; they avoid the need to process the content of slice headers, and they are particularly useful for the Baseline and Extended profiles where slice order can be arbitrary.

- Each byte stream NAL Unit that carries the access unit delimiter shall contain exactly one zero_byte syntax element.

  NOTE 3 – The syntax and semantics of byte stream NAL units are defined in Annex B of ITU-T Rec. H.264 | ISO/IEC 14496-10.

- All Sequence and Picture Parameter Sets (SPS and PPS) necessary for decoding the AVC video stream shall be present within that AVC video stream.

  NOTE 4 – ITU-T Rec. H.264 | ISO/IEC 14496-10 also allows delivery of SPS and PPS by external means. This Specification does not provide support for such delivery, and therefore requires SPS and PPS to be carried within the AVC video stream.

- Each AVC video sequence that contains hrd_parameters() with the low_delay_hrd_flag set to '1', shall carry VUI parameters in which the timing_info_present_flag shall be set to '1'.

NOTE 5 – If the low_delay_hrd_flag is set to '1', then buffer underflow is allowed to occur in the STD model; see 2.14.3 and 2.14.4. Setting the timing_info_present_flag to '1' ensures that the AVC video stream contains sufficient information to determine the DPB output time and the CPB removal time of AVC access units, also in case of underflow.

To provide display specific information such as aspect_ratio, it is strongly recommended that each AVC video stream carries VUI parameters with sufficient information to ensure that the decoded AVC video stream can be displayed correctly by receivers.

2.14.2 Carriage in PES packets

ITU-T Rec. H.264 | ISO/IEC 14496-10 Video is carried in PES packets as PES_packet_data_bytes, using one of the 16 stream_id values assigned to video, while signalling the ITU-T Rec. H.264 | ISO/IEC 14496-10 Video stream by means of the assigned stream-type value in the PMT or PSM (see Table 2-34). The highest level that may occur in an AVC video stream as well as a profile that the entire stream conforms to should be signalled using the AVC video descriptor. If an AVC video descriptor is associated with an AVC video stream, then this descriptor shall be conveyed in the descriptor loop for the respective elementary stream entry in the Program Map Table in case of a Transport Stream or in the Program Stream Map, when PSM is present, in case of a Program Stream. This Recommendation | International Standard does not specify presentation of ITU-T Rec. H.264 | ISO/IEC 14496-10 streams in the context of a program.

For PES packetization, no specific data alignment constraints apply. For synchronization and STD management, PTSs and, when appropriate, DTSs are encoded in the header of the PES packet that carries the ITU-T Rec. H.264 | ISO/IEC 14496-10 video elementary stream data. For PTS and DTS encoding, the constraints and semantics apply as defined in 2.4.3.7 and 2.7.

2.14.3 STD extensions

2.14.3.1 T-STD extensions

The T-STD model includes a transport buffer TB_n and a multiplex buffer MB_n prior to buffer EB_n for decoding of each ITU-T Rec. H.264 | ISO/IEC 14496-10 video elementary stream n. See Figure 2-11.

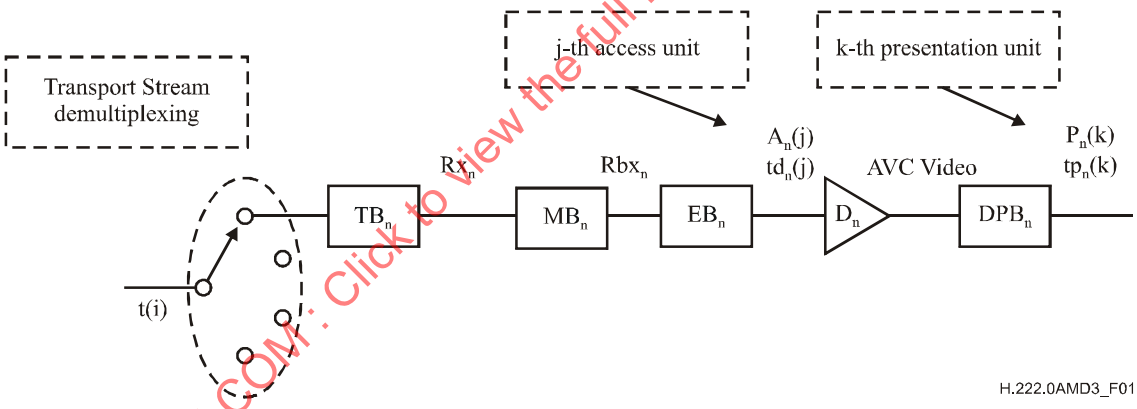

Figure 2-11 – T-STD model extensions for ITU-T Rec. H.264 | ISO/IEC 14496-10 video

DPB_n buffer management

Carriage of an AVC video stream over ITU-T Rec. H.222.0 | ISO/IEC 13818-1 does not impact the size of buffer DPB_n. For decoding of an AVC video stream in the STD the size of DPB_n is as defined in ITU-T Rec. H.264 | ISO/IEC 14496-10. The DPB buffer shall be managed as specified in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10 (clauses C.2 and C.4). A decoded AVC access unit enters DPB_n instantaneously upon decoding of the AVC access unit, hence at the CPB removal time of the AVC access unit. A decoded AVC access unit is presented at the DPB output time. If the AVC video stream provides insufficient information to determine the CPB removal time and the DPB output time of AVC access units, then these time instants shall be determined in the STD model from PTS and DTS timestamps as follows:

1) The CPB removal time of AVC access unit n is the instant in time indicated by DTS(n) where DTS(n) is the DTS value of AVC access unit n.
2) The DPB output time of AVC access unit n is the instant in time indicated by PTS(n) where PTS(n) is the PTS value of AVC access unit n.

NOTE 1 – AVC video sequences in which the low_delay_hrd_flag in hrd parameters() is set to 1 carry sufficient information to determine the DPB output time and the CPB removal time of each AVC access unit. Hence for AVC access units for which STD

underflow may occur, the CPB removal time and the DPB output time are defined by HRD parameters, and not by DTS and PTS timestamps.

TB_n, MB_n and EB_n buffer management

The input to buffer TB_n and its size TBS_n are specified in 2.4.2.3. For buffers MB_n and EB_n, and for the rate Rx_n between TB_n and MB_n and the rate Rbx_n between MB_n and EB_n the following constraints apply for carriage of an ITU-T Rec. H.264 | ISO/IEC 14496-10 stream:

Size EBS_n of buffer EB_n:

EBS_n = cpb_size

Where cpb_size is the size CpbSize[cpb_cnt_minus1] of the CPB for the byte stream format signalled in the NAL hrd_parameters() carried in VUI parameters in the AVC video stream. If NAL hrd_parameters() are not present in the AVC video stream, then the cpb_size shall be the size defined as 1200 × MaxCPB in Annex A of ITU-T Rec. H.264 | ISO/IEC 14496-10 for the level of the AVC video stream.

Size MBS_n of Buffer MB_n:

$$MBS_n = BS_{mux} + BS_{oh} + 1200 \times MaxCPB[level] - cpb\_size$$

where BS_{oh}, packet overhead buffering, is defined as:

$$BS_{oh} = (1/750) \text{ seconds} \times \max\{1200 \times MaxBR[level], 2\,000\,000 \text{ bit/s}\}$$

and BS_{mux}, additional multiplex buffering, is defined as:

$$BS_{mux} = 0.004 \text{ seconds} \times \max\{1200 \times MaxBR[level], 2\,000\,000 \text{ bit/s}\}$$

where MaxCPB[level] and MaxBR[level] are defined for the byte stream format in Table A.1 (Level Limits) in ITU-T Rec. H.264 | ISO/IEC 14496-10 for the level of the AVC video stream, and

where cpb_size is the size CpbSize[cpb_cnt_minus1] of the CPB for the byte stream format signalled in the NAL hrd_parameters() carried in VUI parameters in the AVC video stream. If NAL hrd_parameters() are not present in the AVC video stream, then the cpb_size shall be the size 1200 × MaxCPB defined in Annex A of ITU-T Rec. H.264 | ISO/IEC 14496-10 for the level of the AVC video stream.

Rate Rx_n:

when there is no data in TB_n then Rx_n is equal to zero.

Otherwise: Rx_n = bit_rate

where bit_rate is the bit rate BitRate[cpb_cnt_minus1] of data flow into the CPB for the byte stream format signalled in the NAL hrd_parameters() carried in VUI parameters in the AVC video stream. If NAL hrd_parameters() are not present in the AVC video stream, then the bit_rate shall be the bit rate 1200 × MaxBR[level] defined in Annex A of ITU-T Rec. H.264 | ISO/IEC 14496-10 for the level of the AVC video stream.

Transfer between MB_n and EB_n

If the AVC_timing_and_HRD_descriptor is present with the hrd_management_valid_flag set to '1', then the transfer of data from MB_n to EB_n shall follow the HRD defined scheme for data arrival in the CPB as defined in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10.

Otherwise, the leak method shall be used to transfer data from MB_n to EB_n as follows:

Rate Rbx_n:

$$Rbx_n = 1200 \times MaxBR[level]$$

where MaxBR[level] is defined for the byte stream format in Table A.1 (Level Limits) in ITU-T Rec. H.264 | ISO/IEC 14496-10 for each level.

If there is PES packet payload data in MB_n, and buffer EB_n is not full, the PES packet payload is transferred from MB_n to EB_n at a rate equal to Rbx_n. If EB_n is full, data are not removed from MB_n. When a byte of data is transferred from MB_n to EB_n, all PES packet header bytes that are in MB_n and precede that byte, are instantaneously removed and discarded. When there is no PES packet

payload data present in MB_n, no data is removed from MB_n. All data that enters MB_n leaves it. All PES packet payload data bytes enter EB_n instantaneously upon leaving MB_n.

Removal of AVC access units from EB_n

Each AVC access unit $A_n(j)$ that is present in EB_n is removed instantaneously at time $td_n(j)$. The decoding time $td_n(j)$ is specified by the DTS or from the CPB removal time, as derived from information in the AVC video stream.

STD delay

The total delay of any ITU-T Rec. H.264 | ISO/IEC 14496-10 data other than AVC still picture data through the System Target Decoders buffers TB_n, MB_n, and EB_n shall be constrained by $td_n(j) - t(i) \leq 10$ seconds for all j, and all bytes i in AVC access unit $A_n(j)$.

The delay of any AVC still picture data through the System Target Decoders buffers TB_n, MB_n, and EB_n shall be constrained by $td_n(j) - t(i) \leq 60$ seconds for all j, and all bytes i in AVC access unit $A_n(j)$.

Buffer management conditions

Transport streams shall be constructed so that the following conditions for buffer management are satisfied:

- TB_n shall not overflow and shall be empty at least once every second.
- MB_n, EB_n, and DPB_n shall not overflow.
- EB_n shall not underflow, except when VUI parameters are present for the AVC video sequence with the low_delay_hrd_flag set to '1'. Underflow of EB_n occurs for AVC access unit $A_n(j)$ when one or more bytes of $A_n(j)$ are not present in EB_n at the decoding time $td_n(j)$.

NOTE 2 – An AVC video stream may carry information to determine compliance of the AVC video stream to the HRD, as specified in Annex C of ITU-T Rec. H.264 | ISO/IEC 14496-10. The presence of this information can be signalled in a transport stream using the AVC timing and HRD descriptor with the hrd_management_valid_flag set to '1'. Irrespective of the presence of this information, compliance of an AVC video stream to the T-STD ensures that HRD buffer management requirements for CPB_n are met when each byte in the AVC video stream is delivered to and removed from CPB_n in the HRD at exactly the same instant in time at which the byte is delivered to and removed from EB_n in the T-STD.

2.14.3.2 P-STD extensions

The P-STD model for the decoding of an ITU-T Rec. H.264 | ISO/IEC 14496-10 elementary stream includes a multiplex buffer B_n and a decoder D_n followed by a buffer DPB_n (see Figure 2-12). For each AVC video stream n, the size BS_n of buffer B_n in the P-STD is defined by the P-STD_buffer_size field in the PES packet header.

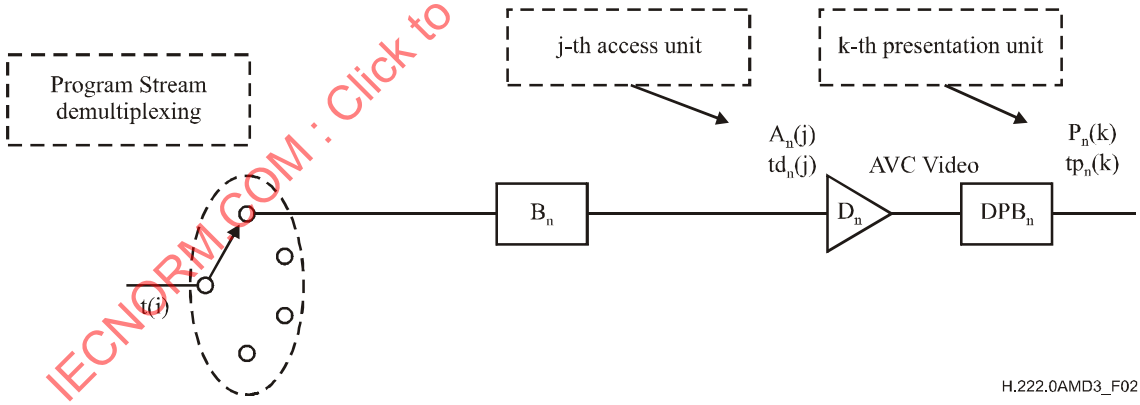

Figure 2-12 – P-STD model extensions for ITU-T Rec. H.264 | ISO/IEC 14496-10 video

DPB_n buffer management

Buffer DPB_n shall be managed in exactly the same way as in the T-STD; see 2.14.3.1.

B_n buffer management

The AVC access unit data enters buffer B_n as specified in 2.5.2.2. At time $td_n(j)$, AVC access unit $A_n(j)$ is decoded and instantaneously removed from B_n. The decoding time $td_n(j)$ is specified by the DTS or by the CPB removal time, derived from information in the AVC video stream. Upon decoding, the AVC access unit instantaneously enters DPB_n or is output without entry into DPB_n, according to the rules specified in ITU-T Rec. H.264 | ISO/IEC 14496-10.

STD delay

The total delay of any ITU-T Rec. H.264 | ISO/IEC 14496-10 data other than AVC still picture data through the System Target Decoders buffer B_n shall be constrained by $td_n(j) - t(i) \leq 10$ seconds for all j, and all bytes i in AVC access unit $A_n(j)$.

The delay of any AVC still picture data through the System Target Decoders buffer B_n shall be constrained by $td_n(j) - t(i) \leq 60$ seconds for all j, and all bytes i in AVC access unit $A_n(j)$.

Buffer management conditions

Program streams shall be constructed so that the following conditions for buffer management are satisfied:

- B_n shall not overflow.
- B_n shall not underflow, except when VUI parameters are present for the AVC video sequence with the low_delay_hrd_flag set to '1' or when trick_mode status is true. Underflow of B_n occurs for AVC access unit $A_n(j)$ when one or more bytes of $A_n(j)$ are not present in B_n at the decoding time $td_n(j)$.

Annex A

CRC decoder model

(This annex forms an integral part of this Recommendation | International Standard)

A.0 CRC decoder model

The 32-bit CRC decoder model is specified in Figure A.1.

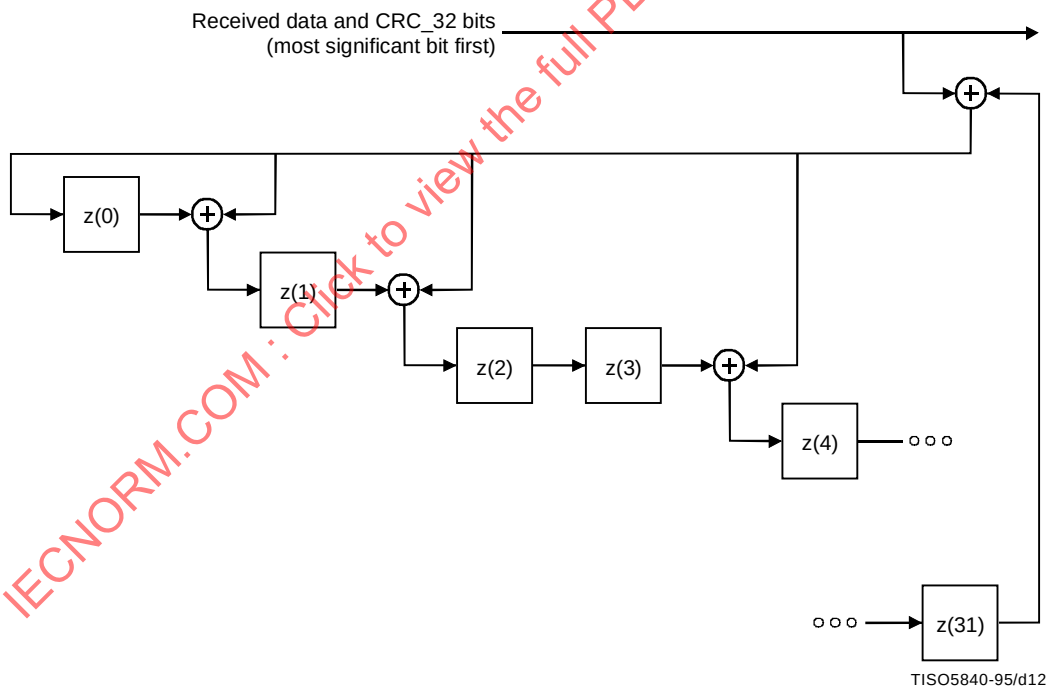

Figure A.1 – 32-bit CRC decoder model

The 32-bit CRC Decoder operates at bit level and consists of 14 adders '+' and 32 delay elements z(i). The input of the CRC decoder is added to the output of z(31), and the result is provided to the input z(0) and to one of the inputs of each remaining adder. The other input of each remaining adder is the output of z(i), while the output of each remaining adder is connected to the input of z(i + 1), with i = 0, 1, 3, 4, 6, 7 , 9, 10, 11, 15, 21, 22, and 25. Refer to Figure A.1 above.

This is the CRC calculated with the polynomial:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \qquad \text{(A-1)}$$

Bytes are received at the input of the CRC decoder. Each byte is shifted into the CRC decoder one bit at a time, with the left most bit (msb) first. For example, if the input is byte 0x01 the seven '0's enter the CRC decoder first, followed by the one '1'. Before the CRC processing of the data of a section the output of each delay element z(i) is set to its initial value '1'. After this initialization, each byte of the section is provided to the input of the CRC decoder, including the four CRC_32 bytes. After shifting the last bit of the last CRC_32 byte into the decoder, i.e., into z(0) after the addition with the output of z(31), the output of all delay elements z(i) is read. In the case where there are no errors, each of the outputs of z(i) shall be zero. At the CRC encoder the CRC_32 field is encoded with a value such that this is ensured.

Annex B

Digital Storage Medium Command and Control (DSM-CC)

(This annex does not form an integral part of this Recommendation | International Standard)

B.0 Introduction

The DSM-CC protocol is a specific application protocol intended to provide the basic control functions and operations specific to managing an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream on digital storage media. This DSM-CC is a low-level protocol above network/OS layers and below application layers.

The DSM-CC shall be transparent in the following sense:

- It is independent of the DSM used;
- it is independent of whether the DSM is located at a local or remote site;
- it is independent of the network protocol with which the DSM-CC is interfaced;
- it is independent of the various operating systems on which the DSM is operated.

B.0.1 Purpose

Many applications of ITU-T Rec. H.222.0 | ISO/IEC 13818-1 DSM Control Commands require access to an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream stored on a variety of digital storage media at a local or remote site. Different DSM have their own specific control commands and thus, a user would need to know different sets of specific DSM control commands in order to access ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams from different DSM. This brings many difficulties to the interface design of an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 or ISO/IEC 11172-1 application system. To overcome this difficulty, a set of common DSM control commands, which is independent of the specific DSM used, is suggested in this annex. This annex is informative only. ISO/IEC 13818-6 defines DSM-CC extension with a broader scope.

B.0.2 Future applications

Beyond the immediate applications supported by the current DSM control commands, future applications based on extensions of DSM command control could include the following:

Video on demand

Video programs are provided as requested by a customer through various communication channels. The customer could select a video program from a list of programs available from a video server. Such applications could be used by hotels, cable TV, educational institutions, hospitals, etc.

Interactive video services

In these applications, the user provides frequent feedback controlling the manipulation of stored video and audio. These services can include video-based games, user-controlled video tours, electronic shopping, etc.

Video networks

Various applications may wish to exchange stored audio and video data through some type of computer network. Users could route AV information through the video network to their terminals. Electronic publishing and multimedia applications are examples of this kind of application.

B.0.3 Benefits

Specifying the DSM control commands independent of the DSM, end-users can perform ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoding without having to fully understand the detailed operation of the specific DSM used.

The DSM control commands are codes to give end users the assurance that the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams can be played and stored with the same semantics, independent of the DSM and user interface. They are fundamental commands for the control of DSM operation.

B.0.4 Basic functions

B.0.4.1 Stream selection

The DSM-CC provides the means to select an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream upon which to perform the succeeding operations. Such operations include creation of a new bitstream. Parameters of this function include:

- index of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream (the mapping between this index and a name meaningful to an application is outside the scope of the current DSM-CC);
- mode (retrieval/storage).

B.0.4.2 Retrieval

The DSM-CC provides the means to:

- play an identified ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream;
- play from a given presentation time;
- set the playback speed (normal or fast);
- set the playback duration (until a specified presentation time, the end of the bitstream in forward play or the beginning in reverse play or the issuance of a stop command);
- set the direction (forward or reverse);
- pause;
- resume;
- change the access point in the bitstream;
- stop.

B.0.4.3 Storage

The DSM-CC provides the means to:

- cause storage of a valid bitstream for a specified duration;
- cause storage to stop.

DSM-CC provides a useful but limited subset of functionality that may be required in DSM based ITU-T Rec. H.222.0 | ISO/IEC 13818-1 applications. It is fully expected that significant additional capabilities will be added through subsequent extensions.

B.1 General elements

B.1.1 Scope

The scope of this work consists of the development of a Recommendation | International Standard to specify a useful set of commands for control of digital storage media on which an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is stored. The commands can perform remote control of a digital storage media in a general way independently of the specific DSM and apply to any ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream stored on a DSM.

B.1.2 Overview of the DSM-CC application

The current DSM-CC syntax and semantics cover the single user to DSM application. The user's system is capable of retrieving an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream and is also (optionally) capable of generating an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream. The control channel over which the DSM commands and acknowledgements are sent is shown in Figure B.1 as an out-of-band channel. This can also be accomplished by inserting the DSM-CC commands and acknowledgements into the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams if an out-of-band channel is not available.

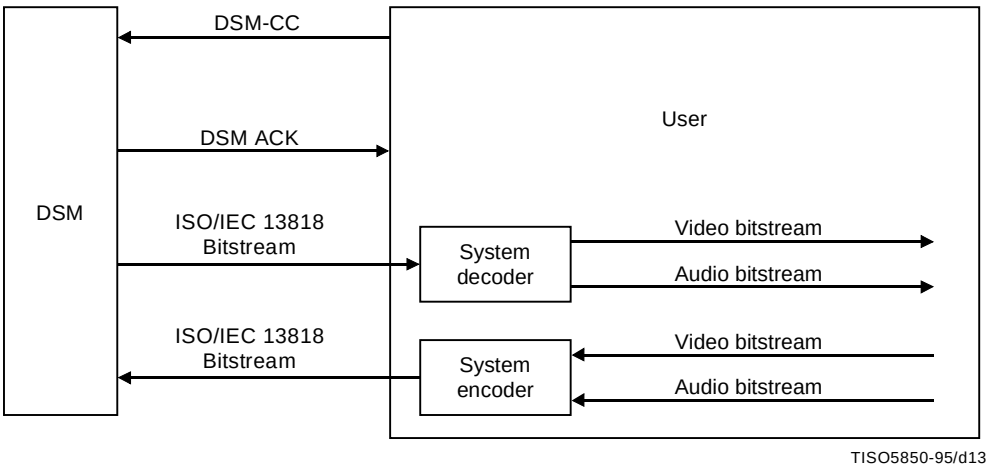

Figure B.1 – Configuration of DSM-CC application

B.1.3 The transmission of DSM-CC commands and acknowledgements

The DSM-CC is encoded into a DSM-CC bitstream according to the syntax and semantics defined in B.2.2 through B.2.9. The DSM-CC bitstream can be transmitted both as a stand-alone bitstream and in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Systems bitstream.

When the DSM-CC bitstream is transmitted in stand-alone mode, its relationship to the Systems bitstream and the decoding process is illustrated in Figure B.2. In this case, the DSM-CC bitstream is not embedded in the Systems bitstream. This transmission mode can be used in the applications when the DSM is connected directly with the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 decoder. It can also be used in the applications where the DSM-CC bitstream could be controlled and transmitted by other types of network multiplexors.

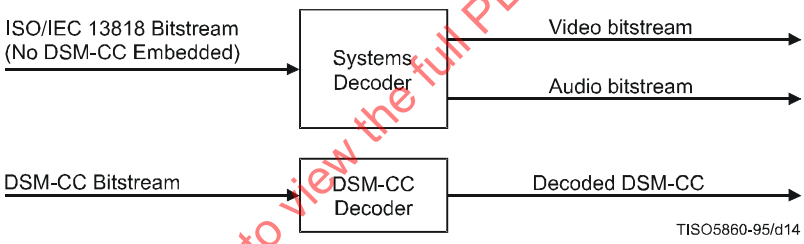

Figure B.2 – BSM-CC bitstream decoded as a stand-alone bitstream

For some applications, it is desirable to transmit the DSM-CC in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems bitstream so that some features of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems bitstream could be applied to the DSM-CC bitstream as well. In this case, the DSM-CC bitstream is embedded in the systems bitstream by the systems multiplexor.

The DSM-CC bitstream is encoded by the systems encoder in the following process. First, the DSM-CC bitstream is packetized into a packetized elementary stream (PES) according to the syntax described in 2.4.3.6. The PES packet is then multiplexed into either a Program Stream (PS) or a Transport Stream (TS) according to the requirement of the transmission media. The decoding procedures are the inverse of the encoding procedures and are illustrated in the block diagram of the Systems decoder depicted in Figure B.3.

In Figure B.3, the output of the Systems decoder is a video bitstream, audio bitstream and/or DSM-CC bitstream. The DSM-CC bitstream is identified by the stream_id, value '1111 0010' as defined by the stream_id Table 2-22. Once the DSM-CC bitstream is identified, it follows the rules as specified by T-STD or P-STD.

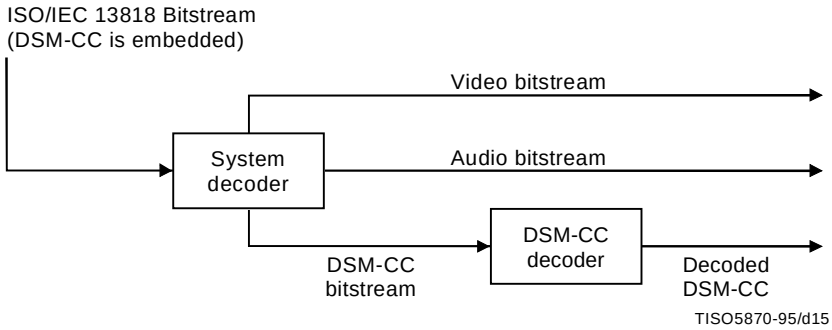

Figure B.3 – DSM-CC bitstream decoded as part of the system bitstream

B.2 Technical elements

B.2.1 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply:

B.2.1.1 DSM-CC: Digital Storage Media Command and Control Commands that are specified by ITU-T Rec. H.222.0 | ISO/IEC 13818-1 for the control of digital storage media at a local or remote site containing an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream.

B.2.1.2 DSM ACK: The acknowledgement from the DSM-CC command receiver to the command initiator.

B.2.1.3 MPEG bitstream: An ISO/IEC 11172-1 Systems stream, ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Program Stream or ITU-T Rec. H.222.0 | ISO/IEC 13818-1 Transport stream.

B.2.1.4 DSM-CC server: A system, either local or remote, used to store and/or retrieve an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream.

B.2.1.5 point of random access: A point in an ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream with the property that for at least one elementary stream within the bitstream, the next access unit, 'N', completely contained in the bitstream can be decoded without reference to previous access units, and for every elementary stream in the bitstream all access units with the same or later presentation times are completely contained subsequently in the bitstream and can be completely decoded by a system target decoder without access to information prior to the point of random access. The bitstream as stored on the DSM may have certain points of random access; the output of the DSM may include additional points of random access manufactured by the DSM's own manipulation of the stored material (e.g., storing quantization matrices so that a sequence header can be generated whenever necessary). A point of random access has an associated PTS, namely the actual or implied PTS of access unit 'N'.

B.2.1.6 current operational PTS value: The actual or implied PTS associated with the last point of random access preceding the last access unit provided from the DSM from the currently selected ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream. If no access unit has been provided from this ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream, the DSM is incapable of providing random access into the current bitstream, then the current operational PTS value is the first point of random access in the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream.

B.2.1.7 DSM-CC bitstream: A sequence of bits satisfying the syntax of B.2.2.

B.2.2 Specification of DSM-CC syntax

- Every DSM control command shall commence with a start_code, as specified in Table B.1.
- Every DSM control command shall have a packet_length to specify the number of byte in a DSM-CC packet.
- When the DSM-CC bitstream is transmitted as a PES packet as defined in 2.4.3.6, the fields up to the packet_length field are identical to those specified in 2.4.3.6. In other words, if the DSM-CC packet is encapsulated in a PES packet, the PES packet start code is the only start code at the beginning of the packet.
- The actual control command or acknowledgement shall follow the last byte of the packet_length field.
- An acknowledgement stream shall be provided by the DSM control bitstream receiver after the requested operation is started or is completed, depending on the command received.
- At all times the DSM is responsible for providing a normative ITU-T Rec. H.222.0 | ISO/IEC 13818-1 stream. This may include manipulating the trick mode bits defined in 2.4.3.6.

Table B.1 – DSM-CC syntax

Syntax	No. of bits	Mnemonic
DSM_CC() {		
packet_start_code_prefix	**24**	**bslbf**
stream_id	**8**	**uimsbf**
packet_length	**16**	**uimsbf**
command_id	**8**	**uimsbf**
If (command_id = = '01') {		
control()		
} else if (command_id = = '02') {		
ack()		
}		
}		

B.2.3 Semantics of fields in specification of DSM-CC syntax

packet_start_code_prefix – This is a 24-bit code. Together with the stream_id that follows it constitutes a DSM-CC packet start code that identifies the beginning of a DSM-CC packet bitstream. The packet_start_code_prefix is the bit string '0000 0000 0000 0000 0000 0001' (0x000001).

stream_id – This 8-bit field specifies the bitstream type and shall have a value '1111 0010' for the DSM-CC bitstream. Refer to Table 2-23.

packet_length – This 16-bit field specifies the number of bytes in the DSM-CC packet immediately following the last byte of this field.

command_id – This 8-bit unsigned integer identifies the bitstream is a control command or an acknowledgement stream. The values are defined in Table B.2.

Table B.2 – Command_id assigned values

Value	Command_id
0x00	Forbidden
0x01	Control
0x02	Ack
0x03-0xFF	Reserved

B.2.4 Control layer

Constraints on setting flags in DSM-CC control

- At most one of the flags for select, playback and storage shall be set to '1' for each DSM control command. If none of these bits are set, then this command shall be ignored.
- At most one of pause_mode, resume_mode, stop_mode, play_flag, and jump_flag shall be set for each retrieval command. If none of these bits are set, then this command shall be ignored.
- At most one of record_flag and stop_mode shall be selected for each storage command. If none of these bits are set, then this command shall be ignored.

See Table B.3.

Table B.3 – DSM-CC control

Syntax	No. of bits	Mnemonic
control() {		
select_flag	**1**	**bslbf**
retrieval_flag	**1**	**bslbf**
storage_flag	**1**	**bslbf**
reserved	**12**	**bslbf**
marker_bit	**1**	**bslbf**
If (select_flag = = '1') {		
bitstream_id [31..17]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
bitstream_id [16..2]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
bitstream_id [1..0]	**2**	**bslbf**
select_mode	**5**	**bslbf**
marker_bit	**1**	**bslbf**
}		
if (retrieve_flag = = '1') {		
jump_flag	**1**	**bslbf**
play_flag	**1**	**bslbf**
pause_mode	**1**	**bslbf**
resume_mode	**1**	**bslbf**
stop_mode	**1**	**bslbf**
reserved	**10**	**bslbf**
marker_bit	**1**	**bslbf**
if (jump_flag = ='1') {		
reserved	**7**	**bslbf**
direction_indicator	**1**	**bslbf**
time_code()		
}		
if (play_flag = ='1'){		
speed_mode	**1**	**bslbf**
direction_indicator	**1**	**bslbf**
reserved	**6**	**bslbf**
time_code()		
}		
}		
if (storage_flag = ='1') {		
reserved	**6**	**bslbf**
record_flag	**1**	**bslbf**
stop_mode	**1**	**bslbf**
if (record_flag = ='1') {		
time_code()		
}		
}		
}		

B.2.5 Semantics of fields in control layer

marker_bit – This is a 1-bit marker that is always set to '1' to avoid start code emulation.

reserved_bits – This 12-bit field is reserved for future use by this Recommendation | International Standard for DSM control commands. Until otherwise specified by ITU-T | ISO/IEC it shall have the value '0000 0000 0000'.

select_flag – This 1-bit flag when set to '1' specifies a bitstream selection operation. When it is set to '0' no bitstream selection operation shall occur.

retrieval_flag – This 1-bit flag when set to '1' specifies that a specific retrieval (playback) action will occur. The operation starts from the current operational PTS value.

storage_flag – This 1-bit flag when set to '1' specifies that a storage operation is to be executed.

bitstream_ID – This 32-bit field is coded in three parts. The parts are combined to form an unsigned integer specifying which ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is to be selected. It is the DSM server's responsibility to map the names of the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstreams stored on its DSM uniquely to a series of numbers which could be represented by the bitstream_ID.

select_mode – This 5-bit unsigned integer specifies which mode of bitstream operation is requested. Table B.4 specifies the defined modes.

Table B.4 – Select mode assigned values

Code	Mode
0x00	Forbidden
0x01	Storage
0x02	Retrieval
0x03-0x1F	Reserved

jump_flag – This 1-bit flag when set to '1' specifies a jump in the playback pointer to a new access unit. The new PTS is specified by a relative time_code with respect to the current operational PTS value. This function is only valid when the current ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is in the "stop" mode.

play_flag – This 1-bit flag when set to '1' specifies to play a bitstream for a certain time period. The speed, direction, and play duration are additional parameters in the bitstream. The play starts from the current operational PTS value.

pause_mode – This is a one-bit code specifying to pause the playback action and keep the playback pointer at the current operational PTS value.

resume_mode – This is a one-bit code specifying to continue the playback action from the current operational PTS value. Resume only has meaning if the current bitstream is in the "pause" state, and the bitstream will be set to the forward play state at normal speed.

stop_mode – This is a one-bit code specifying to stop a bitstream transmission.

direction_indicator – This is a one-bit code to indicate the playback direction. If this bit is set to '1', it stands for a forward play. Otherwise it stands for a backward play.

speed_mode – This is a 1-bit code to specify the speed scale. If this bit is set to '1', it specifies that the speed is normal play. If this bit is set to '0', it specifies that the speed is fast play (i.e., fast forward or fast reverse).

record_flag – This is one-bit flag to specify the request of recording the bitstream from an end user to a DSM for a specified duration or until the reception of a stop command, whichever comes first.

B.2.6 Acknowledgement layer

Constraints on setting flags in DSM-CC control

Only one of the acks bits specified below can be set to '1' for each DSM ack bitstream (see Table B.5).

Table B.5 – DSM-CC Acknowledgement

Syntax	No. of bits	Mnemonic
ack() {		
select_ack	**1**	**bslbf**
retrieval_ack	**1**	**bslbf**
storage_ack	**1**	**bslbf**
error_ack	**1**	**bslbf**
reserved	**10**	**bslbf**
marker_bit	**1**	**bslbf**
cmd_status	**1**	**bslbf**
If (cmd_status = = '1' &&		
(retrieval_ack = ='1' || storage_ack = = '1')) {		
time_code()		
}		
}		

B.2.7 Semantics of fields in Acknowledgement layer

select_ack – This 1-bit field when it is set to '1' indicates that the ack() command is to acknowledge a select command.

retrieval_ack – This 1-bit field when set to '1' indicates that the ack() command is to acknowledge a retrieval command.

storage_ack – This 1-bit field when set to '1' indicates that the ack() command is to acknowledge a storage command.

error_ack – This 1-bit field when set to '1' indicates a DSM error. The defined errors are EOF (end of file on forward play or start of file on reverse play) on a stream being retrieved and Disk Full on a stream being stored. If this bit is set to '1', cmd_status is undefined. The current bitstream is still selected.

cmd_status – This 1-bit flag set to '1' indicates that the command is accepted. When set to '0' it indicates the command is rejected. The semantics vary according to the command received as follows:

- If select_ack is set and cmd_status is set to '1', it specifies that the ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream is selected and the server is ready to provide the selected mode of operation. The current operational PTS value is set to the first point of random access of the newly selected ITU-T Rec. H.222.0 | ISO/IEC 13818-1 bitstream. If cmd_status is set to '0', the operation has failed and no bitstream is selected.
- If retrieval_ack is set and cmd_status is set to '1', it specifies that the retrieval operation is initiated for all retrieval commands. The position of the current operational PTS pointer is reported by the succeeding time_code.
- For the play_flag command with infinite_time_flag != '1', a second acknowledgement will be sent. This will acknowledge that the play operation has ended by reaching the duration defined by the play_flag command.
- If the cmd_status is set to '0' in a retrieval acknowledgement, the operation has failed. Possible reasons for this failure include an invalid bitstream_ID, jumping beyond the end of a file, or a function not supported such as reverse play in standard speed.
- If storage_ack is set, it specifies that the storage operation is being started for the record_flag command or is completed by the stop_mode command. The PTS of the last complete access unit stored is reported by the succeeding time_code.
- If the recording operation is ended by reaching the duration defined by the storage_flag command, another acknowledgement shall be sent and the current operational PTS value after the recording shall be reported.
- If the cmd_status is set to '0' in a storage acknowledgement, the operation has failed. Possible reasons for this failure include an invalid bitstream_ID, or the inability of the DSM to store data.

B.2.8 Time code

Constraints on time code

- A forward operation of specified duration given by a time_code terminates after the actual or implied PTS of an access unit is observed such that PTS minus the current operational PTS value at the start of the operation modulo 2^{33} exceeds the duration.
- A backward operation of specified duration given by a time_code terminates after the actual or implied PTS of an access unit is observed such that current operational PTS value at the start of the operation minus that PTS modulo 2^{33} exceeds the duration.
- For all the commands in the control() layer, the time_code is specified as a relative duration with respect to the current operational PTS value.
- For all the commands in the ack() layer, the time_code is specified by the current operational PTS value.

See Table B.6.

Table B.6 – Time code

Syntax	No. of bits	Mnemonic
time_code() {		
reserved	**7**	**bslbf**
infinite_time_flag	**1**	**bslbf**
if (infinite_time_flag = = '0') {		
reserved	**4**	**bslbf**
PTS [32..30]	**3**	**bslbf**
marker	**1**	**bslbf**
PTS [29..15]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
PTS [14..0]	**15**	**bslbf**
marker_bit	**1**	**bslbf**
}		
}		

B.2.9 Semantics of fields in time code

infinite_time_flag – This 1-bit flag when set to '1' indicates an infinite time period. This flag is set to '1' in applications where a time period for a specific operation could not be defined in advance.

PTS [32..0] – The presentation timestamp of the access unit of the bitstream. Depending upon the function, this can be an absolute value or a relative time delay in cycles of the 90-kHz system clock.

Annex C

Program Specific Information

(This annex does not form an integral part of this Recommendation | International Standard)

C.0 Explanation of Program Specific Information in Transport Streams

Subclause 2.4.4 contains the normative syntax, semantics and text concerning Program Specific Information. In all cases, compliance with the constraints of 2.4.4 is required. This annex provides explanatory information on how to use the PSI functions, and considers examples of how it may be used in practice.

C.1 Introduction

This Recommendation | International Standard provides a method for describing the contents of Transport Stream packets for the purpose of the demultiplexing and presentation of programs. The coding specification accommodates this function through the Program Specific Information (PSI). This annex discusses the use of PSI.

The PSI may be thought of as belonging to six tables:

1) Program Association Table (PAT);
2) TS Program Map Table (PMT);
3) Network Information Table (NIT);
4) Conditional Access Table (CAT).
5) Transport Stream Description Table; and
6) IPMP Control Information Table.

The contents of the PAT, PMT, CAT and TSDT are specified in this Recommendation | International Standard. ICIT is defined in ISO/IEC 13818-11 (MPEG-2 IPMP).

The NIT is a private table, and the PID value of the Transport Stream packets which carry it is specified in the PAT. Both the NIT and ICIT must follow the structure defined in this Recommendation | International Standard.

C.2 Functional mechanism

The tables listed above are conceptual in that they need never be regenerated in a specified form within a decoder. While these structures may be thought of as simple tables, they may be partitioned before they are sent in Transport Stream packets. The syntax supports this operation by allowing the tables to be partitioned into sections and by providing a normative mapping method into Transport Stream packet payloads. A method is also provided to carry private data in a similar format. This is advantageous as the same basic processing in the decoder can then be used for both the PSI data and the private data helping to keep cost down. For advice on the optimum placing of PSI in the Transport Stream, see Annex D.

Each section is uniquely identified by the combination of the following elements:

i) **table_id**

The 8-bit table_id identifies to which table the section belongs.

- Sections with table_id 0x00 belong to the Program Association Table.
- Sections with table_id 0x01 belong to the Conditional Access Table.
- Sections with table_id 0x02 belong to the TS Program Map Table.
- Sections with table_id 0x03 belong to the TS_description_section.
- Sections with table_id 0x04 belong to the ISO_IEC_14496_scene_description_section.
- Sections with table_id 0x05 belong to the ISO_IEC_14496_object_descriptor_section.
- Sections with table_id 0x06 belong to the metadata_section.
- Sections with table_id 0x07 belong to the IPMP_Control_Information_section.

Other values of the table_id can be allocated by the user for private purposes.

It is possible to set up filters looking at the table_id field to identify whether a new section belongs to a table of interest or not.

ii) **table_id_extension**

This 16-bit field exists in the long version of a section. In the Program Association Table it is used to identify the transport_stream_id of the stream – effectively a user-defined label which allows one Transport Stream to be distinguished from another within a network or across networks. In the Conditional Access Table this field currently has no meaning and is therefore marked as "reserved" meaning that it shall be coded as 0xFFFF, but that a meaning may be defined by ITU-T | ISO/IEC in a subsequent revision of this Recommendation | International Standard. In a TS Program Map section the field contains the program_number, and thereby identifies the program to which the data in the section refers. The table_id_extension can also be used as a filter point in certain cases.

iii) **section_number**

The section_number field allows the sections of a particular table to be reassembled in their original order by the decoder. There is no obligation within this Recommendation | International Standard that sections must be transmitted in numerical order, but this is recommended, unless it is desired to transmit some sections of the table more frequently than others, e.g., due to random access considerations.

iv) **version_number**

When the characteristics of the Transport Stream described in the PSI change (e.g., extra programs added, different composition of elementary streams for a given program), then new PSI data has to be sent with the updated information as the most recently transmitted version of the sections marked as "current" must always be valid. Decoders need to be able to identify whether the most recently received section is identical with the section they have already processed/stored (in which case the section can be discarded), or whether it is different, and may therefore signify a configuration change. This is achieved by sending a section with the same table_id, table_id_extension, and section_number as the previous section containing the relevant data, but with the next value version_number.

v) **current_next_indicator**

It is important to know at what point in the bitstream the PSI is valid. Each section can therefore be numbered as valid "now" (current), or as valid in the immediate future (next). This allows the transmission of a future configuration in advance of the change, giving the decoder the opportunity to prepare for the change. There is however no obligation to transmit the next version of a section in advance, but if it is transmitted, then it shall be the next correct version of that section.

C.3 The Mapping of Sections into Transport Stream Packets

Sections are mapped directly into Transport Stream packets, that is to say without a prior mapping into PES packets. Sections do not have to start at the beginning of Transport Stream packets, (although they may), because the start of the first section in the payload of a Transport Stream packet is pointed to by the pointer_field. The presence of the pointer_field is signalled by the payload_unit_start_indicator being set to a value of '1' in PSI packets. (In non-PSI packets, the indicator signals that a PES packet starts in the Transport Stream packet.) The pointer_field points to the start of the first section in the Transport Stream packet. There is never more than one pointer_field in a Transport Stream packet, as the start of any other section can be identified by counting the length of the first and any subsequent sections, since no gaps between sections within a Transport Stream packet are allowed by the syntax.

It is important to note that within Transport Stream packets of any single PID value, one section must be finished before the next one is allowed to be started, or else it is not possible to identify to which section header the data belongs. If a section finishes before the end of a Transport Stream packet, but it is not convenient to open another section, a stuffing mechanism is provided to fill up the space. Stuffing is performed by filling each remaining byte of the packet with the value 0xFF. Consequently the table_id value 0xFF is forbidden, or else this would be confused with stuffing. Once a 0xFF byte has occurred at the end of a section, then the rest of the Transport Stream packet must be stuffed with 0xFF bytes, allowing a decoder to discard the rest of the Transport Stream packet. Stuffing can also be performed using the normal adaptation_field mechanism.

C.4 Repetition rates and random access

In systems where random access is a consideration, it is recommended to re-transmit PSI sections several times, even when changes do not occur in the configuration, as in the general case, a decoder needs the PSI data to identify the contents of the Transport Stream, to be able to start decoding. This Recommendation | International Standard does not place any requirements on the repetition or occurrence rate of PSI sections. Clearly though, repeating sections frequently helps random access applications, whilst causing an increase in the amount of bitrate used by PSI data. If program mappings are static or quasi-static, they may be stored in the decoder to allow faster access to the data than having to wait for it to be re-transmitted. The trade-off between the amount of storage required and the desired impact on channel acquisition time may be made by the decoder manufacturer.

C.5 What is a program?

The concept of a program has a precise definition within this Recommendation | International Standard [refer to 2.1.60 program (system)]. For a Transport Stream the time base is defined by the PCR. This effectively creates a virtual channel within the Transport Stream.

Note that this is not the same definition as is commonly used in broadcasting, where a "program" is a collection of elementary streams not only with a common timebase, but also with a common start and end time. A series of "broadcaster programs" (referred to in this annex as events) can be transmitted sequentially in a Transport Stream using the same program_number to create a "broadcasting conventional" TV-channel (sometimes called a service).

Event descriptions could be transmitted in private_sections().

A program is denoted by a program_number which has significance only within a Transport Stream. The program_number is a 16-bit unsigned integer and thus permits 65535 unique programs to exist within a Transport Stream (program_number 0 is reserved for identification of the NIT). Where several Transport Streams are available to

the decoder (e.g., in a cable network), in order to successfully demultiplex a program, the decoder must be notified of both the transport_stream_id (to find the right multiplex) and the program_number of the service (to find the right program within the multiplex).

The Transport Stream mapping may be accomplished via the optional Network Information Table. Note that the Network Information Table may be stored in decoder non-volatile memory to reduce channel acquisition time. In this case, it needs to be transmitted only often enough to support timely decoder initialization set-up operations. The contents of the NIT are private, but shall take at least the minimum section structure.

C.6 Allocation of program_number

It may not be convenient in all cases to group together all the program element which share a common clock reference as one program. It is conceivable to have a multi-service Transport Stream with only one set of PCRs, common to all. In general, though, a broadcaster may prefer to logically split up the Transport Stream into several programs, where the PCR_PID (location of the clock reference) is always the same. This method of splitting the program elements into pseudo-independent programs can have several uses. Two examples follow:

i) *multilingual transmissions into separate markets*

One video stream may be accompanied by several audio streams in different languages. It is advisable to include an example of the ISO_639_language_descriptor associated with each audio stream to enable the selection of the correct program and audio. It is reasonable to have several program definitions with different program_numbers, where all the programs reference the same video stream and PCR_PID, but have different audio PIDs. It is, however, also reasonable and possible to list the video stream and all the audio streams as one program, where this does not exceed the section size limit of 1024 bytes.

ii) *Very large program definitions*

There is a maximum limit on the length of a section of 1024 bytes (including section header and CRC_32). This means that no single program definition may exceed this length. For the great majority of cases, even with each program element having several descriptors, this size is adequate. However, one may envisage cases in very high bitrate systems, which could exceed this limit. It is then in general possible to identify methods of splitting the references of the streams, so that they do not all have to be listed together. Some program elements could be referenced under more than one program, and some under only one or the other, but not both.

C.7 Usage of PSI in a typical system

A communications system, especially in broadcast applications, may consist of many individual Transport Streams. Each one of the four PSI data structures may appear in each and every Transport Stream in a system. There must always be a complete version of the program association table listing all programs within the Transport Stream and a complete TS program map table, containing complete program definitions for all programs within the Transport Stream. If any streams are scrambled, then there must also be a conditional access table present listing the relevant Entitlement Management Messages (EMM) streams. The presence of a NIT is fully optional.

The PSI tables are mapped into Transport Stream packets via the section structure described above. Each section has a table_id field in its header, allowing sections from PSI tables and private data in private_sections to be mixed in Transport Stream packets of the same PID value or even in the same Transport Stream packet. Note, however, that within packets of the same PID, a complete section must be transmitted before the next section can be started. This is only possible for packets labelled as containing TS Program Map Table section or NIT packets however, since private sections may not be mapped into PAT or CAT packets.

It is required that all PAT sections be mapped into Transport Stream packets with PID = 0x0000 and all CA sections be mapped into packets with PID = 0x0001. PMT sections may be mapped into packets of user-selected PID value, listed as the PMT_PID for each program in the Program Association Table. Likewise, the PID for the NIT-bearing Transport Stream packets is user-selected, but must be pointed to by the entry "program_number = = 0x00" in the PAT, if the NIT exists.

The contents of any CA parameter streams are entirely private, but EMMs and ECMs must also be sent in Transport Stream packets to be compliant with this Recommendation | International Standard.

Private data tables may be sent using the private_section() syntax. Such tables could be used for example in a broadcasting environment to describe a service, an upcoming event, broadcast schedules and related information.

C.8 The relationships of PSI structures

Figure C.1 shows an example of the relationship between the four PSI structures and the Transport Stream. Other examples are possible, but the figure shows the primary connections.

In the following subclauses, each PSI table is described.

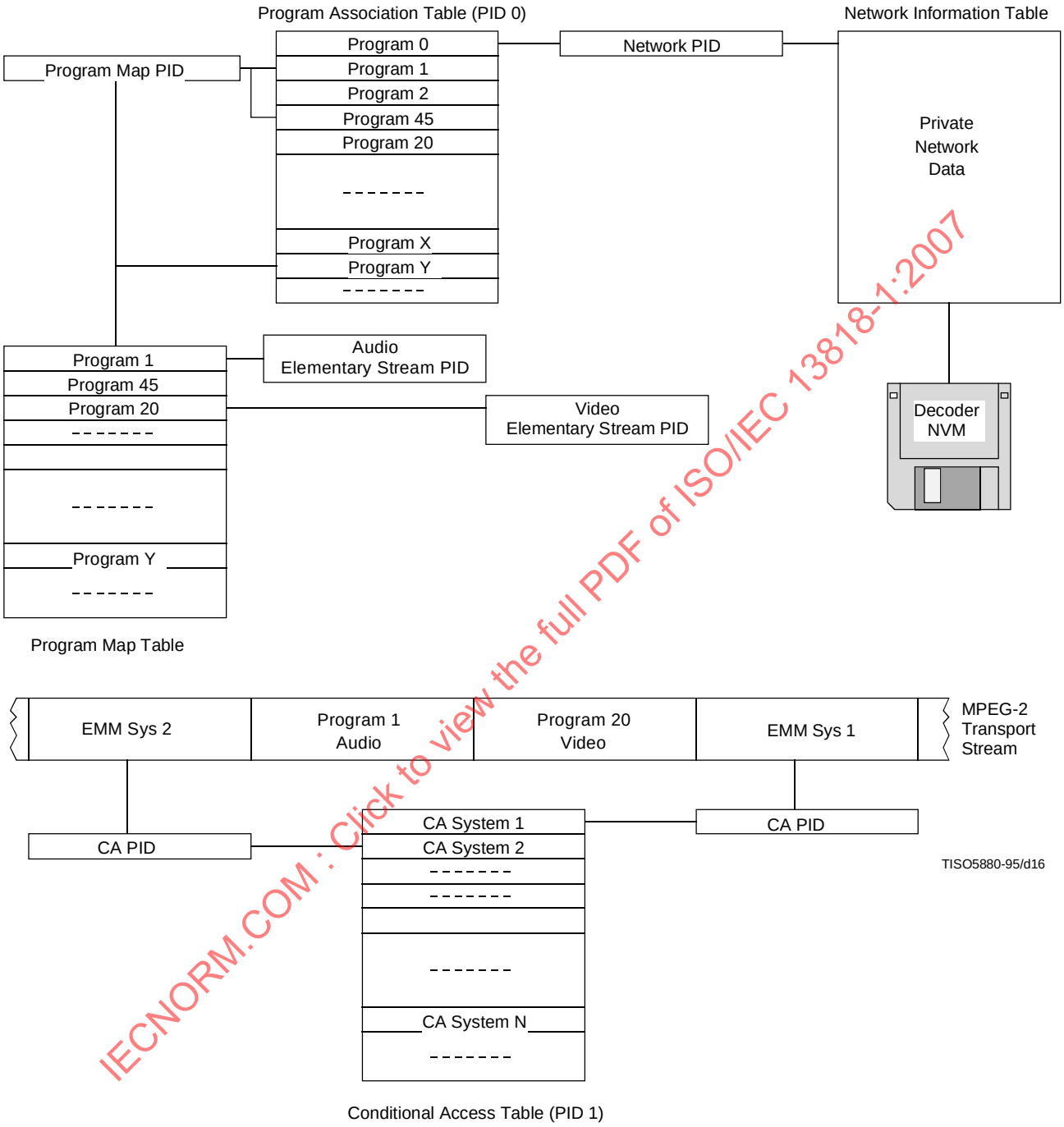

Figure C.1 – Program and network mapping relationships

C.8.1 Program Association Table

Every Transport Stream must contain a complete valid Program Association Table. The Program Association Table gives the correspondence between a program_number and the PID of the Transport Stream packets that carry the definition of that program (the PMT_PID). The PAT may be partitioned into up to 255 sections before it is mapped into Transport Stream packets. Each section carries a part of the overall PAT. This partitioning may be desirable to minimize data loss in error conditions. That is, packet loss or bit errors may be localized to smaller sections of the PAT, thus allowing other sections to still be received and correctly decoded. If all PAT information is put into one section, an

error causing a changed bit in the table_id, for example, would cause the loss of the entire PAT. However, this is still permitted as long as the section does not extend beyond the 1024-byte maximum length limit.

Program 0 (zero) is reserved and is used to specify the Network PID. This is a pointer to the Transport Stream packets which carry the Network Information Table.

The Program Association Table is always transmitted without encryption.

C.8.2 Program Map Table

The Program Map Table provides the mapping between a program number and the program elements that comprise it. This table is present in Transport Stream packets having one or more privately-selected PID values. These Transport Stream packets may contain other private structures as defined by the table_id field. It is possible to have TS PMT sections referring to different programs carried in Transport Stream packets having a common PID value.

This Recommendation | International Standard requires a minimum of program identification: program number, PCR PID, stream types and program elements PIDs. Additional information for either programs or elementary streams may be conveyed by use of the descriptor() construct. Refer to C.8.6.

Private data may also be sent in Transport Stream packets denoted as carrying TS program map table sections. This is accomplished by the use of the private_section(). In a private_section() the application decides whether version_number and current_next_indicator represent the values of these fields for a single section or whether they are applicable to many sections as parts of a larger private table.

NOTE 1 – Transport stream packets containing the Program Map Table are transmitted unencrypted.

NOTE 2 – It is possible to transmit information on events in private descriptors carried within the TS_program_map_section()s.

C.8.3 Conditional Access Table

The Conditional Access (CA) Table gives the association between one or more CA systems, their EMM streams and any special parameters associated with them.

NOTE – The (private) contents of the Transport Stream packets containing EMM and CA parameters if present will, in general, be encrypted (scrambled).

C.8.4 Network Information Table

The contents of the NIT are private and not specified by this Recommendation | International Standard. In general, it will contain mappings of user-selected services with transport_stream_ids, channel frequencies, satellite transponder numbers, modulation characteristics, etc.

C.8.5 Private_section()

Private_sections() can occur in two basic forms, the short version (where only the fields up to and including section_length are included) or the long version (where all the fields up to and including last_section_number are present, and after the private data bytes the CRC_32 field is present).

Private_section()s can occur in PIDs which are labelled as PMT_PIDs or in Transport Stream packets with other PID values which contain exclusively private_sections(), including the PID allocated to the NIT. If the Transport Stream packets of the PID carrying the private_section()s are identified as a PID carrying private_sections (stream_type assignment value 0x05), then only private_sections may occur in Transport Stream packets of that PID value. The sections may be either of the short or long type.

C.8.6 Descriptors

There are several normative descriptors defined in this Recommendation | International Standard. Many more private descriptors may also be defined. All descriptors have a common format: {tag, length, data}. Any privately defined descriptors must adhere to this format. The data portion of these private descriptors are privately defined.

One descriptor (the CA_descriptor()), is used to indicate the location (PID value of transport packets) of ECM data associated with program elements when it is found in a TS PMT section. When found in a CA section it refers to EMMs.

In order to extend the number of private_descriptors available, the following mechanism could be used: A private descriptor_tag could be privately defined to be constructed as a composite descriptor. This entails privately defining a further sub_descriptor as the first field of the private data bytes of the private descriptor. The described structure is as indicated in Tables C.1 and C.2.