ETSI EN 301 192 V1.4.1 (2004-11)

European Standard (Telecommunications series)

Digital Video Broadcasting (DVB); DVB specification for data broadcasting

Reference
REN/JTC-DVB-157

Keywords
broadcasting, data, digital, DVB, MPEG, video

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

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

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

Foreword

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

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

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

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

National transposition dates	
Date of adoption of this EN:	22 October 2004
Date of latest announcement of this EN (doa):	31 January 2005
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	31 July 2005
Date of withdrawal of any conflicting National Standard (dow):	31 July 2005

1 Scope

The present document is designed to be used in conjunction with EN 300 468 [2] and TR 101 211 [4]. The DVB System provides a means of delivering MPEG-2 Transport Streams (TS) via a variety of transmission media. These TSs have traditionally been oriented to containing MPEG-2 Video and Audio. Data broadcasting is seen as an important extension of the MPEG-2 based DVB transmission standards. Examples for data broadcasting are the download of software over satellite, cable or terrestrial links, the delivery of Internet services over broadcast channels (IP tunnelling), interactive TV etc. Four different application areas with different requirements for the data transport have been identified. For each application area a data broadcasting profile is specified in the present document. The following is a short description of the application areas and the profiles.

Data piping:

- The data broadcast specification profile for data pipes supports data broadcast services that require a simple, asynchronous, end-to-end delivery of data through DVB compliant broadcast networks. Data broadcast according to the data pipe specification is carried directly in the payloads of MPEG-2 TS packets (see ISO/IEC 13818-1 [1]).

Data streaming:

- The data broadcast specification profile for data streaming supports data broadcast services that require a streaming-oriented, end-to-end delivery of data in either an asynchronous, synchronous or synchronized way through DVB compliant broadcast networks. Data broadcast according to the data streaming specification is carried in Program Elementary Stream (PES) packets which are defined in MPEG-2 Systems (see ISO/IEC 13818-1 [1]).

- Asynchronous data streaming is defined as the streaming of only data without any timing requirements (e.g. RS-232 data).

- Synchronous data streaming is defined as the streaming of data with timing requirements in the sense that the data and clock can be regenerated at the receiver into a synchronous data stream (e.g. E1, T1). Synchronized data streaming is defined as the streaming of data with timing requirements in the sense that the data within the stream can be played back in synchronization with other kinds of data streams (e.g. audio, video).

Multiprotocol encapsulation:

- The data broadcast specification profile for multiprotocol encapsulation supports data broadcast services that require the transmission of datagrams of communication protocols via DVB compliant broadcast networks. The transmission of datagrams according to the multiprotocol encapsulation specification is done by encapsulating the datagrams in DSM-CC sections (see ISO/IEC 13818-6 [5]), which are compliant with the MPEG-2 private section format (see ISO/IEC 13818-1 [1]).

- The data broadcast specification support a standard mechanism for signalling IP/MAC services deployed within DVB networks and enables the implementation of DVB receivers that are completely self-tuning when accessing IP/MAC streams on one or more transport streams. The signalling mechanism is provided via the IP/MAC Notification Table (INT). The mechanism builds on EN 300 468 [2], TR 101 162 [3] and ISO/IEC 13818-6 [5] for signalling and the current specification for data carriage.

Data carousels:

- The data broadcast specification for data carousels supports data broadcast services that require the periodic transmission of data modules through DVB compliant broadcast networks. The modules are of known sizes and may be updated, added to, or removed from the data carousel in time. Modules can be clustered into a group of modules if required by the service. Likewise, groups can in turn be clustered into SuperGroups.

- Data broadcast according to the data carousel specification is transmitted in a DSM-CC data carousel which is defined in MPEG-2 DSM-CC (see ISO/IEC 13818-6 [5]). The present document defines additional structures and descriptors to be used in DVB compliant networks. The method is such that no explicit references are made to PIDs and timing parameters enabling preparation of the content off-line.

Object carousels:

- The object carousel specification has been added in order to support data broadcast services that require the periodic broadcasting of DSM-CC User-User (U-U) Objects through DVB compliant broadcast networks, specifically as defined by DVB Systems for Interactive Services (SIS) (see ETS 300 802 [10]). Data broadcast according to the DVB object carousel specification is transmitted according to the DSM-CC Object Carousel and DSM-CC Data Carousel specification which are defined in MPEG-2 DSM-CC (see ISO/IEC 13818-6 [5]).

Higher protocols based on asynchronous data streams:

- The data broadcast specification profile for higher protocols based on asynchronous data streams supports the transmission of protocols that require a stream-oriented delivery of asynchronous data through DVB compliant broadcast networks. The data frames of these protocols are carried in Program Elementary Stream (PES) packets which are defined in MPEG-2 Systems (see ISO/IEC 13818-1 [1]).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication and/or edition number or version number) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

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

[1] ISO/IEC 13818-1: "Information technology - Generic coding of moving pictures and associated audio information: Systems".

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

[3] ETSI TR 101 162: "Digital Video Broadcasting (DVB); Allocation of Service Information (SI) codes for DVB systems".

[4] ETSI TR 101 211: "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

[5] ISO/IEC 13818-6: "Information technology - Generic coding of moving pictures and associated audio information - Part 6: Extensions for DSM-CC".

[6] ETSI EN 300 472: "Digital Video Broadcasting (DVB); Specification for conveying ITU-R System B Teletext in DVB bitstreams".

[7] IETF RFC 1112 (1989): "Host extensions for IP multicasting".

[8] IETF RFC 2045 (1996): "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies".

[9] IETF RFC 2046 (1996): "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types".

[10] ETSI ETS 300 802: "Digital Video Broadcasting (DVB); Network-independent protocols for DVB interactive services".

[11] ISO/IEC TR 8802-1: "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 1: Overview of Local Area Network Standards".

[12] ISO/IEC 8802-2: "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 2: Logical link control".

[13] ETSI EN 300 743: "Digital Video Broadcasting (DVB); subtitling systems".

[14] ISO 8859-1: "Information technology - 8-bit single-byte coded graphic character sets - Part 1: Latin alphabet No. 1".

[15] ISO 639-2: "Code for the representation of names of languages - Part 2: Alpha-3 code".

[16] IETF RFC 1950 (1996): "ZLIB Compressed Data Format Specification version 3.3".

[17] "RTCM Recommended Standards for Differential GNSS (Global Navigation Satellite Systems) Service", Version 2.2, Radio Technical Commission For Maritime Services, January 1998.

[18] BPN 027-2: "EBU B/TPEG Transport Protocol Experts Group (TPEG) TPEG specifications - Part 2: Syntax, Semantics and Framing Structure", Version 1.1, June 1999.

[19] ETSI TS 102 006: "Digital Video Broadcasting (DVB); Specification for System Software Update in DVB Systems".

[20] IETF RFC 2464 (1998): "Transmission of IPv6 Packets over Ethernet Networks".

[21] IETF RFC 1661 (1994): "The Point-to-Point Protocol (PPP)".

[22] ETSI TS 101 197-1: "Digital Video Broadcasting (DVB); DVB SimulCrypt; Part 1: Head-end architecture and synchronization".

3 Abbreviations

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

AFI	Authority and Format Identifier
BAT	Bouquet Association Table
bslbf	bit string, left bit first
CA	Conditional Access
CAT	Conditional Access Table
CRC	Cyclic Redundancy Code
DAVIC	Digital Audio VIsual Council
DDB	DownloadDataBlock
dGNSS	differential GNSS
DII	DownloadInfoIndication
DSI	DownloadServerInitiate
DSM-CC	DSM-CC data carousel specification (Digital Storage Media- Command & Control)
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
EIT	Event Information Table
EMM	Entitlement Management Message
FEC	Forward Error Correction
gi	GroupInfoBytes
GNSS	Global Navigation Satellite Systems
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers (USA)
INT	IP/MAC Notification Table
IP	Internet Protocol
ISDN	International Services Digital Network
ISO	International Organization for Standardization
kbits	kilo bits
LLC	Logical Link Control
MAC	Media Access Control
Mbits	Mega bits

MHP	Multimedia Home Platform
mi	ModuleInfoBytes
mi	ModuleInfoBytes
MIME	Multipurpose Internet Mail Extensions
MPE	Multi-Protocol Encapsulation
MPEG	Moving Pictures Expert Group
ms	millisecond
MSB	Most Significant Bit
NIT	Network Information Table
NSAP	Network Service Access Point
OUI	Organizational Unique Identifier
PCR	Program Clock Reference
PES	Program Elementary Stream
PID	Packet IDentifier
PMT	Program Map Table
PPP	Point to Point Protocol
PSI	Program Specific Information
PSTN	Public Switched Telecommunication Network
PTS	Presentation Time Stamps
RF	Radio Frequency
RFC	Request For Comment
rpchof	remainder polynomial coefficients, highest order first
RS	Reed-Solomon
RTCM	Radio Technical Commission For Maritime Services
SDT	Service Description Table
SI	Service Information
SIS	Systems for Interactive Services
SNAP	SubNetwork Attachment Point
SSU	System Software Update
TPEG	Transport Protocol Experts Group
TS	Transport Stream
T-STD	Transport System Target Decoder
uimsbf	unsigned integer most significant bit first
U-U	User-User

4 Data piping

4.1 Data transport specification

The data broadcast service shall insert the data to be broadcast directly in the payload of MPEG-2 TS packets.

The data broadcast service may use the payload_unit_start_indicator field and the transport_priority field of the MPEG-2 Transport Stream packets in a service private way. The use of the adaptation_field shall be MPEG-2 compliant.

The delivery of the bits in time through a data pipe is service private and is not specified in the present document.

4.2 PSI and SI specifications

The data broadcast service shall indicate the use of a data pipe by including one or more data_broadcast_descriptors in SI (see EN 300 468 [2]). Each descriptor shall be associated with a particular data pipe via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map section for the stream that is used as a data pipe.

4.2.1 Data_broadcast_descriptor

The data_broadcast_descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0001 to indicate a DVB data pipe (see TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor (if present in the PSI program map section) for the stream that is used as a data pipe.

selector_length: This field shall be set to zero.

selector_byte: This field is not present.

4.2.2 Stream type

The specification of the stream_type in the program map section is not defined in the present document.

5 Asynchronous data streaming

5.1 Data transport specification

The data broadcast service shall insert the data to be broadcast in PES packets as defined by MPEG-2 Systems ISO/IEC 13818-1 [1]. The PES packets shall be of non-zero length. The mapping of the PES packets into MPEG-2 Transport Stream packets is defined in MPEG-2 Systems ISO/IEC 13818-1 [1].

The asynchronous data streaming specification uses the standard PES packet syntax and semantics with the following constraints:

stream_id: This field shall be set to the value of 0xBF (private_stream_2).

PES_packet_length: This is a 16-bit field which shall be set to a non-zero value.

5.2 PSI and SI specifications

The data broadcast service shall indicate the use of an asynchronous data stream by including one of more data broadcast descriptors in SI (see EN 300 468 [2]). Each descriptor shall be associated with a particular stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map section for the stream that is used as a data stream.

5.2.1 Data_broadcast_descriptor

The data broadcast descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0002 to indicate an asynchronous data stream (see TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor (if present in the PSI program map section) for the stream on which the data is broadcast.

selector_length: This field shall be set to zero.

selector_byte: This field is not present.

5.2.2 Stream type

The presence of an asynchronous data stream in a service shall be indicated in the program map of that service by setting the stream type of that stream to the value of 0x06 or a user private value.

6 Synchronous and synchronized data streaming

6.1 Data transport specification

The data broadcast service shall insert the data to be broadcast in PES packets as defined by MPEG-2 Systems. The PES packets shall be of non-zero length. The mapping of the PES packets into MPEG-2 Transport Stream packets is defined in MPEG-2 Systems ISO/IEC 13818-1 [1].

The synchronous and synchronized data streaming specifications use the standard PES packet syntax and semantics with the following constraints:

stream_id: This field shall be set to the value of 0xBD (private_stream_1) for synchronous and synchronized data streams.

PES_packet_length: This is a 16-bit field which shall be set to a non-zero value.

The data is inserted in PES packets using the PES_data_packet structure. The syntax and semantics of the PES_data_packet structure are defined in table 1.

Table 1: Syntax for PES_data_packet structure

Syntax	Number of bits	Mnemonic
PES_data_packet () {		
data_identifier	8	uimsbf
sub_stream_id	8	uimsbf
PTS_extension_flag	1	bslbf
output_data_rate_flag	1	bslbf
Reserved	2	bslbf
PES_data_packet_header_length	4	uimsbf
if (PTS_extension_flag=="1") {		
Reserved	7	bslbf
PTS_extension	9	bslbf
}		
if (output_data_rate_flag=="1") {		
Reserved	4	bslbf
output_data_rate	28	uimsbf
}		
for (i=0;i<N1;i++) {		
PES_data_private_data_byte	8	bslbf
}		
for (i=0;i<N2;i++) {		
PES_data_byte	8	bslbf
}		
}		

The semantics of the PES_data_packet are as follows:

data_identifier: This 8-bit field identifies the type of data carried in the PES data packet. It is coded as in table 2 (see also TR 101 162 [3] and EN 300 472 [6]).

Table 2: Coding for data_identifier field

data_identifier	value
0x00 to 0x0F	reserved for future use
0x10 to 0x1F	reserved for EBU data (see EN 300 472 [6])
0x20	DVB subtitling (see EN 300 743 [13])
0x21	DVB synchronous data stream
0x22	DVB synchronized data stream
0x23 to 0x7F	reserved for future use
0x80 to 0xFF	user defined

The data_identifier field shall be set to the same value for each PES packet conveying data in the same data stream.

sub_stream_id: This is an 8-bit field. Its use is user private.

PTS_extension_flag: This is a 1-bit field. It shall be set to "1" for synchronous data streams. For synchronized data streams a value of "1" indicates the presence of the PTS_extension field in the PES_data_packet. If the PTS_extension field is not present for synchronized data streams, this flag shall be set to "0".

output_data_rate_flag: This is a 1-bit field. It shall be set to "0" for synchronized data streams. For synchronous data streams a value of "1" indicates the presence of the output_rate field in the PES_data_packet. If the output_rate field is not present for synchronous data streams, this flag shall be set to "0".

PES_data_packet_header_length: This is a 4-bit field. It shall specify the length of the optional fields in the packet header including the PES_data_private_data_bytes.

PTS_extension: This is a 9-bit field. This field extends the PTS conveyed in the PES header of this PES packet. This field contains the 9-bit Program Clock Reference (PCR) extension as defined in MPEG-2 Systems (see ISO/IEC 13818-1 [1]) that extends the time resolution of data PTS's (synchronous or synchronized) from the MPEG-2 standard resolution of 11,1 µs (90 kHz) to 37 ns (27 MHz).

output_data_rate: This is a 28-bit field that shall indicate the bit rate of the regenerated signal for a synchronous data stream. The output data rate is encoded as a 28-bit positive integer.

PES_data_private_data_byte: The use of these bytes is service specific. DVB compliant receivers may skip over these bytes if present.

PES_data_byte: These bytes convey the data to be broadcast.

6.2 PSI and SI specifications

The data broadcast service shall indicate the use of a synchronous or synchronized data stream by including one of more data_broadcast_descriptors in SI (see EN 300 468 [2]). Each descriptor shall be associated with a particular stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map section for the stream that is used as a data stream.

6.2.1 Data_broadcast_descriptor

The data broadcast descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0003 to indicate a synchronous data stream and to 0x0004 for synchronized data streams (see TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor (if present in the PSI program map section) for the stream on which the data is broadcast.

selector_length: This field shall be set to zero.

selector_byte: This field is not present.

6.2.2 Stream type

The presence of a synchronous data stream or a synchronized data stream in a service shall be indicated in the program map section of that service by setting the stream type of that stream to the value of 0x06 or a user defined value.

7 Multiprotocol encapsulation

7.1 Data transport specification

Datagrams are encapsulated in datagram_sections which are compliant to the DSMCC_section format for private data (see ISO/IEC 13818-6 [5]). The mapping of the section into MPEG-2 Transport Stream packets is defined in MPEG-2 Systems ISO/IEC 13818-1 [1].

The syntax and semantics of the datagram_section are defined in table 3.

Table 3: Syntax of datagram_section

Syntax	Number of bits	Mnemonic
datagram_section() {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
MAC_address_6	8	uimsbf
MAC_address_5	8	uimsbf
reserved	2	bslbf
payload_scrambling_control	2	bslbf
address_scrambling_control	2	bslbf
LLC_SNAP_flag	1	bslbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
MAC_address_4	8	uimsbf
MAC_address_3	8	uimsbf
MAC_address_2	8	uimsbf
MAC_address_1	8	uimsbf
if (LLC_SNAP_flag == "1") {		
LLC_SNAP()		
} else {		
for (j=0;j<N1;j++) {		
IP_datagram_data_byte	8	bslbf
}		
}		
if (section_number == last_section_number) {		
for (j=0;j<N2;j++) {		
stuffing_byte	8	bslbf
}		
}		
if (section_syntax_indicator =="0") {		
checksum	32	uimsbf
} else {		
CRC_32	32	rpchof
}		
}		

The semantics of the datagram_section are as follows:

table_id: This is an 8-bit field which shall be set to 0x3E (DSM-CC sections with private data (ISO/IEC 13818-6 [5])).

section_syntax_indicator: This field shall be set as defined by ISO/IEC 13818-6 [5].

private_indicator: This field shall be set as defined by ISO/IEC 13818-6 [5].

reserved: This is a 2-bit field that shall be set to "11".

section_length: This field shall be set as defined by ISO/IEC 13818-6 [5].

MAC_address_[1..6]: This 48-bit field contains the MAC address of the destination. The MAC address is fragmented in 6 fields of 8-bits, labelled MAC_address_1 to MAC_address_6. The MAC_address_1 field contains the most significant byte of the MAC address, while MAC_address_6 contains the least significant byte. Figure 1 illustrates the mapping of the MAC address bytes in the section fields.

NOTE: The order of the bits in the bytes is not reversed and that the Most Significant Bit (MSB) of each byte is still transmitted first.

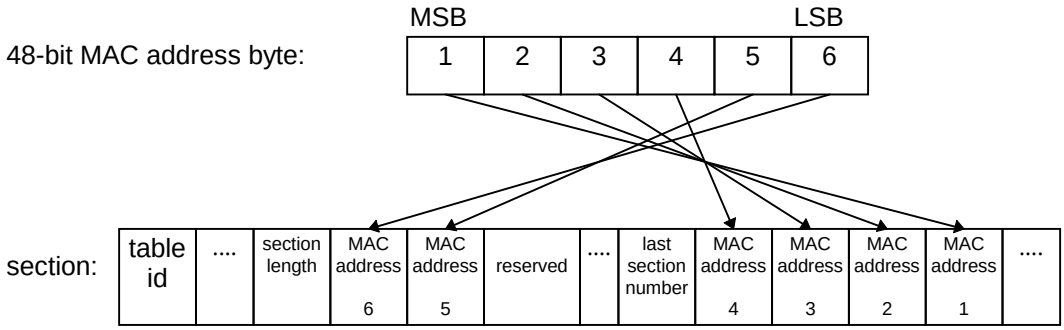

Figure 1: Mapping of MAC address bytes to section fields

The MAC_address fields contain either a clear or a scrambled MAC address as indicated by the address_scrambling_control field.

payload_scrambling_control: This 2-bit field defines the scrambling mode of the payload of the section. This includes the payload starting after the MAC_address_1 but excludes the checksum or CRC32 field (see table 4). The scrambling method applied is user private.

Table 4: Coding of the payload_scrambling_control field

value	payload scrambling control
00	unscrambled
01	defined by service
10	defined by service
11	defined by service

address_scrambling_control: This 2-bit field defines the scrambling mode of MAC address in this clause (see table 5). This field enables a dynamic change of MAC addresses. The scrambling method applied is user private.

Table 5: Coding of the address_scrambling_control field

value	address scrambling control
00	unscrambled
01	defined by service
10	defined by service
11	defined by service

LLC_SNAP_flag: This is a 1-bit flag. If this flag is set to "1" the payload carries an LLC/SNAP encapsulated datagram following the MAC_address_1 field. The LLC/SNAP structure shall indicate the type of the datagram conveyed. If this flag is set to "0", the section shall contain an IP datagram without LLC/SNAP encapsulation.

current_next_indicator: This is a 1-bit field. It shall be set to a value of "1".

section_number: This is an 8-bit field. If the datagram is carried in multiple sections, then this field indicates the position of the section within the fragmentation process. Otherwise it shall be set to zero.

last_section_number: This 8-bit field shall indicate the number of the last section that is used to carry the datagram, i.e. the number of the last section of the fragmentation process.

LLC_SNAP: This structure shall contain the datagram according to the ISO/IEC 8802-2 [12] Logical Link Control (LLC) and ISO/IEC TR 8802-1 [11] a SubNetwork Attachment Point (SNAP) specifications. If the payload of the section is scrambled (see payload_scrambling_mode), these bytes are scrambled.

IP_datagram_data_byte: These bytes contain the data of the datagram. If the payload of the section is scrambled (see payload_scrambling_mode), these bytes are scrambled.

stuffing_byte: This is an optional 8-bit field whose value is not specified. If the payload of the section is scrambled (see payload_scrambling_mode), these bytes are scrambled. They are to assist with block encryption and data processing in wide bus environments. The number of stuffing_bytes used should meet the data alignment requirements defined in the data_broadcast_descriptor.

checksum: This field shall be set as defined by ISO/IEC 13818-6 [5]. It is calculated over the entire datagram_section.

CRC_32: This field shall be set as defined by ISO/IEC 13818-6 [5]. It is calculated over the entire datagram_section.

7.2 MPE PSI and SI specifications

The data broadcast service shall indicate the transmission of datagrams by including one or more data broadcast descriptors in SI (see EN 300 468 [2] and TR 101 162 [3]). Each descriptor shall be associated with a stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map table for the stream that is used to transmit the datagrams.

7.2.1 Data_broadcast_descriptor

The data broadcast descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0005 to indicate the use of the multiprotocol encapsulation specification (see also TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor that may be present in the PSI program map section for the stream on which the data is broadcast.

selector_length: This field shall be set to 0x02.

selector_byte: The selector bytes shall convey the multiprotocol_encapsulation_info structure which is defined in table 6.

Table 6: Syntax for multiprotocol_encapsulation_info structure

Syntax	Number of bits	Mnemonic
multiprotocol_encapsulation_info () {		
MAC_address_range	3	uimsbf
MAC_IP_mapping_flag	1	bslbf
alignment_indicator	1	bslbf
reserved	3	bslbf
max_sections_per_datagram	8	uimsbf
}		

The semantics of the multiprotocol_encapsulation_info structure are as follows:

MAC_address_range: This 3-bit field shall indicate the number of MAC address bytes that the service uses to differentiate the receivers according to table 7.

Table 7: Coding of the MAC_address_range field

MAC_address_range	valid MAC_address bytes
0x00	reserved
0x01	6
0x02	6, 5
0x03	6, 5, 4
0x04	6, 5, 4, 3
0x05	6, 5, 4, 3, 2
0x06	6, 5, 4, 3, 2, 1
0x07	reserved

MAC_IP_mapping_flag: This is a 1-bit flag. The service shall set this flag to "1" if it uses the IP to MAC mapping as described in RFC 1112 [7] for IPv4 multicast addresses and RFC 2464 [20] for IPv6 multicast addresses. If this flag is set to "0", the mapping of IP addresses to MAC addresses is done outside the scope of the present document.

alignment_indicator: This is a 1-bit field that shall indicate the alignment that exists between the bytes of the datagram_section and the Transport Stream bytes according to table 8.

Table 8: Coding of the alignment_indicator field

value	alignment in bits
0	8 (default)
1	32

reserved: This is a 3-bit field that shall be set to "111".

max_sections_per_datagram: This is an 8-bit field that shall indicate the maximum number of sections that can be used to carry a single datagram unit.

7.2.2 Stream type

The presence of a multiprotocol data stream in a service shall be indicated in the program map section of that service by setting the stream type of that stream to the value of 0x0D (see ISO/IEC 13818-6 [5]) or a user defined value.

8 IP/MAC Notification Table signalling for Multiprotocol Encapsulation

Please note that the address resolution for MultiProtocol Encapsulation (MPE) was agreed after the specification of the MPE itself. Therefore its use is optional but highly recommended.

8.1 Definitions, scope and types of services

8.1.1 Definitions

For the purposes of the present section, the following terms and definitions apply:

IP/MAC platform: A set of IP/MAC streams and/or receiver devices managed by an organization. The IP/MAC platform represents a harmonized IP/MAC address space (i.e. one without address conflicts). An IP/MAC platform may span several transport streams within one or multiple DVB networks. Several IP/MAC platforms may co-exist in the same transport stream. An IP/MAC platform is identified by its platform_id.

IP/MAC stream: A data stream including an address header containing an IP and/or MAC address. It is encapsulated in an MPEG-2 Transport Stream multiplex. An example would be an IP multicast stream conveyed in MPE sections.

Platform_id: A platform_id uniquely identifies an IP/MAC platform. Once a receiver has chosen or is assigned to an IP/MAC platform, all IP/MAC addresses have unique meanings. There cannot be any address conflicts, i.e. different IP/MAC streams using the same IP/MAC addresses, within a given IP/MAC platform. The platform_id is assigned by DVB Project Office and published in TR 101 162 [3]. It allows a unique identification of an IP/MAC platform. Platform_ids should not be presented directly to the user. For this reason the standard allows a platform_id to be associated with a human readable platform_name.

8.1.2 Scope of the IP/MAC Notification Table

The following clauses define a method to handle information about IP/MAC streams within DVB networks. In particular:

- It defines the IP/MAC Notification Table (INT). This table provides a flexible mechanism for carrying information about the location of IP/MAC streams within DVB networks. Through the use of a flexible syntax, extensive targeting and notification descriptor mechanisms, the table can be easily extended to cover additional requirements in the DVB IP/MAC domain.

- It introduces the IP/MAC platform concept: An IP/MAC platform represents a set of IP/MAC streams and/or receiver devices. Such a data platform may span several transport streams within one or multiple DVB networks and represents a single IP network with a harmonized address space (i.e. one without address conflicts). The IP/MAC platform concept allows for the coexistence of several non-harmonized IP/MAC address spaces on the same DVB network. Note that several non-harmonized IP/MAC address spaces (IP/MAC platforms) may co-exist on a single transport stream.

- It defines a signalling mechanism for locating the INT under various network configurations. The INT can be accessed via one or more linkage descriptors located in the NIT or via the IP/MAC Notification BAT. The IP/MAC Notification BAT/NIT itself may be available on one or more transport streams on the network, and located via a further level of indirection through a linkage descriptor in the NIT. This last scheme is mainly applicable for very large networks.

The present document should be seen in context with TR 101 162 [3] and EN 300 468 [2] as it describes descriptors to be used in the *IP Notification Table*.

8.1.3 Types of IP/MAC Notification Services

The structure of the INT allows multiple types of notification. The discrimination is based on the **action_type** tag.

The present document defines one IP/MAC Notification type, which signals the location of IP/MAC streams in DVB networks.

8.2 Network (SI) signalling

The linkage descriptor with the linkage type of 0x0B (*IP/MAC Notification* service) specifies a transport stream and a service carrying an *IP/MAC Notification Table* within a network or bouquet. This descriptor shall be carried in the first loop of the NIT or in the first loop of a specifically identified BAT (called *IP/MAC Notification* BAT).

The preferred location of this descriptor is in the NIT. However, in case it is prohibitive to an operator to carry this descriptor in the NIT (e.g. due to size constraints of the table), the descriptor may be located in the *IP/MAC Notification* BAT. The *IP/MAC Notification* BAT is identified by the *IP/MAC Notification* bouquet_id. Allocations of the value of bouquet_id are found in the TR 101 162 [3].

The linkage descriptor may occur more than once, for example if the IP/MAC Notification Table is transmitted on several transport streams within the network. All IP/MAC Notification Tables within a network shall be referred to by at least one linkage descriptor of type 0x0B.

If the *IP/MAC Notification* BAT/NIT announces multiple IP/MAC platforms, a receiver would typically need to choose between the platforms. In such case, if the receiver was not designed to work only on a given platform_id, it might present to the user a choice based on the platform_name fields. Note however, that some receivers may be able to be part of multiple IP/MAC platforms simultaneously.

A DVB data receiver following this implementation would access the NIT on its default transport stream. It would scan this NIT for one or more linkage descriptors of type 0x0B. If more than one linkage descriptor is present it checks whether the platform_ids are identical. If they are, the receiver knows that the IP/MAC platform consists of more than one transport stream each carrying an INT. If they are different it might prompt the user to select an IP/MAC platform.

For some networks it may not be practical to have many linkage descriptors of type 0x0B in the NIT. Therefore if a data receiver cannot locate a linkage_descriptor of type 0x0B it checks for linkage_type 0x0C and if found, scans an additional intermediary BAT/NIT searching for linkage types 0x0B. Intermediary tables should be cached by the receiver to allow it rapid access to the INT when switching transport streams.

If a receiver does not find a valid linkage descriptor of type 0x0B or 0x0C in the actual Transport Stream, it may check for the presence of a linkage descriptor of type 0x04 pointing to a Transport Stream containing complete Network/Bouquet SI.

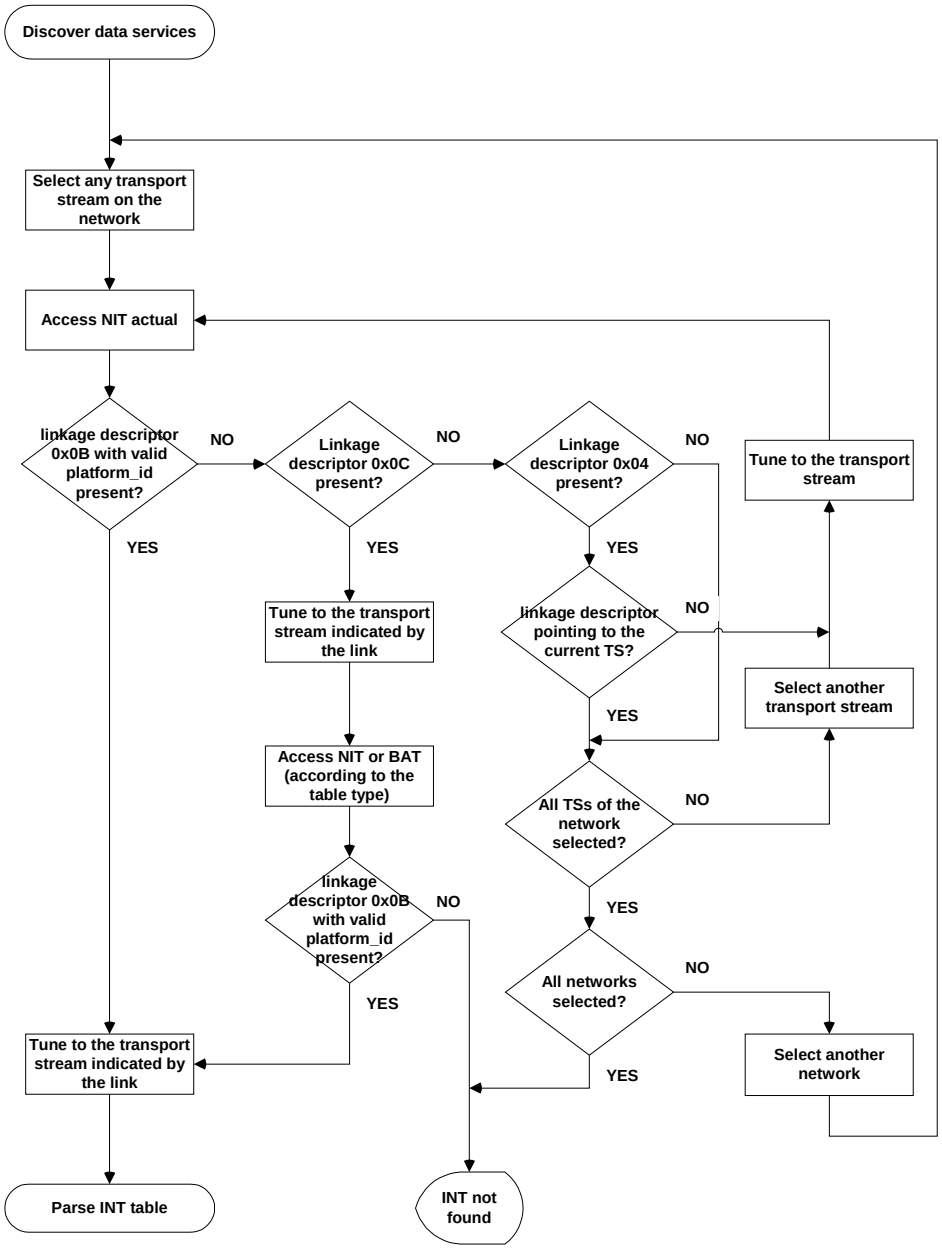

Figure 2: Example procedure to locate IP/MAC Notification Table for a receiver which knows, in advance, the platform_id

8.2.1 Linkage descriptor for an IP/MAC Notification Table

Table 9: Syntax for the linkage descriptor of type 0x0B

Syntax	Number of bits	Identifier
linkage_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
transport_stream_id	16	uimsbf
original_network_id	16	uimsbf
service_id	16	uimsbf
linkage_type	8	uimsbf
if (linkage_type == 0x0B) {		
platform_id_data_length	8	uimsbf
for (i=0; i<N; i++) {		
platform_id	24	uimsbf
platform_name_loop_length	8	uimsbf
for (i=0; i<N; i++) {		
ISO_639_language_code	24	bslbf
platform_name_length	8	uimsbf
for (i=0; i<platform_name_length; i++) {		
text_char	8	uimsbf
}		
}		
}		
for (i=0; i<N; i++) {		
private_data_byte	8	uimsbf
}		
}		
}		

Semantics of the private data bytes for linkage type 0x0B:

transport_stream_id: This is a 16-bit field which identifies the TS containing the INT.

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system of the transport stream carrying the INT.

service_id: This is a 16-bit field which identifies the service containing the INT.

linkage_type: This is an 8-bit field specifying the type of linkage, and shall be set to 0x0B.

platform_id_data_length: This field specifies the total length in bytes of the following platform_id-loop.

platform_id: This is a 24-bit field containing a platform_id of the organization providing IP/MAC streams on DVB transport streams/services. Allocations of the value of platform_id are found in the TR 101 162 [3].

platform_name_loop_length: This 8-bit field defines the length in bytes of the following platform name loop.

ISO_639_language_code: This 24-bit field contains the ISO 639-2 [15] three character language code of the language of the following platform name. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO 8859-1 [14] and inserted in order into the 24-bit field.

platform_name_length: This 8-bit field specifies the total length in bytes of the following platform_name.

text_char: This is an 8-bit field. A string of "text_char" fields specifies the platform name as described above. Text information is coded using the character sets and methods described in annex A of EN 300 468 [2].

private_data_byte: This is an 8-bit field, the value of which is privately defined.

8.2.2 Deferred linkage descriptor for IP/MAC Notification Tables

This linkage descriptor defines a pointer to a transport stream carrying an *IP/MAC Notification* BAT or NIT with detailed signalling information about *IP/MAC Notification Tables*. The linkage type for this descriptor shall be 0x0C and when present, shall be inserted into a NIT.

It is different from a linkage descriptor of type 0x0B in the sense that this descriptor does not contain any platform_id specific data. It may be exploited by the receiver to quickly acquire the multiplex carrying the *IP/MAC Notification* BAT or NIT without needing to scan all multiplexes.

The use of the linkage descriptor of type 0x0C is therefore complementary to the use of the linkage descriptor of type 0x0B in the NIT or *IP/MAC Notification* BAT. The table_type field indicates whether the *IP/MAC Services Notification* Scan Linkage Descriptor points to a NIT or BAT on the target transport stream. In case of pointing to an *IP/MAC Notification* BAT, this descriptor shall contain the bouquet_id of the corresponding *IP/MAC Notification* BAT, and the descriptor may occur more than once. The use of this descriptor is optional.

Table 10: Syntax for the Linkage Descriptor of type 0x0C

Syntax	Number of bits	Identifier
`linkage_descriptor(){`		
`  descriptor_tag`	8	uimsbf
`  descriptor_length`	8	uimsbf
`  transport_stream_id`	16	uimsbf
`  original_network_id`	16	uimsbf
`  service_id`	16	uimsbf
`  linkage_type`	8	uimsbf
`  if (linkage_type == 0x0C){`		
`  table_type`	8	bslbf
`  if (table_type == 0x02){`		
`  bouquet_id`	16	uimsbf
`  }`		
`  }`		
`}`		

Semantics for the linkage descriptor of type 0x0C:

transport_stream_id: This is a 16-bit field which identifies the TS containing the *IP/MAC Notification* BAT or NIT.

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system of *IP/MAC Notification* BAT or NIT indicated.

service_id: This is a 16-bit field which is not relevant, and shall be set to 0x0000.

linkage_type: This is an 8-bit field specifying the type of linkage, and shall be set to 0x0C.

table_type: This is an 8-bit field containing a flag pointing either to the *IP/MAC Notification* BAT or NIT.

Table 11: Table_type values

Value	Description
0x00	not defined
0x01	NIT
0x02	BAT
0x03 to 0xFF	reserved for future use

bouquet_id: This is a 16-bit field which serves as a label to identify the bouquet described by the IP/MAC Notification BAT.

8.3 PSI Signalling

The IP/MAC Notification Table (INT) is signalled as a DVB service. The PMT of the transport stream carrying the *INT* shall contain the data_broadcast_id_descriptor with the data_broadcast_id of 0x000B to indicate the elementary stream used for the *IP/MAC Notification* Table.

This descriptor provides the reference to an IP/MAC Notification Table and hence is considered essential.

The descriptor may contain specific platform_ids, in which case the list of platform_ids shall be complete.

The data_broadcast_id_descriptor in the PMT shall contain the reference (platform_id and action_type) of each INT sub-table which is broadcast in the corresponding elementary stream.

8.3.1 Data broadcast Id descriptor selector byte definition for *IP/MAC Notification Table*

Table 12: Syntax for the IP/MAC_notification_info structure

Syntax	Number of bits	Identifier
IP/MAC_notification_info(){		
platform_id_data_length	8	uimsbf
for (i=0; i<N; i++){		
platform_id	24	uimsbf
action_type	8	uimsbf
reserved	2	bslbf
INT_versioning_flag	1	bslbf
INT_version	5	uimsbf
}		
}		
for (i=0; i<N; i++){		
private_data_byte	8	uimsbf
}		
}		

Semantics of the IP/MAC Notification info for data_broadcast_id 0x000B:

platform_id_data_length: This field specifies the total length in bytes of the following platform_id-loop.

platform_id: This is a 24-bit field which serves as a label to uniquely identify this IP/MAC platform. Allocations are specified in TR 101 162 [3].

action_type: This is an eight-bit field that shall be equal to the action_type field defined in the INT as indicated in table 14.

INT_versioning_flag: if it is set to 0 no relevant versioning information is carried in the version field. If it is set to 1 the INT_version field shall reflect changes in the INT.

INT_version: If the INT_version_flag is set to 1, the version shall be incremented on each change of the INT and shall be the same as the version_number in the INT section header.

private_data_byte: This is an 8-bit field, the value of which is privately defined.

8.4 IP/MAC Notification Table

8.4.1 Description

The IP/MAC Notification Table (INT) is used with the data_broadcast_id_descriptor (0x000B) where the action_type is set to one of the values specified in table 14.

The INT is broadcast in SI table format; the format is laid out in clause 5 of EN 300 468 [2]. The maximum length for an INT section is 4 096 bytes.

The INT is divided into sub-tables. An INT sub-table is a collection of sections with the same value of table_id, platform_id, action type and version number. The INT sub-table is identified by its platform_id and action_type. The platform_id_hash is intended for fast (e.g. hardware) filtering but as it is not unique, the full platform_id should be used to identify the sub-table.

Each IP/MAC stream shall be announced by an INT sub-table with action_type 0x01 within the same transport stream. Optionally, an INT sub-table may also announce IP/MAC streams on other transport streams.

8.4.2 PSI, SI and related INT signalling

The PMT references the INT by including the data_broadcast_id_descriptor (data_broadcast_id = 0x000B) in the ES_info loop, where the action_type in the *IP/MAC_Notification_info* is set to INT action_type field value. The *IP/MAC_Notification_info* platform_id must contain a valid platform_id corresponding to an INT platform_id.

Once a candidate INT has been selected, a search of the table for a sub-table corresponding to the action_type and the IP/MAC platform's platform_id may proceed.

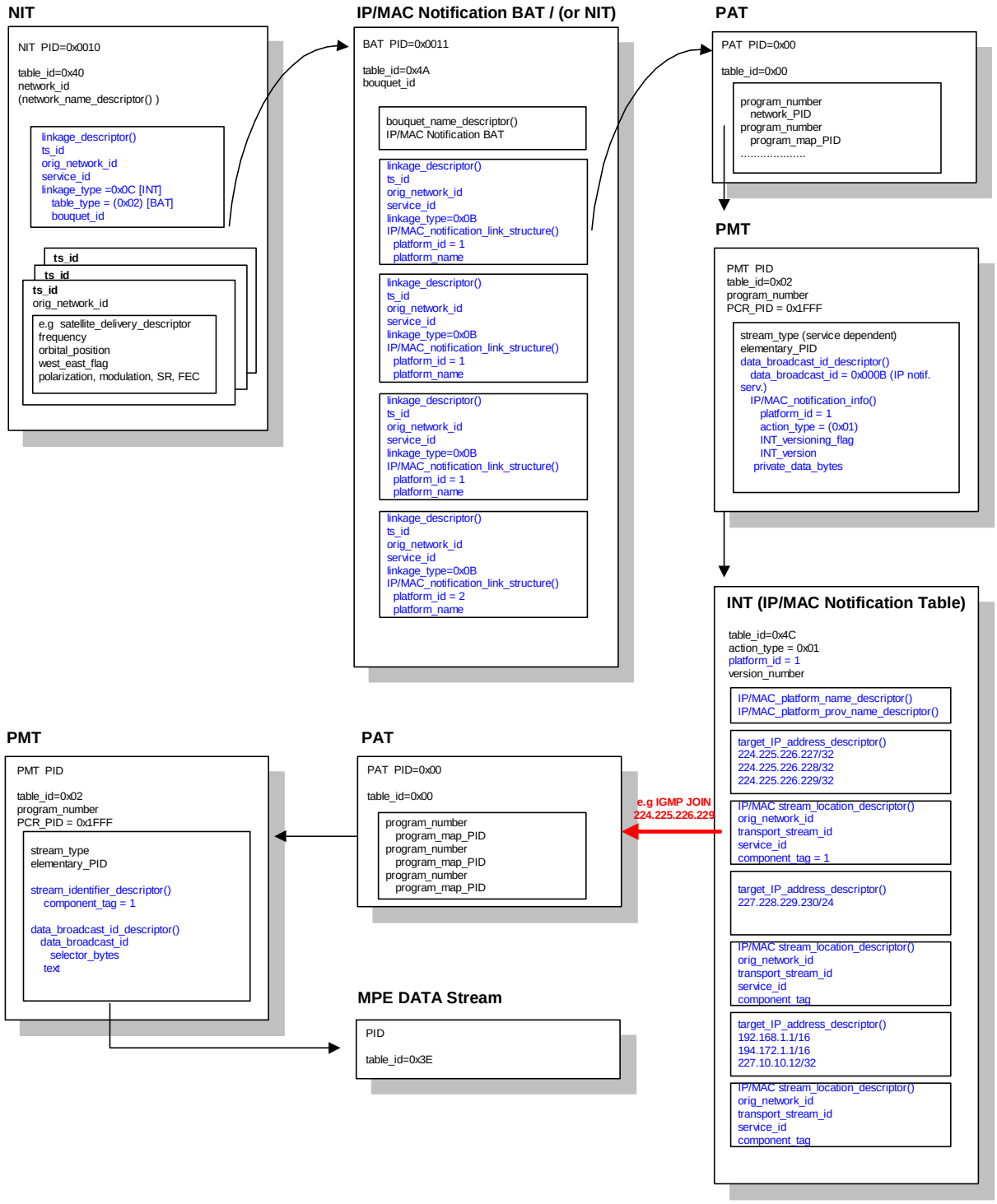

Figure 3: Example for IP/MAC streams with INT reference

8.4.3 Description of the IP/MAC Notification Table

The INT is used for several purposes. The present document describes one application area for the INT:

- The signalling of the availability and location of IP/MAC streams in DVB networks (action_type=0x01).

The INT describes the availability and location of IP/MAC streams. There may be one or many INTs covering all IP/MAC streams for a network. The INT is referenced by the data_broadcast_id_descriptor (data_broadcast_id = 0x000B), in the ES_info loop of the PMT, where the action_type in the *IP/MAC_Notification_info_structure*, is equal to the action_type field in the INT. The platform_id has to correspond with the platform_id field coded in one of the sub-tables of the INT.

To assist receiver devices with limited section filter capabilities in locating an appropriate INT sub-table, the platform_id in the INT sub-table is hashed using a simple XOR function and included as part of the table_id_extension (table_id_extension = platform_id_hash + action_type). The definition of the INT has been limited to IP/MAC stream announcements/location information. The need for other action types was however foreseen during the specification process. In an effort to facilitate these, yet to be defined actions, an action_type field forms part of the sub-table index that means part of the table_id_extension. In addition the processing_order field is provided to assist in selecting the most appropriate action_type.

The INT is divided into sub-tables using standard DVB table syntax. As such, there may be one or more sections forming the sub-table. Note that all sub-tables with the same platform_id but possibly different action_type values form a logical group since they together convey all information pertaining to devices covered by the respective platform_id holder, but they form different sub-tables.

A sub-table section is further divided into 2 independent loops:

- The 1st loop (platform_descriptor_loop) is used to describe the IP/MAC platform (related to the platform_id).
- The 2nd loop associates an operational_descriptor_loop with a target_descriptor_loop.

The target_descriptor_loop contains zero or more descriptors which are used exclusively for targeting. If the loop contains at least one descriptor, the receiver device must be explicitly targeted by at least one descriptor. If the loop is empty, all receivers under the related platform_id shall be concerned.

The operational_descriptor_loop mostly contains descriptors relating to the localization or assignment processes. Descriptors in this loop take precedence over (or override) descriptors in the platform_descriptor_loop unless specifically stated otherwise in the descriptor's functional description. The operational_descriptor_loop may be empty, implying no additional descriptors are necessary (to those specified in the platform_descriptor_loop).

8.4.4 Semantics of the INT

Syntax:

Table 13: Syntax of the IP/MAC_notification_section

Name	Number of bits	Identifier	Remarks
IP/MAC_notification_section() {			
table_id	8	uimsbf	0x4C
section_syntax_indicator	1	bslbf	1b
reserved_for_future_use	1	bslbf	1b
reserved	2	bslbf	11b
section_length	12	uimsbf	
action_type	8	uimsbf	see table 14
platform_id_hash	8	uimsbf	
reserved	2	bslbf	11b
version_number	5	uimsbf	
current_next_indicator	1	bslbf	1b
section_number	8	uimsbf	
last_section_number	8	uimsbf	
platform_id	24	uimsbf	
processing_order	8	uimsbf	0x00
platform_descriptor_loop()			
for (i=0, i<N1, i++) {			
target_descriptor_loop()			
operational_descriptor_loop()			
}			
CRC_32	32	rpchof	
}			

8.4.4.1 Fields description

table_id: Uniquely defined for IP/MAC Notification Table (0x4C).

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. It specifies the number of bytes of the section, starting immediately following the section_length fields and including the CRC. The section_length shall not exceed 4 093 so that the entire section has a maximum length of 4 096.

action_type: Identifies the action to be performed. Coded according to table 14.

Table 14: action_type coding

action_type	Action Specification
0x00	reserved
0x01	location of IP/MAC streams in DVB networks
0x02 to 0xFF	reserved for future use

platform_id_hash: The platform_id_hash is formed by XORing all three bytes of the platform_id together to form a single byte value (platform_id_hash = platform_id[23..16] ^ platform_id [15..8] ^ platform_id [7..0]).

version_number: This 5-bit field is the version number of the sub-table. The version_number shall be incremented by 1 when a change in the information carried within the sub_table occurs. When it reaches value 31, it wraps around to 0. When the current_next_indicator is set to '1', then the version_number shall be that of the currently applicable sub_table defined by the table_id, platform_id and action_type. When the current_next_indicator is set to '0', then the version_number shall be that of the next applicable sub_table defined by the table_id, platform_id and action_type.

current_next_indicator: This 1-bit indicator, when set to '1' indicates that the sub_table is the currently applicable sub_table. When the bit is set to '0', it indicates that the sub_table sent is not yet applicable and shall be the next sub_table to be valid.

section_number: This 8-bit field gives the number of the section. The section_number of the first section in the sub_table shall be "0x00". The section_number shall be incremented by 1 with each additional section with the same table_id, platform_id and action_type.

last_section_number: This 8-bit field indicates the number of the last section (that is, the section with the highest section_number) of the sub_table of which this section is part.

platform_id: This is a 24 bit field which serves as a label to identify a given IP/MAC platform. Allocation of the value of this field are found in the TR 101 162 [3].

processing_order: Indicates the sequence in which to perform actions. If the INT requires more than one action this field can be used to indicate the order to perform these actions. Coded according to table 15. In the same sense, if more than one INT sub-table is available for the same platform_id, this field can be used to set up a priority in the resolution of the IP/MAC addresses.

Table 15: processing_order coding

processing_order	Specification
0x00	first action
0x01 to 0xFE	subsequent actions (ascending)
0xFF	no ordering implied

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 EN 300 468 [2] after processing the entire private section.

8.4.4.2 platform_descriptor_loop

The platform_descriptor_loop carries information about the IP/MAC platform.

Table 16: Syntax of the platform_descriptor_loop

Name	Number of bits	Identifier
platform_descriptor_loop () {		
reserved	4	bslbf
platform_ descriptor_loop_length	12	uimsbf
for (i=0; i<N1; i++)		
platform_descriptor()		
}		

8.4.4.3 target_descriptor_loop

The target_descriptor_loop discriminates between individual devices. This descriptor loop may contain target IP/MAC address, smartcard or private, etc, descriptors. This descriptor loop forms a list of all target devices to be addressed and the operational loop applied. If this descriptor loop is empty, the operational loop applies to all devices.

A receiver device not recognizing a target descriptor (new or unknown target descriptor) must assume this target descriptor does not target this receiver device.

Table 17: Syntax of the target_descriptor_loop

Name	Number of bits	Identifier
target_descriptor_loop () {		
reserved	4	bslbf
target_descriptor_loop_length	12	uimsbf
for (i = 0; i < N1; i++)		
target_descriptor()		
}		

8.4.4.4 operational_descriptor_loop

The operational_descriptor_loop contains action, informational, and operational descriptors, which apply only to those target devices that meet the requirements of the target descriptor loop.

Table 18: Syntax of the operational_descriptor_loop

Name	Number of bits	Identifier
operational_descriptor_loop () {		
reserved	4	bslbf
operational_descriptor_loop_length	12	uimsbf
for (i = 0; i < N1; i++)		
operational_descriptor()		
}		

8.4.5 INT descriptors

8.4.5.1 Descriptor identification and location

Table 19: INT descriptors

Descriptor	Tag Value	Allowed in Loop		
		Platform	Target	Operational
reserved	0x00			
target_smartcard_descriptor	0x06		*	
target_MAC_address_descriptor	0x07		*	
target_serial_number_descriptor	0x08		*	
target_IP_address_descriptor	0x09		*	
target_IPv6_address_descriptor	0x0A		*	
IP/MAC_platform_name_descriptor	0x0C	*		
IP/MAC_platform_provider_name_descriptor	0x0D	*		
target_MAC_address_range_descriptor	0x0E		*	
target_IP_slash_descriptor	0x0F		*	
target_IP_source_slash_descriptor	0x10		*	
target_IPv6_slash_descriptor	0x11		*	
target_IPv6_source_slash_descriptor	0x12		*	
IP/MAC_stream_location_descriptor	0x13	*		*
ISP_access_mode_descriptor	0x14	*		*
telephone_descriptor	0x57	*		*
private_data_specifier_descriptor	0x5F	*	*	*
user private	0x80 to 0xFE			
reserved	0xFF			

Descriptors from the DVB SI range (0x40 to 0x7F) shall have their standard semantics as defined in EN 300 468 [2]. Equally MPEG descriptors in the range 0x00 to 0x3F can not be used in the INT.

NOTE: Descriptor tags from 0x00 to 0x3F share a common descriptor name space with UNT descriptors (see TS 102 006 [19]).

All descriptors may appear more than once in the INT sub-table at the locations indicated above. In the case of name_descriptors multiple occurrences require different ISO 639 language codes.

8.4.5.2 IP/MAC_platform_name_descriptor

Location: **Loop of table:** platform

Action_type: all

This descriptor is to be used to provide the name of the IP/MAC platform.

Table 20: Syntax of the IP/MAC_platform_name_descriptor

Name	Number of bits	Identifier	Remarks
`IP/MAC_platform_name_descriptor () {`			
`descriptor_tag`	8	uimsbf	0x0C
`descriptor_length`	8	uimsbf	
`ISO_639_language_code`	24	bslbf	
`for (i=0; i<N; i++) {`			
`text_char`	8	uimsbf	
`}`			
`}`			

ISO_639_language_code: This 24-bit field contains the ISO 639-2 [15] three character language code of the language of the following platform name. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO 8859-1 [14] and inserted in order into the 24-bit field.

EXAMPLE: French has 3-character code "fre", which is coded as:
'**0110 0110 0111 0010 0110 0101**".

text_char: This is an 8-bit field. A string of "text_char" fields specifies the platform name. Text information is coded using the character sets and methods described in annex A of EN 300 468 [2].

8.4.5.3 IP/MAC_platform_provider_name_descriptor

Location: **Loop of table:** platform

Action_type: all

This descriptor is to be used to provide the name of the IP/MAC platform provider.

Table 21: Syntax of the IP/MAC_platform_provider_name_descriptor

Name	Number of bits	Identifier	Remarks
IP/MAC_platform_provider_name_descriptor () {			
descriptor_tag	8	uimsbf	0x0D
descriptor_length	8	uimsbf	
ISO_639_language_code	24	bslbf	
for (i=0; i<N; i++) {			
text_char	8	uimsbf	
}			
}			

ISO_639_language_code: This 24-bit field contains the ISO 639-2 [15] three character language code of the language of the following platform provider name. Both ISO 639-2/B and ISO 639-2/T may be used. Each character is coded into 8 bits according to ISO 8859-1 [14] and inserted in order into the 24-bit field.

EXAMPLE: French has 3-character code "fre", which is coded as:
'**0110 0110 0111 0010 0110 0101".**

text_char: This is an 8-bit field. A string of "text_char" fields specifies the platform provider name. Text information is coded using the character sets and methods described in annex A of EN 300 468 [2].

8.4.5.4 target_serial_number_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor targets an action for a specific receiver device.

Table 22: Syntax of the serial_number_descriptor

Name	Number of bits	Identifier	Remarks
target_serial_number_descriptor () {			
descriptor_tag	8	uimsbf	0x08
descriptor_length	8	uimsbf	
for (i=0; i<N; i++) {			
serial_data_byte	8	uimsbf	
}			
}			

serial_data_byte: This information is intended to target devices based on some manufacturing id. No further definition on the semantics of this information is implied.

8.4.5.5 target_smartcard_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor targets a specific receiver device based upon its smartcard identifier. The smart card identifier is conveyed in the private data bytes.

Table 23: Syntax of the target_smartcard_descriptor

Name	Number of bits	Identifier	Remarks
target_smartcard_descriptor () {			
descriptor_tag	8	uimsbf	0x06
descriptor_length	8	uimsbf	
super_CA_system_id	32	uimsbf	
for (i=0; i<N; i++) {			
private_data_byte	8	uimsbf	
}			
}			

super_CA_system_id: DVB CA identifier as per TS 101 197-1 [22] (Simulcrypt).

Smart cards numbers are conveyed in the private_data field.

8.4.5.6 target_MAC_address_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices. The MAC_addr_mask field is used to define which bits of the MAC address are used for comparison. A bit set to one indicates that this bit shall be compared against the bit in the equivalent position of the MAC_addr field. Repeated MAC_addr fields are applied as a logical OR (i.e. a list of addressed receivers).

Table 24: Syntax of the target_MAC_address_descriptor

Name	Number of bits	Identifier	Remarks
target_MAC_address_descriptor () {			
descriptor_tag	8	uimsbf	0x07
descriptor_length	8	uimsbf	
MAC_addr_mask	48	uimsbf	
for (i=0; i<N; i++) {			
MAC_addr	48	uimsbf	
}			
}			

MAC_addr_mask: This is a 48-bit field, specifying the address mask.

MAC_addr: This is a 48-bit field, specifying a MAC address.

8.4.5.7 target_MAC_address_range_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a group of receiver devices falling in a range of MAC addresses. All receivers from MAC_addr_low up to and including MAC_addr_high are selected.

Table 25: Syntax of the target_MAC_address_range_descriptor

Name	Number of bits	Identifier	Remarks
target_MAC_address_range_descriptor () {			
descriptor_tag	8	uimsbf	0x0E
descriptor_length	8	uimsbf	
for (i=0; i<N; i++) {			
MAC_addr_low	48	uimsbf	
MAC_addr_high	48	uimsbf	
}			
}			

MAC_addr_low: A 48 bit field that contains the lowest MAC address in the range.

MAC_addr_high: A 48 bit field that contains the highest MAC address in the range.

8.4.5.8 target_IP_address_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices. The IPv4_addr_mask field is used to define which bits of the IP address are used for comparison. A bit set to one indicates that this bit shall be compared against the bit in the equivalent position of the IPv4_addr field. Repeated IPv4_addr fields are applied as a logical OR (i.e. a list of addressed receivers).

Table 26: Syntax of the target_IP_address_descriptor

Name	Number of bits	Identifier	Remarks
target_IP_address_descriptor () {			
descriptor_tag	8	uimsbf	0x09
descriptor_length	8	uimsbf	
IPv4_addr_mask	32	uimsbf	
for (i=0; i<N; i++) {			
IPv4_addr	32	uimsbf	
}			
}			

IPv4_addr_mask: A 32-bit field that specifies the IPv4 mask.

IPv4_addr: A 32-bit field that specifies an IPv4 unicast/multicast/broadcast address. The IPv4 address is fragmented into 4 fields of 8 bits where the first byte contains the most significant byte of the IPv4 address (dotted decimal notation).

8.4.5.9 target_IP_slash_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices, through their IPv4_address(es). The IP address notation includes a subnet mask in the shortform slash format e.g. XXX.XXX.XXX.XXX/NN. This allows host or subnet announcements in unicast. In multicast it provides the possibility to define address ranges more efficiently.

Table 27: Syntax of the target_IP_slash_descriptor

Name	Number of bits	Identifier	Remarks
target_IP_slash_descriptor () {			
descriptor_tag	8	uimsbf	0x0F
descriptor_length	8	uimsbf	
for (I=0; i<N; i++) {			
IPv4_addr	32	uimsbf	
IPv4_slash_mask	8	uimsbf	
}			
}			

IPv4_address: A 32 bit field that specifies an IPv4 unicast/multicast/broadcast address. The IPv4 address is fragmented into 4 fields of 8 bits where the first byte contains the most significant byte of the IPv4 address (dotted notation).

IPv4_slash_mask: An 8 bit field that specifies the IPv4 mask in slash ("/") or shortform notation e.g. 192.168.37.1/24.

8.4.5.10 target_IP_source_slash_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices, through both IPv4_source and destination address(es). The IP address notation includes a subnet mask in the shortform slash format.

Table 28: Syntax of the target_IP_source slash_descriptor

Name	Number of bits	Identifier	Remarks
target_IP_slash_descriptor () {			
descriptor_tag	8	uimsbf	0x10
descriptor_length	8	uimsbf	
for (i=0; i<N; i++) {			
IPv4_source_addr	32	uimsbf	
IPv4_source_slash_mask	8	uimsbf	
IPv4_dest_addr	32	uimsbf	
IPv4_dest_slash_mask	8	uimsbf	
}			
}			

IPv4_source_address: A 32 bit field that specifies an IPv4 unicast/multicast/broadcast source address. The IPv4 address is fragmented into 4 fields of 8 bits where the first field contains the most significant block of the IPv4 address (dotted notation).

IPv4_source_slash_mask: An 8 bit field that specifies the IPv4 mask in slash ("/") or shortform notation e.g. IP address/subnetmask.

IPv4_dest_address: A 32 bit field that specifies an IPv4 unicast/multicast/broadcast destination address. The IPv4 address is fragmented into 4 fields of 8 bits where the first field contains the most significant block of the IPv4 address (dotted notation).

IPv4_dest_slash_mask: An 8 bit field that specifies the IPv4 mask in slash ("/") or shortform notation e.g. IP address/subnetmask.

8.4.5.11 target_IPv6_address_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices. The IPv6_addr_mask field is used to define which bits of the IPv6 address are used for comparison. A bit set to one indicates that this bit shall be compared against the bit in the equivalent position of the IPv6_addr field. Repeated IPv6_addr fields are applied as a logical OR (i.e. a list of addressed receivers).

Table 29: Syntax of the target_IPv6_address_descriptor

Name	Number of bits	Identifier	Remarks
target_IPv6_address_ descriptor () {			
descriptor_tag	8	uimsbf	0x0A
descriptor_length	8	uimsbf	
IPv6_addr_mask	128	uimsbf	
for (I=0;i<N;I++) {			
IPv6_addr	128	uimsbf	
}			
}			

IPv6_addr_mask: A 128-bit field that specifies the IPv6 mask.

IPv6_addr: A 128 bit field that specifies an IPv6 address. The IPv6 address is fragmented into 16 fields of 8 bits where the first byte contains the most significant byte of the IPv6 address.

8.4.5.12 target_IPv6_slash_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices, through their IPv6_address(es). The IP address notation includes a subnet mask in the shortform slash format.

Table 30: Syntax of the target_IPv6_slash_descriptor

Name	Number of bits	Identifier	Remarks
target_IPv6_slash_descriptor () {			
descriptor_tag	8	uimsbf	0x11
descriptor_length	8	uimsbf	
for (i=0; i<N; i++) {			
IPv6_addr	128	uimsbf	
IPv6_slash_mask	8	uimsbf	
}			
}			

IPv6_address: A 128 bit field that specifies an IPv6 unicast/multicast/anycast address. The IPv6 address is fragmented into 8 fields of 16 bits where the first field contains the most significant block of the IPv6 address (dotted notation).

IPv6_slash_mask: An 8 bit field that specifies the IPv6 mask in slash ("/") or shortform notation e.g. IP address/subnetmask.

8.4.5.13 target_IPv6_source_slash_descriptor

Location: **Loop of table:** target

Action_type: 0x01

This descriptor is used to target a single, or a group of receiver devices, through both IPv6_source and destination address(es). The IP address notation includes a subnet mask in the shortform slash format.

Table 31: Syntax of the target_IPv6_source slash_descriptor

Name	Number of bits	Identifier	Remarks
target_IPv6_source_slash_descriptor () {			
descriptor_tag	8	uimsbf	0x12
descriptor_length	8	uimsbf	
for (i=0; i<N; i++) {			
IPv6_source_addr	128	uimsbf	
IPv6_source_slash_mask	8	uimsbf	
IPv6_dest_addr	128	uimsbf	
IPv6_dest_slash_mask	8	uimsbf	
}			
}			

IPv6_source_address: A 128 bit field that specifies an IPv6 unicast/multicast/anycast source address. The IPv6 address is fragmented into 8 fields of 16 bits where the first field contains the most significant block of the IPv6 address (dotted notation).

IPv6_source_slash_mask: An 8 bit field that specifies the IPv6 mask in slash ("/") or shortform notation e.g. IP address/subnetmask.

IPv6_dest_address: A 128 bit field that specifies an IPv6 unicast/multicast/anycast destination address. The IPv6 address is fragmented into 8 fields of 16 bits where the first field contains the most significant block of the IPv6 address (dotted notation).

IPv6_dest_slash_mask: An 8 bit field that specifies the IPv6 mask in slash ("/") or shortform notation e.g. IP address/subnetmask.

8.4.5.14 IP/MAC stream_location_descriptor

Location: **Loop of table:** platform, operational

Action_type: 0x01

This descriptor directly locates the IP/MAC stream in a DVB network via the parameters network_id, original_network_id, transport_stream_id, service_id and component tag.

Table 32: Syntax of the IP/MAC_stream_location_descriptor

Name	Number of bits	Identifier	Remarks
IP/MAC_stream_location_descriptor () {			
descriptor_tag	8	uimsbf	0x13
descriptor_length	8	uimsbf	
network_id	16	uimsbf	
original_network_id	16	uimsbf	
transport_stream_id	16	uimsbf	
service_id	16	uimsbf	
component_tag	8	uimsbf	
}			

network_id: This is a 16-bit field which serves as a label to identify the delivery system from any other delivery system. Allocations of the value of this field are found in TR 101 162 [3].

original_network_id: This 16-bit field gives the label identifying the network_id of the originating delivery system.

transport_stream_id: This is a 16 bit field which serves as a label for identification of the present document from any other multiplex within the delivery system.

service_id: This is a 16 bit field which serves as a label to identify this service from any other service within the TS. The service_id is the same as the program_number in the corresponding program_map_section.

component_tag: This 8-bit field identifies the component stream for associating it with a description given in a component descriptor. Within a program map section each stream identifier descriptor shall have a different value for this field.

8.4.5.15 ISP_access_mode_descriptor

Location: **Loop of table:** platform, operational

Action_type: 0x01

This descriptor is used to give information about accessing the ISP through alternative media (PSTN, etc.) rather than the DVB network where the IP/MAC_stream_location_descriptor may be used.

Both DVB and alternative media may be used at the same type (asymmetric connection) if both descriptors are used at the same time.

Table 33: Syntax of the ISP_access_mode_descriptor

Name	Number of bits	Identifier	Remarks
ISP_access_mode_descriptor () {			
descriptor_tag	8	uimsbf	0x14
descriptor_length	8	uimsbf	
access_mode	8	bslbf	
}			

access_mode: This is an 8-bit field indicating the access mode. Coded according to table 34.

Table 34: Access mode values

Access_mode value	Access mode
0x00	not used
0x01	dialup
0x02 to 0xFF	reserved for future use

In case of dialup access mode, at least one telephone descriptor shall be present, indicating the telephone number(s) used to access the referred IP/MAC stream.

8.4.5.16 telephone descriptor (Informative)

Location: **Loop of table:** platform, operational

Action_type: 0x01

The telephone descriptor may be used to indicate a telephone number which may be used to access the IP services or ISP provider in conjunction with a modem (PSTN or ISDN) to use narrow band interactive channels. The use of the Point to Point Protocol (PPP) according to RFC 1661 [21] is assumed.

Table 35: Syntax of the telephone_descriptor

Name	Number of bits	Identifier	Remarks
telephone_descriptor() {			
descriptor_tag	8	uimsbf	0x57
descriptor_length	8	UiMSBf	
reserved_future_use	2	Bslbf	
foreign_availability	1	Bslbf	
connection_type	5	UiMSBf	
reserved_future_use	1	Bslbf	
country_prefix_length	2	UiMSBf	
international_area_code_char	3	UiMSBf	
operator_code_length	2	UiMSBf	
reserved_future_use	1	Bslbf	
national_area_code_length	3	UiMSBf	
core_number_length	4	UiMSBf	
for (i=0; i<N; i++){			
country_prefix_char	8	uimsbf	
}			
for (i=0; i<N; i++){			
international_area_code_char	8	uimsbf	
}			
for (i=0; i<N; i++){			
operator_code_char	8	uimsbf	
}			
for (i=0; i<N; i++){			
national_area_code_char	8	uimsbf	
}			
for (i=0; i<N; i++){			
core_number_char	8	uimsbf	
}			
}			

Semantics for the telephone_descriptor:

foreign_availability: This is a 1-bit flag. When set to "1" it indicates that the number described can be called from outside of the country specified by the country_prefix. When set to "0" it indicates that the number can only be called from inside the country specified by the country_prefix.

connection_type: This is a 5-bit field which indicates connection types. One example of the use of the connection type is to inform the IRD that when, if an interaction is initiated, if the connection is not made within 1 minute, then the connection attempt should be aborted.

country_prefix_length: This 2-bit field specifies the number of 8-bit alphanumeric characters in the country prefix.

international_area_code_length: This 3-bit field specifies the number of 8-bit alphanumeric characters in the international area code.

operator_code_length: This 2-bit field specifies the number of 8-bit alphanumeric characters in the operator code.

national_area_code_length: This 3-bit field specifies the number of 8-bit alphanumeric characters in the national area code.

core_number_length: This 4-bit field specifies the number of 8-bit alphanumeric characters in the core number.

country_prefix_char: This 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [14] gives one alphanumeric character of the country prefix.

international_area_code_char: This 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [14] gives one alphanumeric character of the international area code.

operator_code_char: This 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [14] gives one alphanumeric character of the operator code.

national_area_code_char: This 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [14] gives one alphanumeric character of the national area code.

core_number_char: This 8-bit field which shall be coded in accordance with ISO/IEC 8859-1 [14] gives one alphanumeric character of the core number.

8.4.5.17 private_data_specifier_descriptor (informative)

Location: **Loop of table:** all

Action_type: all

This descriptor is used to identify the specifier of any private descriptors or private fields within descriptors.

Table 36: Syntax of the private_data_specifier_descriptor

Name	Number of bits	Identifier	Remarks
private_data_specifier_descriptor () {			
descriptor_tag	8	uimsbf	0x5F
descriptor_length	8	uimsbf	
private_data_specifier	32	uimsbf	
}			

private_data_specifier: The assignment of values for this field is given in TR 101 162 [3].

9 Time Slicing and MPE-FEC

The contents of this clause are normative, unless otherwise noted.

9.1 Definitions

The following terms and definitions apply to the current section of the document.

Burst: Burst is a set of sections delivered on an elementary stream. Between two consecutive bursts there is a period of time when no sections are transmitted on the particular elementary stream. Each burst indicates the start time of the next burst within the elementary stream.

Conditional Access (CA): using DVB Common Scrambling Algorithm on Transport Stream packets.

Datagram: A network layer (OSI-layer 3) data frame. In the case of Internet Protocol, a datagram is an IP datagram.

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

EMM: entitlement management message; 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.

MPE-FEC: Method to deliver Reed Solomon (RS) parity data for datagrams delivered on Multi-Protocol Encapsulation (MPE) sections.

Receiver: Receiver is an entity within an end-user equipment consisting of Radio Frequency (RF) front-end, channel decoding and demultiplexing. Input of a Receiver is an RF signal, and the output is Network layer datagrams.

Time Slicing: Method to deliver MPE sections and MPE-FEC sections in bursts.

9.2 Time slicing (informative)

The concept of Time Slicing is to send data in bursts using significantly higher bitrate compared to the bitrate required if the data was transmitted using conventional bandwidth management. Within a burst, the time before the start of the next burst (delta-t) is indicated. Between the bursts, data of the elementary stream is not transmitted, allowing other elementary streams to use the bandwidth otherwise allocated. This enables a Receiver to stay active only for a fragment of the time, while receiving bursts of a requested service.

Time Slicing also supports the possibility to use the Receiver to monitor other transport streams (for example in case of a DVB-T network transport streams located in neighbouring cells) during the off-times. By accomplishing the switching between transport streams during an off period, the reception of a service is seemingly uninterrupted. In a normal DVB system without Time Slicing, a smooth switching between transport streams (hand-over) would require two Receivers in a single terminal.

9.2.1 Receiver (informative)

See clause 9.1 for definition of the term "Receiver" in the context of this clause. The Receiver supports access to services delivered on DVB transmission to a terminal.

Time Slicing enables the Receiver part to be periodically switched off, through which power saving may be achieved.

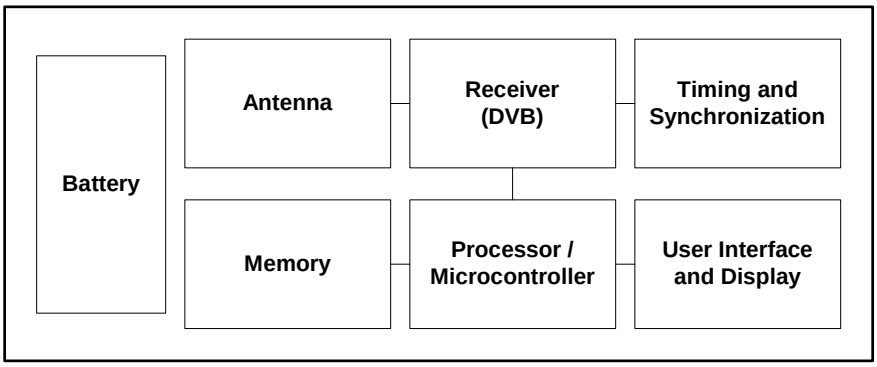

Figure 4: Terminal

9.2.2 Delta-t method (informative)

The basic goal of the **delta-t** method is to signal the time from the start of the currently received MPE (or MPE-FEC) section to the start of the next burst. To keep the delta-t insensitive to any constant delays within the transmission path, delta-t timing information is relative (e.g. "next burst will start 5 500 ms from the present time").

Delivering delta-t in MPE (or MPE-FEC) section removes the need to synchronize clocks between transmitter and Receiver. High flexibility is supported since parameters such as Burst Size, Burst Duration, Burst Bandwidth and Off-time may freely vary between elementary streams as well as between bursts within an elementary stream. The Receiver has to support sufficient accuracy for one Off-time only, as the clock is restarted by each burst.

Within the MPE section header, a 6-byte field is allocated for MAC address. The length of the used MAC address is signalled in the data_broadcast_descriptor inserted in the SDT or EIT. The minimum MAC address length is one byte, leaving up to five bytes for other use. Four of these five bytes are in the present document allocated for delivering Time Slicing and MPE-FEC real time parameters. This gives an additional benefit, as no bandwidth is required for delivering the parameters. Note that transmitting the above mentioned five bytes is mandatory regardless of whether they are used for MAC address or not.

In the case of multicast IP streams the MAC address is actually redundant data, as the MAC address is a function of the multicast group IP address. For all IP streams, the IP datagram header following immediately the MPE section header includes source and destination IP addresses uniquely identifying the IP stream. The Receiver can either ignore the MAC address entirely, filtering IP addresses only, or use the one byte MAC address to differentiate IP streams within the elementary stream. Even if hardware filtering within the demux is implemented on section level only, the IP layer would be able to filter any unused IP datagrams based on the IP addresses.

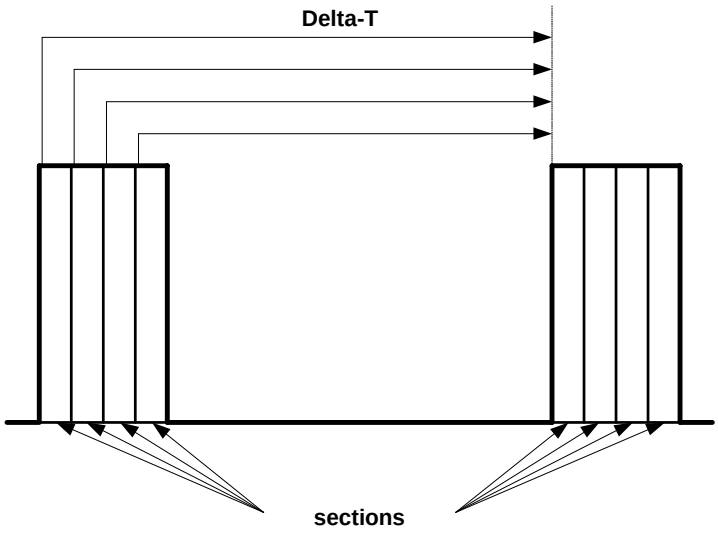

Figure 5: Each MPE section header contains delta-t indicating time to the beginning of the next burst

In bad reception conditions, parts of a burst may be lost. If the delta-t information were lost, the Receiver would not know for how long time it could go off and therefore it would be forced to stay on waiting for the next burst. To avoid these conditions, delta-t (together with other real time parameters) shall be delivered in the header of each section within a burst. Even in very bad reception conditions, if only one section is received, proper delta-t information can be accessed and power saving achieved.

As delta-t indicates the relative time rather than absolute one, the method is virtually unaffected by any constant delays within the transmission path. However, jitter on such delays does affect the accuracy of delta-t. This jitter is later referred to as **Delta-t Jitter**. If delta-t indicates the earliest possible time when the next burst may start, any Delta-t Jitter can be handled by decreasing the delta-t and therefore decreasing the accuracy of the delta-t. Note however that the accuracy of delta-t has an affect on the achieved power saving.

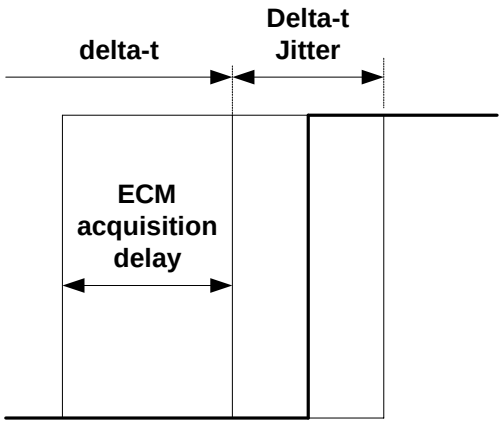

Figure 6: Delta-t Jitter

For Time Slicing, Delta-t Jitter of 10 ms can be accepted. This should be easily achieved as a typical transmission path already supports far better accuracy. On the other hand, virtually no gain is achieved by decreasing the value below 10 ms, as it already is less than a typical jitter in **Synchronization Time**.

Synchronization Time is the time required by a Receiver to switch on, lock into the signal and start the reception of sections. For example in DVB-T implementations the time is estimated to be about 200 ms to 250 ms, one limiting factor being approximately 100 ms required for synchronization of OFDM symbol reception. For services with CA, the time to acquire an ECM (i.e. the maximum repetition interval for the associated ECM stream in the vicinity of the data burst), together with the time taken to process the ECM to produce the Control Word needed for descrambling the start of the data burst also need to be considered. A terminal that buffers the data burst in the scrambled packet form may achieve a shorter synchronization time, as the Receiver would not need to be switched on for ECM processing ahead of the burst.

Two ways for achieving the ECM acquisition are possible, either the ECM repetition rate is increased, or the ECM acquisition delay is increased. With this implementation, ECMs are not protected through MPE-FEC, therefore, in very poor reception conditions, the receiver should increase this ECM reception delay in order to ensure it will get a valid ECM (e.g. to allow the reception of 2 to 4 ECMs).

Synchronization Time is implementation dependent, and typically differs noticeably from time to time (i.e. has noticeable jitter).

One can see how Delta-t Jitter has a similar effect to Synchronization Time. When the maximum Delta-t Jitter is known accurately, we may assume that on average each burst starts 1/2 × Delta-t Jitter later than the time indicated by delta-t. However, to be on the safe side, calculations later in the present document add 3/4 × Delta-t Jitter to the Synchronization Time.

9.2.3 Burst sizes and off-times (informative)

The size of a burst must be less than the memory available in a Receiver. When a burst is received, a Receiver has to buffer the data within its memory, to be consumed during the time between bursts. We may assume a Receiver can support 2 Mbits memory for buffering an incoming burst. Note that a Receiver supporting reception of multiple time sliced elementary streams simultaneously may need to support a 2 Mbits buffer for each time-sliced elementary stream, unless the elementary streams use smaller Burst Sizes. Supporting services with CA requires the receiver to receive at least two streams, the one containing the scrambled services, and the one receiving the EMMs, both transmitted in bursts over different PIDs. (ECMs are transmitted continuously, but the receiver does not need to listen to them all the time. The receiver just needs to power up early enough to get one ECM prior to a scrambled data burst.)

Burst Size refers to the number of Network Layer bits within a burst. Network Layer bits consist of section payload bits. Each MPE or MPE-FEC section contains 16 bytes overhead caused by the header and CRC-32. Assuming an average IP datagram size of 1 kB, this indicates 1,5 % overhead. In addition, the transport packet header causes overhead, which depends on the length of a section. If the length of a section is 1 kB, the overhead is approximately 2,2 %. The present document assumes 4 % overhead caused by section and transport packet headers.

Burst Bandwidth is an approximate momentary bandwidth used by a time sliced elementary stream while transmitting a burst. **Constant Bandwidth** is the average bandwidth required by the elementary stream when not time sliced. Both Burst and Constant Bandwidth include transmission of transport packets (188 bytes). For a Burst Size of 1 Mbits and Burst Bandwidth of 1 Mbps, the **Burst Duration** (time from the beginning to the end of the burst) is 1,04 s (due to the 4 % overhead).

Off-time is the time between bursts. During Off-time, no transport packets are delivered for the elementary stream of interest.

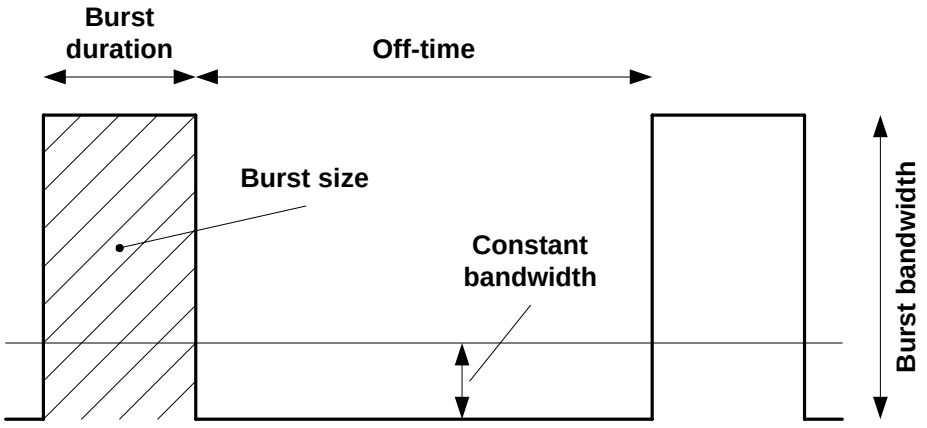

Figure 7: Burst parameters

Note that during the On-time (i.e. while a burst is transmitted), transport packets of other elementary streams may also be transmitted. This occurs when the Burst Bandwidth is less than the bandwidth of the transport stream (i.e. the burst uses only a part of the bandwidth available on the transport stream).

Maximum Burst Duration is the maximum duration of a burst, and shall be signalled for each time sliced elementary stream. A burst shall not start before T1 and shall end not later than at T2, where T1 is the time indicated by delta-t on the previous burst, and T2 is T1 + Maximum Burst Duration. In poor reception conditions, a Receiver may use this information to know when a burst has ended.

To enable a Receiver to reliably distinguish bursts from each other, the next burst shall not start before T2 of the current burst (i.e. delta-t shall signal time beyond T2). Distinction between bursts in a reliable way is required especially when MPE-FEC is used.

Note that this parameter can also be used to support Delta-t Jitter up to 5 s.

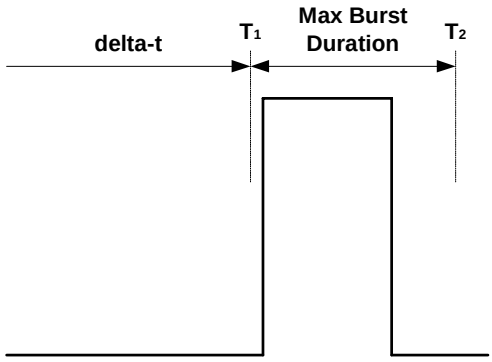

Figure 8: Maximum Burst Duration

Figure 9 shows simplified formulas to calculate the length of a burst, length of the off-time and achieved power saving. Correction factor 0,96 compensates for the overhead caused by transport packet and section headers. Note that the formulas are supported for an explanatory purpose only.

Bd	Burst Duration (seconds)
Bs	Burst Size (bits)
Bb	Burst Bandwidth (bits per second)
Cb	Constant Bandwidth (bits per second)
Ot	Off-time (seconds)
St	Synchronization Time (seconds)
Ps	Power Saving (per cent)
Dj	Delta-t Jitter (seconds)
CA	ECM synchronization time

$$Bd = \frac{Bs}{Bb \times 0{,}96}$$

$$Ot = \frac{Bs}{Cb \times 0{,}96} - Bd$$

$$Ps = \left(1 - \frac{(Bd + St + CA + (3/4 \times Dj)) \times Cb \times 0{,}96}{Bs}\right) \times 100\%$$

Figure 9: Formulas to calculate the length of a burst, off-time and the achieved saving on power consumption

If the Burst Size is 2 Mbits (over MPE and MPE-FEC section payloads) and the Burst Bandwidth is 15 Mbps (over related transport packets), the maximum Burst Duration is 140 ms (from the beginning of the first transport packet, to the end of the last one). If the elementary stream carries one streaming service at constant bandwidth of 350 kbps, and MPE-FEC is not supported, the average Off-time is 6,10 s. Assuming a Synchronization Time of 250 ms and a Delta-t Jitter of 10 ms, 93 % saving on power consumption may be achieved. Delta-t Jitter has only small effect on the power saving, as changing the value from 0 to 100 ms decreases the achieved power saving only from 94 % to 92 %.

9.2.4 Support for switching between transport streams (informative)

In the case of DVB transmission, Time Slicing enables one Receiver to monitor other transport streams without interrupting service reception. During the time between bursts, the Receiver may scan for other available signals, compare the signal strengths, and even switch between transport streams without interrupting the service reception.

Processing such tasks has an effect on achieved power saving, as the Receiver must remain on during the process. However, the effect may be kept at an acceptable level. For example for a DVB-T Receiver the required time for checking the signal strength on a single frequency is typically less than 20 ms. Using intelligent methods to anticipate available signals (i.e. neighbouring cells), a Receiver can significantly decrease the number of frequencies to check. Assuming the checking is accomplished once each cycle, the time required is only a fraction of the Off-time.

Careful synchronization may be implemented in the headend, so that the same service is transmitted on different slices in time in neighbouring cells. This would ensure seemingly uninterrupted (zero packet loss) reception when switching from one transport stream to another. Further consideration of burst synchronization is outside the scope of the present document.

9.2.5 Mixing Time Sliced elementary stream into a multiplex (informative)

Figure 10 illustrates a simplified construction of a headend for which the transmission is dedicated to IP services only.

The MPE Encapsulator is assumed to take responsibility for generating MPE sections from incoming IP datagrams, transport packet scrambling, if CA is used, as well as to add the required PSI/SI data and CA data, if required. Also, MPE-FEC Frames, when used, are generated in the MPE Encapsulator. The output stream of the MPE encapsulator is composed of MPEG-2 transport packets.

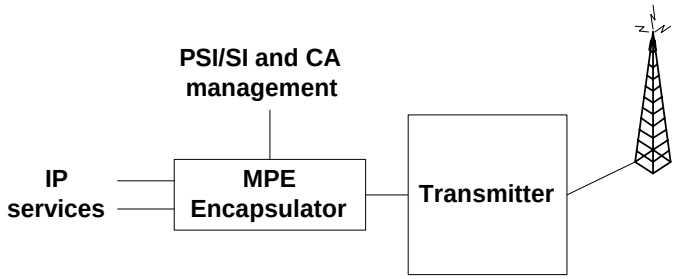

Figure 10: Headend construction for dedicated multiplex

As there are no other services (i.e. no non-time sliced services), the headend functionality remains simple. Time Slice bursts are generated in the MPE Encapsulator. A burst may use the maximum bandwidth. Any off period (time when no data bursts for any elementary stream are transmitted) may be filled with null packets. PSI/SI and ECM sections may be spread over the transport stream by allocating a constant bandwidth for them. EMMs are transmitted as a specific IP service to be sliced in the same way as other data. Note that fine tuned time slicing never leaves off periods, as there is always a burst of one elementary stream being transmitted.

Figure 11 illustrates the construction of a headend for a the transmitted multiplex containing both IP services and other services (e.g. digital TV). The major difference to the case of a dedicated multiplex is the requirement for a multiplexer. Note also that this is similar to the case where a transport stream containing Time Sliced elementary streams is remultiplexed. CA scrambling for the IP services may occur either in the MPE encapsulator, or in the multiplexer.

It is assumed that a constant bandwidth is allocated for all time sliced elementary streams. The rest of the transport stream bandwidth is available for non-time sliced elementary streams.

The process of multiplexing typically increases Delta-t Jitter. This has a negative affect on the accuracy of delta-t, therefore decreasing the achieved saving on power consumption. As noted before, a typical DVB-T transmission path including multiplexer(s) can guarantee jitter well under the required 10 ms. Therefore the usage of a multiplexer in general does not have significant effect on Time Slicing. However, it is important that the increase in Delta-t Jitter is taken into account in delta-t signalling.

Remultiplexing a Time Sliced transport stream generated for a terrestrial network may be considered a very unlikely business scenario. As Time Slicing is primarily aimed for terrestrial networks only, support for remultiplexing is not considered an important requirement for Time Slicing. In case remultiplexing increases the Delta-t Jitter significantly, bursts and the related signalling may need to be rebuilt.

Other services may set requirements on how the bandwidth is divided between elementary streams. E.g. PCR packets are recommended to appear in the transport stream every 40 ms. Since Burst Bandwidth is allowed to be less than the full bandwidth of the transport stream, this can easily be solved in number of ways. The MPE Encapsulator could e.g. support a fixed bandwidth of null packets, guaranteeing that the multiplexer always can add transport packets from other services, without a noticeable increase of jitter on any of the incoming streams.

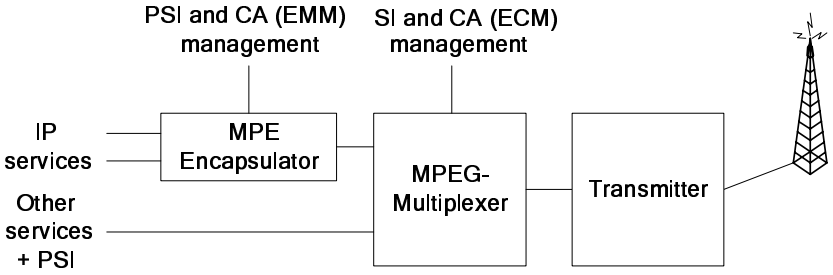

Figure 11: Headend construction for mixed multiplex

One possible way to avoid mixing time sliced and non-time sliced streams into a common multiplex - and to avoid usage of a multiplexer - is to use the hierarchical transmission mode. In his case the multiplex containing time sliced services is transmitted on high priority - ensuring better robustness in mobile environment - while the multiplex for non-time sliced services is transmitted on low priority - giving wider bandwidth for services intended for fixed reception. This effectively supports two multiplexes on a single transmission. A simplified construction of the headend supporting hierarchical transmission is illustrated in figure 12.

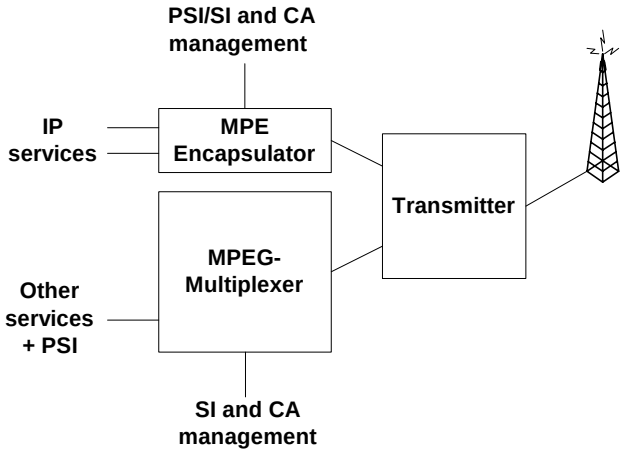

Figure 12: Headend construction for hierarchical transmission

9.2.6 Time Slicing and PSI/SI (informative)

PSI/SI are not time sliced. Existing PSI/SI does not support delivery of delta-t parameter within tables, and adding such support would not be compatible with existing implementations. In addition, a mobile handheld terminal does not require PSI/SI to be time sliced.

SI tables accessed by a mobile handheld convergence terminal are NIT and INT. Other tables are typically not required, as they carry no additional information for a terminal accessing services delivered via MPE. The content of the NIT is static by nature, so a terminal typically only accesses it when attaching a network. When changing from one transport stream to another, the terminal may need to read the content of INT, but not more than once. Changes in the INT can optionally be signalled in the PSI (PMT table), ensuring that constant filtering of the INT is not required.

The PAT and PMT tables of the PSI are re-transmitted at least once in every 100 ms. If the Burst Duration is longer than 100 ms, the terminal has access to all PSI tables while receiving the burst. For shorter bursts, the terminal may choose to keep the Receiver on until all required PAT and PMT tables are received. The CAT is static by nature, so a terminal supporting CA will need to read the content of the Conditional Access Table (CAT) only once after a change of transport stream.

9.2.7 Time Slicing and CA (informative)

Elementary streams using time slicing may be scrambled using the DVB Common Scrambling Algorithm. ECM streams are included in the Transport Stream in the normal manner, and associated with the service-carrying elementary streams via CA descriptors in the PMT, as for any other kind of DVB service.

A terminal receiving a scrambled stream will need to acquire an appropriate ECM before it can start to descramble a burst of a time-sliced elementary stream. The repetition interval between relevant ECMs in the vicinity of a data burst will have some effect on the power saving that can be achieved by the time slicing method. It is therefore recommended to use an ECM repetition interval for the associated ECM of no more than 100 ms in the vicinity of a data burst.

An EMM stream supporting services using time slicing should also use time slicing so that the power saving is also achieved for reception of EMMs. Such EMM data is therefore conveyed using MPE encapsulation and time slicing with or without MPE-FEC. The encapsulation for the Entitlement Management Message (EMM) data is specified in clause 9.8.

9.3 MPE-FEC

MPE-FEC is introduced to support reception in situations of high Packet Loss Ratio (PLR) on the MPE section level. Such high PLR may occur e.g. on mobile channels when the speed is too high and/or the C/N is too low. With the MPE-FEC about 25 % of TS data is allocated to parity overhead.

The MPE-FEC is introduced in such a way that MPE-FEC ignorant (but MPE capable) DVB Receiver will be able to receive the MPE stream in a fully backwards-compatible way, provided it does not reject stream_type defined in clause 9.6. This backwards compatibility holds both when the MPE-FEC is used with and without Time Slicing.

The use of MPE-FEC is not mandatory and is defined separately for each elementary stream in the transport stream. For each elementary stream it is possible to choose whether or not MPE-FEC is used, and if it is used, to choose the trade-off between FEC overhead and RF performance. Time critical services, without MPE-FEC and therefore minimal delay, could therefore be used together with less time critical services using MPE-FEC, on the same transport stream but on different elementary streams.

9.3.1 MPE-FEC frame

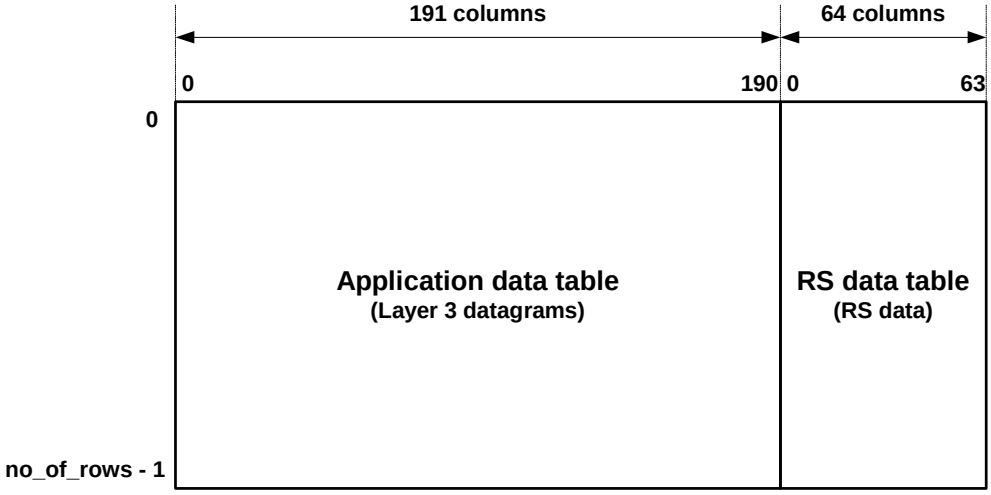

Figure 13: The structure of the MPE-FEC Frame

The MPE-FEC Frame is arranged as a matrix with 255 columns and a flexible number of rows (see figure 13). The number of rows is signalled in the time_slice_fec_identifier_descriptor. The maximum allowed value for this is 1 024, which makes the total MPE-FEC Frame size almost 2 Mbits. Each position in the matrix hosts an information byte. The left part of the MPE-FEC Frame, consisting of the 191 leftmost columns, is dedicated for OSI layer 3 (Network layer) datagrams (e.g. IP datagrams) and possible padding, and is called the *Application data table*. The right part of the MPE-FEC Frame, consisting of the 64 rightmost columns, is dedicated for the parity information of the FEC code and is called the *RS data table*. Each byte position in the Application data table has an address ranging from 0 to $191 \times$ no_of_rows - 1 (see figure 14). In the same way, each byte position in the RS data table has an address ranging from 0 to $64 \times$ no_of_rows - 1.

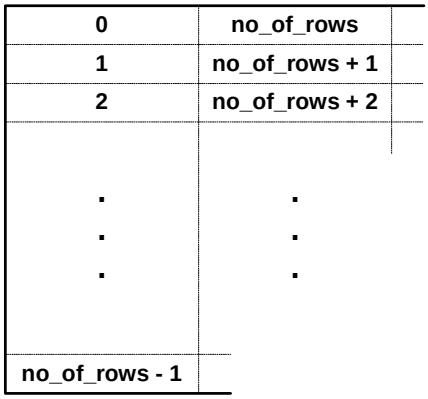

Figure 14: Addressing of byte positions in data tables

Layer 3 datagrams are introduced datagram-by-datagram, starting with the first byte of the first datagram in the upper left corner of the matrix and going downwards to the first column, see figure 15. The length of the datagrams may vary arbitrarily from datagram to datagram. Immediately after the end of one datagram the following datagram starts. If a datagram does not end precisely at the end of a column, it continues at the top of the following column. When all datagrams have entered the application data table any unfilled byte positions are padded with zero bytes, which makes the leftmost 191 columns completely filled.

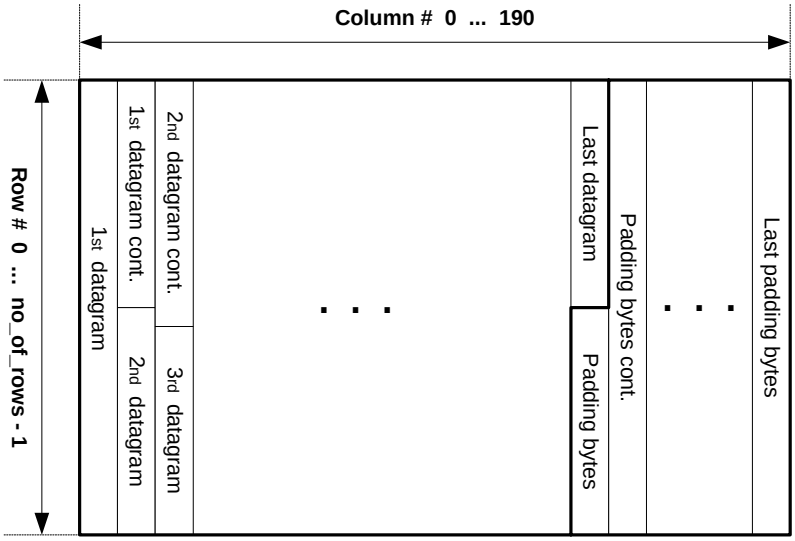

Figure 15: The layout of the Application data table

With all the leftmost 191 columns filled it is now possible, for each row, to calculate the 64 parity bytes from the 191 bytes of data and possible padding. The code used is Reed-Solomon RS(255,191,64). Each row then contains one RS codeword. Some of the rightmost columns of the RS data table may be discarded and hence not transmitted, to enable puncturing (see figure 16). The exact number of punctured RS columns does not need to be explicitly signalled and may change dynamically between frames. With this also the RS data table is completely filled and the MPE-FEC Frame is completed, see figure 16.

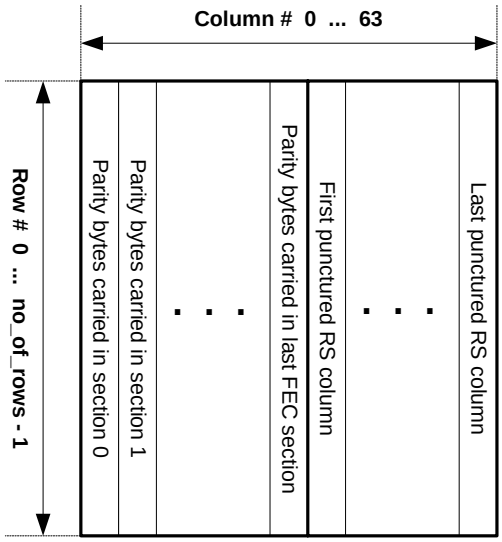

Figure 16: The layout of the RS data table

9.3.2 Carriage of MPE-FEC Frame

The datagrams are carried in MPE sections in compliance with DVB standard, irrespective of whether MPE-FEC is being used. This makes reception fully backwards compatible with MPE-FEC ignorant Receivers, provided they do not reject stream_type defined in clause 9.6. Each section carries in the section header a start address for the payload. This address indicates the byte position in the Application data table of the first byte of the section payload. In case the layer 3 datagram is divided over multiple MPE sections, each MPE section indicates the byte position in the Application data table of the first byte of the datagram fragment carried within the section. The Receiver will then be able to put the received datagram in the right byte positions in the Application data table and mark these positions as "reliable" for the RS decoder, provided the section CRC-32 check shows that the section is correct.

The last section of the Application data table contains a table_boundary flag, which indicates the end of the layer 3 datagrams within the Application data table. If all previous sections within the Application data table have been received correctly the Receiver does not need to receive any MPE-FEC sections and, if Time Slicing is used, the Receiver can be switched off without receiving and decoding RS data.

If MPE-FEC sections are also received, the number of padding columns (columns filled with padding bytes only) in the Application data table is indicated with 8 bits in the section header of the MPE-FEC sections - this value is only needed if RS decoding is performed.

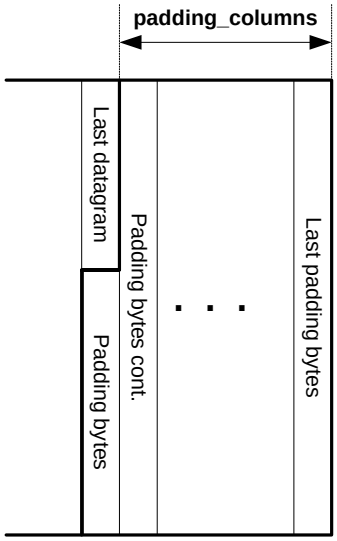

Figure 17: Padding columns

The parity bytes are carried in MPE-FEC sections. Each section carries exactly one column of RS data table. Punctured columns are not transmitted and not signalled explicitly.

9.3.3 RS decoding

The number of rows in the MPE-FEC Frame is signalled in the time_slice_fec_identifier_descriptor. The number of padding columns in the Application data table is signalled in the header of each MPE-FEC section. The number of punctured RS columns can be calculated as 63 - last_section_number, since last_section_number indicates the section number of the last section. As the section numbering is zero based, the indicated number is one less than the number of columns.

The Receiver introduces the number of padding bytes in the Application data table, as indicated in the MPE-FEC sections. It marks these padding bytes as reliable. If the Receiver has received the last section of the Application data table correctly, i.e. the table_boundary_flag, it can mark any remaining padding bytes in the Application data table as reliable. If the Receiver did not receive the last section correctly it will have to assume that all byte positions after the last correctly received section until the first padding column is lost data, even if some of it effectively is padding, and mark the corresponding byte positions as unreliable. Since the number of padding columns is signalled the maximum number of padding bytes that could be marked as unreliable is equal to no_of_rows-1. The effect on coding performance because of such unreliable padding bytes is marginal.

All MPE and MPE-FEC sections are protected by a CRC-32 code, which reliably detects all erroneous sections. For every correctly received section belonging to the Application data table or to the RS data table, the Receiver looks in the section header for the start address of the payload within the section and is then able to put the payload in the right position of the respective table.

After this procedure there are in general a number of remaining "holes", which corresponds to lost sections. All correctly received bytes, and Application data padding, can then be marked as "reliable" and all byte positions in the "holes", and in the punctured RS columns, can be marked as "unreliable" in the RS decoding.

All byte positions within the MPE-FEC Frame (Application data table + RS data table) are now marked as either "reliable" or "unreliable". With such reliability (erasure) information the RS decoder is able to correct twice the number of erroneous or unreliable bytes, which means the code can correct up to 64 such bytes per 255-byte codeword.

If there are more than 64 unreliable byte positions in a row the RS decoder will *not* be able to correct anything and will therefore typically just output the byte errors without error correction. The Receiver will therefore have perfect knowledge about the positions of any remaining byte errors within the MPE-FEC Frame after RS decoding. If a datagram is only partly corrected the receiver will be able to detect this and (optionally) discard this datagram.

In addition to the CRC-32, which detects erroneous sections, the RS decoder also very reliably detects erroneous TS packets. If the MPEG-2 demultiplexer discards erroneous packets it could be designed not to build sections, which contain lost TS packets. In this way almost only correct sections would be built and the role of the CRC-32 would be to provide *additional* error detection functionality, which normally is not needed. In very rare cases it could happen that the RS decoder fails to detect an erroneous TS packet, which also happens to have the right PID, and that an erroneous section therefore could be constructed. In these cases the CRC-32 would discover such a section error.

9.3.3.1 Application data padding columns - Code shortening

By introducing a certain number of zero-valued Application data padding columns in the rightmost part of the Application data table, it is possible to make the code *stronger*. These padding columns are used only for calculation of parity bytes but not transmitted. In the Receiver they are reintroduced and marked as "reliable" for the RS decoder. With e.g. 127 padding columns, there are 64 columns left for datagrams. With the 64 parity columns the effective code rate of the code becomes 1/2. However, the price for this is that the effective codeword length is decreased by roughly 50 %. The number of Application data padding columns is signalled with 8 bits in the header of MPE-FEC section, and may be different for consecutive MPE-FEC frames. The allowed range is 0 to 190.

9.3.3.2 Discarding RS data columns - Puncturing

Effectively *weaker* code rates than that of the mother code may be achieved by *puncturing*. Puncturing is performed by discarding one or more of the last RS data columns. The number of discarded (punctured) RS columns may vary dynamically between MPE-FEC Frames within the range 0 to 63 and can be calculated as 63 - last_section_number. A special case is when no RS columns are transmitted, i.e. puncturing is 64 columns. Puncturing will decrease the overhead introduced by the RS data and thus decrease the needed bandwidth. The drawback of puncturing is an effectively weaker code rate.

9.4 The Buffer Model for the Receiver (informative)

Figure 18 illustrates a simplified model for the Time Slicing/MPE-FEC buffer which is used in the Receiver to store the time slicing burst and to offer constant bit stream for streaming services during off time. The data is received at the rate of B_b and the leakage rate, i.e. the rate at the output of the buffer, is R_{out}. The buffer has a certain size and when the data is written into the buffer, there is a certain processing delay (including, e.g. MPE-FEC decoding time) before the data can be read out.

For each elementary stream, the maximum average bit rate over one time slicing cycle denoted by C_b is signalled in the time_slice_fec_identifier descriptor, and it is defined as:

$$C_b = \frac{B_s}{B_d + O_t} = \frac{B_s}{T_c},$$

where B_s and B_d are the size and the duration of the burst, respectively, O_t is the off-time between the bursts, and T_c is the cycle time for the burst (see figure 19). All parameters are here defined with respect to layer 3 datagrams.

By knowing C_b and its own processing time, the Receiver can, for instance, check if the leakage rate R_{out} is high enough to successfully receive the particular elementary stream.

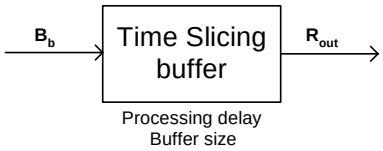

Figure 18: The Time Slicing/MPE-FEC buffer in the Receiver

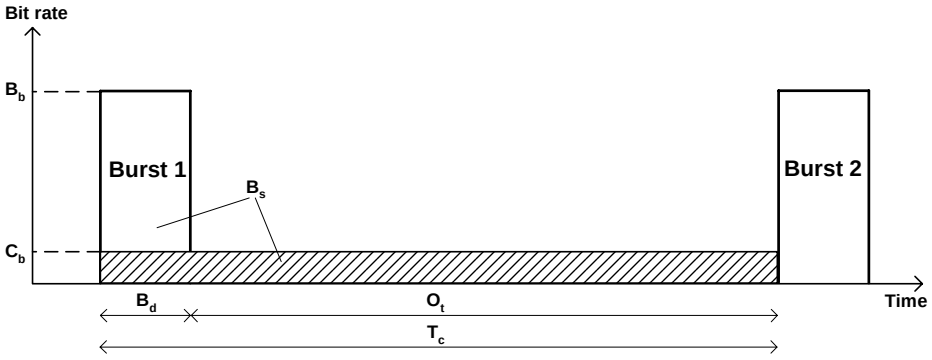

Figure 19: The average bit rate over one time slicing cycle Cb

9.5 Time Slice and FEC identifier descriptor

This descriptor identifies whether Time Slicing or MPE-FEC are used on an elementary stream.

When this descriptor specifies an elementary stream other than an EMM stream being Time Sliced and/or MPE-FEC being used, the data_broadcast_descriptor shall appear in the SDT for the service. In the latter descriptor, data_broadcast_id shall be set to 0x0005, and MAC_address_range to value of 0x01 or 0x02, indicating that the MAC_address_1…4 bytes are not used to differentiate Receivers within the elementary stream.

This descriptor is defined in following locations:

Network Information Table (NIT):

- When located in the first descriptor loop, the descriptor applies to all elementary streams with stream_type 0x90 within all transport streams announced within the sub-table.
- When located in the second descriptor loop, the descriptor applies to all elementary streams with stream_type 0x90 within the specific transport stream. This descriptor overwrites any descriptors in the first descriptor loop.

IP/MAC Notification Table (INT):

- When located in the platform_descriptor_loop, the descriptor applies to all elementary streams referenced within the sub-table. This descriptor overwrites any descriptors in the NIT.
- When located in the operational_descriptor_loop, the descriptor applies to all elementary streams referenced within the operational_descriptor_loop following the appearance of the descriptor. This descriptor overwrites any descriptors in the platform descriptor loop and in the NIT. If an elementary stream is referenced from multiple locations within an INT, each shall contain the same signalling.

Conditional Access Table (CAT):

- When located in the descriptor_loop, the descriptor applies to all EMM streams referenced in the table that use MPE or MPE-FEC. The use of MPE or MPE-FEC for an EMM stream can be determined from the table_id values of the elementary stream sections. A terminal may also know from CA_system_ID values, private data in the CA_descriptor, or other CA system-specific means whether MPE/MPE-FEC is used for an EMM stream. The CAT is defined in the MPEG-2 specification (ISO/IEC 13818-1) [1].

Table 37: Time Slice and FEC identifier descriptor

Syntax	Number of bits	Identifier
time_slice_fec_identifier_descriptor () {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
time_slicing	1	bslbf
mpe_fec	2	uimsbf
reserved_for_future_use	2	bslbf
frame_size	3	uimsbf
max_burst_duration	8	uimsbf
max_average_rate	4	uimsbf
time_slice_fec_id	4	uimsbf
for(i=0; i<N; i++) {		
id_selector_byte	8	bslbf
}		
}		

Semantics for Time Slice and FEC identifier descriptor

descriptor_tag: Shall be set to value of 0x77.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

time_slicing: This 1-bit field indicates, whether the referenced elementary stream is Time Sliced. The value "1" indicates Time Slicing being used, and the value "0" indicates that Time Slicing is not used.

mpe_fec: This 2-bit field indicates, whether the referenced elementary stream uses MPE-FEC, and which algorithm is used. Coding is according to table 38.

Table 38: MPE-FEC algorithm

value	MPE-FEC	algorithm
00	MPE-FEC not used	n/a
01	MPE-FEC used	Reed-Solomon(255, 191, 64)
10 to 11	reserved for future use	reserved for future use

reserved_for_future_use: This 2-bit field shall be set, when not used, to "11".

frame_size: This 3-bit field is used to give information that a decoder may use to adapt its buffering usage. The exact interpretation depends on whether Time Slicing and/or MPE-FEC are used.

In case Time Slicing is used (i.e. time_slicing is set to "1"), this field indicates the maximum number of bits on section payloads allowed within a Time Slice burst on the elementary stream. For MPE sections, bits are counted over ip_datagram_data_bytes or LLC_SNAP field (whichever is supported), excluding any possible stuffing_bytes. For MPE-FEC sections, bits are counted over rs_data_bytes.

When MPE-FEC is used (i.e. mpe_fec is set to 0x1), this field indicates the exact number of rows on each MPE-FEC Frame on the elementary stream.

If both Time Slicing and MPE-FEC are used on an elementary stream, both constraints (i.e. the maximum burst size and the number of rows) apply.

If time_slice_fec_id is set to "0", the coding of the frame_size is according to table 39. If time_slice_fec_id is set to any other value, coding of the frame_size is currently not defined.

Table 39: Size coding

Size	Max Burst Size	MPE-FEC Frame rows
0x00	512 kbits = 524 288 bits	256
0x01	1 024 kbits	512
0x02	1 536 kbits	768
0x03	2 048 kbits	1 024
0x04 to 0x07	reserved for future use	reserved for future use

max_burst_duration: This 8-bit field is used to indicate the maximum burst duration in the elementary stream. A burst shall not start before T1 and shall end not later than at T2, where T1 is the time indicated by delta-t in the previous burst, and T2 is T1 + maximum burst duration.

If the time_slice_fec_id is set to "0", the indicated value for maximum burst duration shall be from 20 ms to 5,12 s, the resolution is 20 ms, and the field is decoded according to the following formula:

$$\text{Maximum burst duration} = (\text{max\_burst\_duration} + 1) \times 20 \text{ ms}$$

If the time_slice_fec_id is set to any other value than "0", the coding of the max_burst_duration is currently not defined.

When time_slicing is set to '0' (i.e. Time Slicing not used), this field is reserved for future use and shall be set to 0xFF when not used.

max_average_rate: This 4-bit field is used to define the maximum average bit rate in MPE section payload level over one time slicing cycle or MPE-FEC cycle and it is given by:

$$C_b = \frac{B_s}{T_c},$$

where B_s is the size of the current Time Slicing burst or MPE-FEC Frame in MPE section payload bits and T_c is the time from the transport packet carrying the first byte of the first MPE section in the current burst/frame to the transport packet carrying the first byte of the first MPE section in the next burst/frame within the same elementary stream.

Note that when MPE-FEC is used, the RS data is not included in B_s.

If time_slice_fec_id is set to "0", the coding of the max_average_rate is according to table 40. If time_slice_fec_id is set to any other value, coding of the max_average_rate is currently not defined.

Table 40: Coding for max_average_rate

Bit rate	Description
0000	16 kbps
0001	32 kbps
0010	64 kbps
0011	128 kbps
0100	256 kbps
0101	512 kbps
0110	1 024 kbps
0111	2 048 kbps
1000 to 1111	reserved for future use

time_slice_fec_id: This 4-bit field identifies the usage of following id_selector_byte(s). Currently no use is defined, and this field shall be set to value 0x0, and id_selector_byte(s) shall not be present. Note that this field affects on coding of frame_size, max_burst_duration and max_average_rate fields on the actual descriptor, and the address field of real-time parameters on the referred elementary stream.

id_selector_byte: The definition of the id_selector_byte(s) of the time_slice_fec_identifier_descriptor will depend on the time_slice_fec_id.

9.5.1 Definition of Reed-Solomon RS(255,191,64) code

Reed-Solomon RS (255,191, t = 32) code shall be applied to the set of 191 bytes of each Application data table row to generate a Reed-Solomon codeword. Any byte position, which does not contain datagram bytes, shall be padded with zero bytes before the RS calculation.

NOTE: The Reed-Solomon code has length 255 bytes, dimension 191 bytes and allows correcting up to 32 random erroneous bytes in a received word of 255 bytes. When reliable erasure information is used, such as provided by the CRC32 of the MPE and/or MPE-FEC sections, the code allows correcting up to 64 random erroneous bytes.

Code Generator Polynomial: $g(x) = (x+\lambda^0)(x+\lambda^1)(x+\lambda^2)...(x+\lambda^{63})$, where $\lambda = 02_{HEX}$

Field Generator Polynomial: $p(x) = x^8 + x^4 + x^3 + x^2 + 1$

9.6 Carriage of application data

The following applies to each elementary stream for which the time_slice_fec_identifier_descriptor indicates that Time Slicing is used:

- Each burst shall only contain complete sections (i.e. sections shall not be fragmented between bursts). Each burst shall only contain complete datagrams (i.e. datagrams shall not be fragmented between bursts). Each burst shall contain at least one MPE section.

The following applies on each elementary stream for which the time_slice_fec_identifier_descriptor indicates that MPE-FEC is used:

- Each MPE-FEC Frame shall only contain complete datagrams (i.e. datagrams shall not be fragmented between MPE-FEC Frames).

For each MPE-FEC Frame, at least one MPE section shall be delivered.

The section_syntax_indicator on each MPE section on the elementary stream shall be set to '1', indicating that CRC_32 is used at the end of the section.

The following applies to each elementary stream for which the time_slice_fec_identifier_descriptor indicates that Time Slicing and/or MPE-FEC is used:

- The datagrams of delivered services within the elementary stream shall be transmitted in the payload of MPE sections. See clause 7.1 for more details on how datagrams are encapsulated into MPE sections.
- Bytes MAC_address_1 … MAC_address_4 in each MPE section delivered on the elementary stream shall carry valid real time parameters as defined in clause 9.8.
- The elementary stream shall have stream_type 0x90. Note that other stream_types may be defined later.
- Each delivered MPE section shall contain a complete or a fragment of a valid datagram containing network layer address. MPE sections shall not be empty (i.e. MPE sections shall always contain the data of a datagram). In case of IP datagrams, each delivered MPE section shall contain a complete valid datagram containing valid IP headers. An MPE section shall not contain the data of multiple datagrams.

9.7 Carriage of ECMs for time-sliced services

In order to allow the receiver to get one or more ECM before the burst is received, the receiver has to wake up a little earlier. This delay is calculated in order to allow the receiver to receive one or more ECMs (depending on the reception conditions) before the data burst itself. The delay is calculated by knowing the ECM repetition rate which is transmitted in a specific descriptor, the **ECM_repetition_rate_descriptor** that may be inserted in the PMT after the CA_descriptor. In the absence of this descriptor, the receiver shall assume a default interval between ECMs of 100 ms.

The syntax of the ECM_repetition_rate descriptor is as follows:

Table 40a: ECM repetition rate descriptor

Syntax	Number of bits	Identifier
ECM repetition rate_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
CA_system_ID	16	uimsbf
ECM repetition rate	16	uimsbf
for (i=0; i<N; i++) {		
private_data_byte	8	uimsbf
}		
}		

Semantic definition of fields in ECM repetition rate descriptor

descriptor_tag: Shall be set to value of 0x78.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

CA_system_ID: This 16-bit field specifies the CA system in accordance with TR 101 162 [3].

ECM repetition rate: This parameter gives the maximum delay between two consecutive ECMs of the same ECM elementary stream expressed in milliseconds.

9.8 Carriage of EMMs for time-sliced services

EMMs are carried in sliced mode in MPE sections, optionally using MPE-FEC. The EMMs are carried as the payload of UDP/IP datagrams, with a maximum payload length of 1 472 bytes. The processing of the datagram payload is specific to the CA system.

9.9 Carriage of RS data

When time_slice_fec_identifier_descriptor specifies MPE-FEC is used on an elementary stream, the RS data of each MPE-FEC Frame shall be delivered in MPE-FEC sections described in table 41. MPE-FEC sections are carried in the same elementary stream with the corresponding Application data.

Each MPE-FEC section shall carry exactly one column of the corresponding RS table. The length of a column is indicated in field frame_size of the corresponding time_slice_fec_identifier_descriptor.

The number of MPE-FEC sections used to carry RS data of an MPE-FEC Frame shall not exceed the number of columns of the RS table. However, the number of MPE-FEC sections delivered may be less, indicating that not all RS data is transmitted. In the latter case, the RS-decoder shall consider bytes on columns not delivered as unreliable. The number of delivered MPE-FEC sections is indicated using last_section_number.

The position of the delivered RS data in the RS table is indicated by the section_number field. Section 0 carries the first (left most) column of the RS table, section 1 carries the second column, and so on. The columns not delivered shall be the rightmost columns of an RS table.

MPE-FEC sections are compliant to the DSMCC_section Type "User private" (see ISO/IEC 13818-6 [5]). The mapping of the section into MPEG-2 Transport Stream packets is defined in MPEG-2 Systems ISO/IEC 13818-1 [1].

The syntax and semantics of the MPE-FEC_section are defined in table 41.

Table 41: MPE-FEC section

Syntax	Number of bits	Identifier
MPE-FEC_section () {		
table_id	8	uimsbf
section_syntax_indicator	1	bslbf
private_indicator	1	bslbf
reserved	2	bslbf
section_length	12	uimsbf
padding_columns	8	uimsbf
reserved_for_future_use	8	bslbf
reserved	2	bslbf
reserved_for_future_use	5	bslbf
current_next_indicator	1	bslbf
section_number	8	uimsbf
last_section_number	8	uimsbf
real_time_parameters()		
for(i=0; i<N; i++) {		
rs_data_byte	8	uimsbf
}		
CRC_32	32	uimsbf
}		

The semantics for the MPE-FEC section:

table_id: Shall be set to value of 0x78.

section_syntax_indicator: This field shall be set to 1 and be interpreted as defined by ISO/IEC 13818-6 [5].

private_indicator: This field shall be set to 0 and be interpreted as defined by ISO/IEC 13818-6 [5].

reserved: Shall be set to '11'.

section_length: Specifies the number of remaining bytes in the section immediately following this field up to the end of the section, including the CRC.

The number of rs_data_bytes carried in the section shall be exactly the number of rows in the corresponding MPE-FEC Frame, as indicated in a corresponding time_slice_fec_identifier_descriptor.

padding_columns: This 8-bit field indicates the number of full columns of the Application data table of the actual MPE-FEC Frame filled with padding bytes only. The value indicated shall be from 0 to 190. Note that the value may vary frame by frame.

reserved_for_future_use: These eight bits shall be set, when not used, to '11111111'.

reserved: Shall be set to '11'.

reserved_for_future_use: This 5-bit field shall be set, when not used, to '11111'.

current_next_indicator: Shall be set a value of '1'.

section_number: This 8-bit field gives the number of the section. The section_number of the first section carrying RS data of an MPE-FEC Frame shall be '0x00'. The section_number shall be incremented by 1 with each additional section carrying RS data of the concerned MPE-FEC Frame.

last_section_number: Shall indicate the number of the last section that is used to carry the RS data of the current MPE-FEC Frame.

The number of sections shall not exceed the number of columns in the corresponding RS table.

rs_data_byte: Contain the RS data delivered.

CRC_32: This field shall be set as defined by ISO/IEC 13818-6 [5]. It is calculated over the entire MPE-FEC_section.

9.10 Real time parameters

On an elementary stream where Time Slicing and/or MPE-FEC are used, each MPE section and MPE-FEC section shall carry real time parameters described in table 42. For the MPE sections, real time parameters are carried within the MAC_address_4…MAC_address_1, as illustrated in figure 20.

Table 42: Time Slicing and MPE-FEC real time parameters

Syntax	Number of bits	Identifier
`real_time_parameters () {`		
`delta_t`	12	uimsbf
`table_boundary`	1	bslbf
`frame_boundary`	1	bslbf
`address`	18	uimsbf
`}`		

delta_t: Usage of this 12-bit field depends on whether Time Slicing is used on the elementary stream.

The following applies when Time Slicing is used (regardless whether MPE-FEC is used or not):

- The field indicates the time (delta-t) to the next Time Slice burst within the elementary stream. The time information is in all MPE sections and MPE-FEC sections within a burst and the value may differ section by section. The resolution of the delta-t is 10 ms. Value 0x00 is reserved to indicate that no more bursts will be transmitted within the elementary stream (e.g. end of service). In such a case, all MPE sections and MPE-FEC sections within the burst shall have the same value in this field.

EXAMPLE: delta-t value 0xC00 = 3072 indicates the time to the next burst is 30,72 s.

- Delta-t information is the time from the transport packet carrying the first byte of the current MPE section or MPE-FEC section to the transport packet carrying the first byte of next burst. Therefore the delta-t information may differ between MPE sections and between MPE-FEC sections within a burst.

- The time indicated by delta-t shall be beyond the end of the maximum burst duration of the actual burst. This ensures a decoder can always reliably distinguish two sequential bursts within an elementary stream.

The following applies when only MPE-FEC is used (i.e. Time Slicing is not used):

- The field supports a cyclic MPE-FEC Frame index within the elementary stream. The value of the field increases by one for each subsequent MPE-FEC Frame. After value '111111111111', the field restarts from '000000000000'.

- In case of large portions of lost data this parameter makes it possible to identify to which MPE-FEC Frame the actual received section belongs.

table_boundary: This 1-bit flag, when set to '1', indicates that the current section is the last section of a table within the current MPE-FEC Frame. If the section is an MPE section, this flag indicates the last section of Application data table. A decoder not supporting MPE-FEC may ignore all subsequent sections until the end of the MPE-FEC Frame (indicated with frame_boundary).

For each MPE-FEC Frame exactly one MPE section with this flag set shall be transmitted. For each MPE-FEC Frame for which any RS data is transmitted exactly one MPE-FEC section with this flag set shall be transmitted.

When MPE-FEC is not used on the elementary stream, this flag is reserved for future use, and shall be set to '1' when not used.

frame_boundary: This 1-bit flag, when set to '1', indicates that the current section is the last section within the current burst (Time Slicing supported) and MPE-FEC Frame (MPE-FEC supported).

For each Time Slice burst exactly one section with this flag set shall be transmitted. For each MPE-FEC Frame exactly one section with this flag set shall be transmitted.

Note that for each MPE-FEC Frame, MPE sections are delivered before MPE-FEC sections. Therefore, if frame_boundary is set in an MPE section, MPE-FEC sections shall not be delivered for the frame.

On an elementary stream where Time Slicing is used, a Receiver may ignore any sections after the section with the frame_boundary set, until the time indicated by the delta-t.

address: This 18-bit field specifies the byte position in the corresponding MPE-FEC Frame table for the first byte of the payload carried within the section. All sections delivering data for any MPE-FEC Frame table shall be delivered in ascending order according to the value of this field.

If the time_slice_fec_id in the time_slice_fec_identifier_descriptor associated with the elementary stream is set to '0', the coding of the address is the following.

The bytes position is a zero-based linear address within an MPE-FEC Frame table, starting from the first row of the first column, and increasing towards the end of the column. At the end of the column, the next byte position is at the first row of the next column.

The first section carrying data of a given MPE-FEC Frame shall be the MPE section carrying the Application data datagram at address '0'. All sections carrying Application data datagrams of a given MPE-FEC Frame shall be transmitted prior to the first section carrying RS-data of the MPE-FEC Frame (i.e. sections carrying Application data datagrams shall not be interleaved with sections carrying RS-data within a single MPE-FEC frame). Within an elementary stream, all sections carried between the first and the last section of an MPE-FEC Frame shall carry the data belonging to the MPE-FEC Frame (i.e. only MPE sections carrying datagrams and MPE-FEC sections carrying RS-data are allowed). Within an elementary stream, sections delivering data of different MPE-FEC Frames shall not be interleaved.

The section following the last section carrying Application data datagram on an MPE-FEC Frame, shall contain either the first section carrying the RS-data of the same MPE-FEC Frame, or the first Application data section of the next MPE-FEC Frame. In the latter case, RS-data of the first MPE-FEC Frame is not transmitted.

For each MPE-FEC Frame, exactly one MPE section shall be transmitted with address field set to value '0'. For each MPE-FEC Frame for which any RS data is transmitted, exactly one MPE-FEC section shall be transmitted with address field set to value '0'. Padding shall not exist between delivered Application data in the Application data table. Datagrams shall not overlap in an Application data table. Padding shall not exist between delivered RS data in the RS table.

If the time_slice_fec_id in the time_slice_fec_identifier_descriptor associated with the elementary stream is set to any other value than "0", the coding of the address is currently not defined.

NOTE 1: The addressing starts from zero within each MPE-FEC Frame table.

NOTE 2: When both Time Slicing and MPE-FEC are used on an elementary stream, each burst on the elementary stream shall contain exactly one MPE-FEC Frame (i.e. MPE-FEC Frame shall not be split over multiple bursts).

If MPE-FEC is not used on the elementary stream, this field is reserved for future use, and shall be set to value of 0x3FFFF (all bits set) when not used.

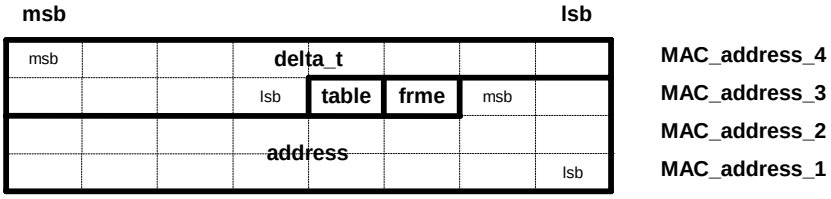

Figure 20: Mapping of real time parameters in MAC_address field

10 Data carousels

10.1 Data transport specification

The specification of DVB data carousels is based on the DSM-CC data carousel specification (ISO/IEC 13818-6 [5]). The DSM-CC data carousel specification embodies the cyclic transmission of data to receivers. The data transmitted within the data carousel is organized in "modules" which are divided into "blocks". All blocks of all modules within the data carousel are of the same size, except for the last block of each module which may be of a smaller size. Modules are a delineation of logically separate groups of data within the data carousel. Modules can be clustered into a group of modules if required by the service. Likewise, groups can in turn be clustered into SuperGroups.

The data carousel specification uses four messages of the DSM-CC download specification. The data is carried in DownloadDataBlock messages, while the control over the modules is provided by DownloadInfoIndication, DownloadServerInitiate, and DownloadCancel messages. The Download-ServerInitiate message describes the groups in a SuperGroup, while the DownloadInfoIndication message describes the modules in a group. Based on the control messages, the receivers may acquire a subset of the modules from the network. The syntax and semantics of these messages are defined in (see ISO/IEC 13818-6 [5]).

The use of these messages in DVB data carousels is described in the present document.

10.1.1 Structure of DVB data carousel

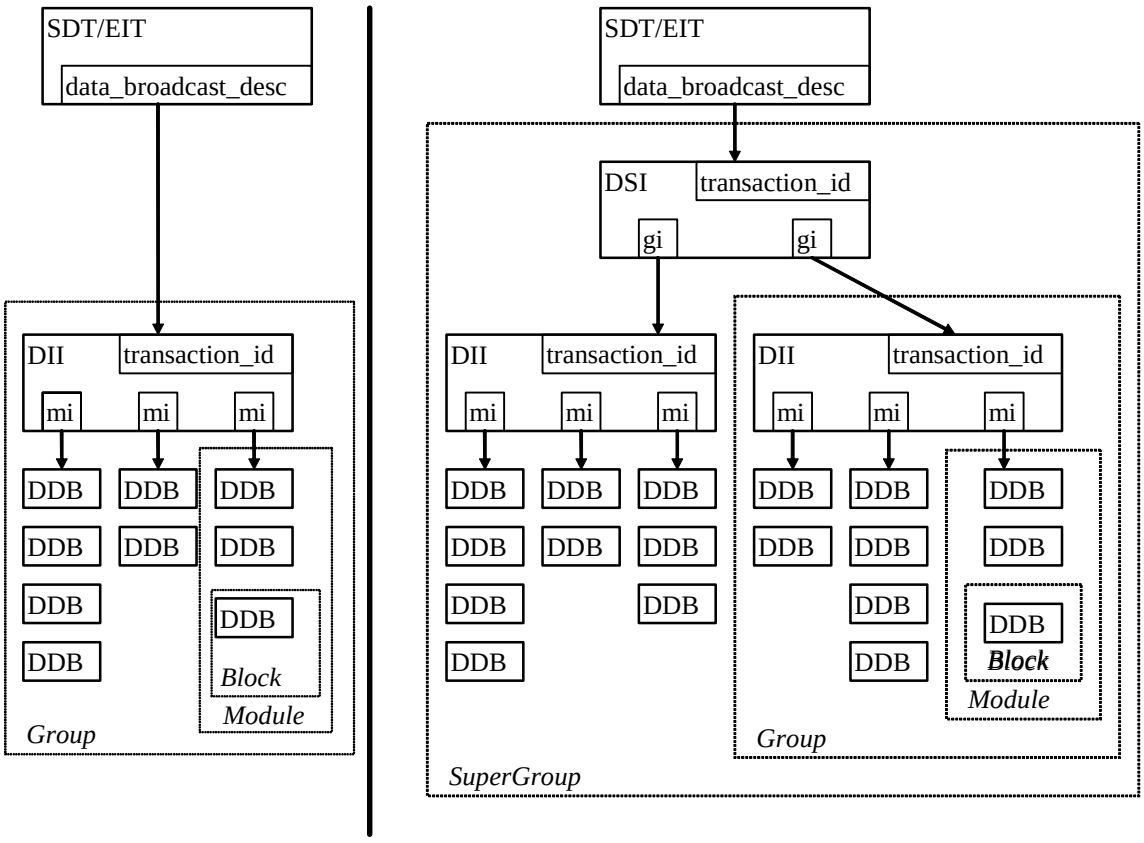

Figure 21: Structure of the DVB data carousel

DVB data carousels can have one or two layers of control information as illustrated in figure 4. The simplest form of DVB data carousels is a data carousel with one control layer which describes a single group. In this case, the SDT/EIT tables contain a data_broadcast_descriptor that points to a DownloadInfoIndication message. This message describes the modules in the data carousel using the ModuleInfoByte field. This field contains of a loop of descriptors that may contain miscellaneous information, e.g. a pointer to the location of the DownloadDataBlock messages.

If two layers of clustering are required, a DownloadServerInitiate message is used to describe the different groups in the SuperGroup. The DownloadInfoIndication message is used in the same way as with the one-layer data carousel. The DownloadServerInitiate message describes the groups with the GroupInfoByte field and allows for platform differentiation. The GroupInfoByte field consist also of a loop of descriptors that may contain miscellaneous information.

The decoder should be able to work with both types of carousels. The service provider can choose which type of carousel to use.

Groups and modules can be transmitted on dedicated PIDs and/or shared PIDs. If no explicit location references are given, the location is inherited from the control message. Each arrow in figure 21 represents the access information that is required to acquire the message(s) to which the arrow points. Within DVB data carousels this information consists of:

1) a component tag, i.e. a pointer to a particular stream in the service; and

2) a transaction/module identifier, i.e. a unique identifier of a control message or a module.

Receivers can use these values to filter the messages from the stream efficiently.

Furthermore in the DownloadServerInitiate and DownloadInfoIndication messages parameters for the sizes of modules and blocks have been specified, based on DSM-CC.

Within the present document the use of the compatibilityDescriptor() as specified by DSM-CC has been limited to a forward reference mechanism from the DownloadServerInitiate message to DownloadInfoIndication messages.

The compatibilityDescriptor of the DSI message is located in the GroupInfoIndication field and is called groupCompatibility(). All DownloadDataBlock and DownloadInfoIndication messages within a SuperGroup (in the case of a two layer data carousel) or a group (in the case of a single layer data carousel) have the same downloadId. This implies that groups can share modules because all ModuleId's are unique within the scope of the downloadId.

Each control message has a transaction_id which is the unique identifier of the message. Transaction_id and module_id can be used to efficiently filter the data of the data carousel, based on the following semantics:

- **For the two-layer carousel:**
  - For DownloadServerInitiate messages the 2 least significant bytes of the transactionId shall be in the range 0x0000 to 0x0001.
  - For DownloadInfoIndication messages the 2 least significant bytes of the transactionId shall be in the range 0x0002 to 0xFFFF.
- **For the one-layer carousel:**
  - For DownloadInfoIndication messages the 2 least significant bytes of the transactionId shall be in the range 0x0000 to 0x0001.

10.1.2 DownloadServerInitiate message

The DownloadServerInitiate message is used to build a SuperGroup. The semantics for DVB data carousels are as follows:

serverId: This field shall be set to 20 bytes with the value of 0xFF.

compatibilityDescriptor(): This structure shall only contain the compatibilityDescriptorLength field of the compatibilityDescriptor() as defined in DSM-CC (see ISO/IEC 13818-6 [5]). It shall be set to the value of 0x0000. The privateDataByte fields shall contain the GroupInfoIndication structure as defined in table 43.

privateDataLength: This field defines the length in bytes of the following GroupInfoIndication structure.

privateDataByte: These fields shall convey the GroupInfoIndication structure as defined in table 43.

Table 43: GroupInfoIndication structure

Syntax	Number of bytes
GroupInfoIndication() {	
NumberOfGroups	2
for(i=0;i< numberOfGroups;i++) {	
GroupId	4
GroupSize	4
GroupCompatibility()	
GroupInfoLength	2

for(i=0;i<N;I++) {
groupInfoByte
}
	1

}	
PrivateDataLength	2

for(i=0;i< privateDataLength;i++) {
privateDataByte
}
	1

}	

Semantics of the GroupInfoIndication structure:

numberOfGroups: This is a 16-bit field that indicates the number of groups described in the loop following this field.

groupId: This is a 32-bit field which shall be equal to transactionId of the DownloadInfoIndication message that describes the group.

groupSize: This is a 32-bit field that shall indicate the cumulative size in bytes of all the modules in the group.

groupCompatibility: The GroupCompatibility structure is equal to the CompatibilityDescriptor structure of DSM-CC (see ISO/IEC 13818-6 [5]).

groupInfoLength: This is a 16-bit field indicating the length in bytes of the descriptor loop to follow.

groupInfoByte: These fields shall convey a list of descriptors which each define one or more attributes. The descriptors included in the loop shall describe the characteristics of the group.

privateDataLength: This field defines the length in bytes of the following privateDataByte fields.

privateDataByte: These fields are user defined.

10.1.3 DownloadInfoIndication message

The DownloadInfoIndication message contains the description of the modules within a group as well as some general parameters of the data carousel (such as downloadId and blockSize). Each module is described by a number of attributes. The attributes moduleId, moduleSize, and moduleVersion are defined as fields in the DownloadInfoIndication message by DSM-CC (see ISO/IEC 13818-6 [5]). Other module attributes shall be carried as descriptors as defined below. The moduleId range of 0xFFF0-0xFFFF is reserved for DAVIC compliant applications. The semantics of the DownloadInfoIndication message for DVB data carousels are as follows:

compatibilityDescriptor(): This structure shall only contain the compatibilityDescriptorLength field of the compatibilityDescriptor() as defined in DSM-CC (see ISO/IEC 13818-6 [5]). It shall be set to the value of 0x0000.

moduleInfoLength: This field defines the length in bytes of the moduleInfo field for the described module.

moduleInfoByte: These fields shall convey a list of descriptors which each define one or more attributes of the described module, except when the moduleId is within the range of 0xFFF0-0xFFFF. In this case, the moduleInfoByte structure contains the ModuleInfo structure as defined by DAVIC with the privateDataByte field of that structure as a loop of descriptors.

privateDataLength: This field defines the length in bytes of the privateDataByte field.

privateDataByte: These fields are user defined.

10.1.4 DownloadDataBlock message

The DownloadDataBlock messages contain the blocks of the fragmented modules. They are conveyed in the payload of MPEG-2 Transport Stream packets as specified in the DSM-CC specification (see ISO/IEC 13818-6 [5]).

10.1.5 DownloadCancel

The DownloadCancel message may be used to indicate to the receivers that the data carousel has aborted the periodic transmission of the modules. DownloadCancel messages may be sent at either the group or the super group level. They are conveyed in the payload of MPEG-2 Transport Stream packets as specified in the DSM-CC specification (see ISO/IEC 13818-6 [5]).

privateDataLength: This field defines the length in bytes of the privateDataByte fields.

privateDataByte: These fields are user defined.

10.2 Descriptors

10.2.1 Descriptor identification and location

Table 44 contains the descriptors that are defined by the DVB data carousel specifications. Note that these descriptors have their own private descriptor tag space.

Table 44: Defined descriptors, values, and allowed locations

Descriptor	Tag value	DII - moduleInfo	DSI - groupinfo	Short description
Reserved	0x00			
Type	0x01	+	+	type descriptor of data
Name	0x02	+	+	name descriptor of data
Info	0x03	+	+	textual description
module_link	0x04	+		concatenated data module
CRC32	0x05	+		Cyclic Redundancy Code (CRC)
Location	0x06	+	+	location of data
est_download_time	0x07	+	+	estimated download time
group_link	0x08		+	links DII messages describing a group
compressed_module	0x09	+		indicates compression structure
SSU_module_type	0x0A	+		TS 102 006 [19] (DVB SSU)
subgroup_association	0x0B		+	TS 102 006 [19] (DVB SSU)

Table 45: Allocation of tag values for DVB data carousel private descriptors

Private descriptor tag in DVB data carousels	Value
0x00 to 0x0B	currently allocated by DVB in table 44
0x0C to 0x6F	reserved for future use by DVB
0x70 to 0x7F	reserved for DVB MHP
0x80 to 0xFF	private descriptors

10.2.2 Type descriptor

The type_descriptor contains the type of the module or group as a sequence of characters. Table 46 shows the syntax of the type_descriptor.

Table 46: Syntax of type_descriptor

Type_descriptor(){	Number of bytes	Value
Descriptor_tag	1	0x01
Descriptor_length	1	
For (i=0; i<N;i++) {		
text_char	1	Text string, e.g. "text/plain"
}		
}		

Semantics of the type_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the type descriptor it is set to 0x01.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

text_char: This is an 8-bit field. A string of "char" fields specifies the type of the module or group following the Media Type specifications RFC 2045 [8] and RFC 2046 [9].

10.2.3 Name descriptor

The name_descriptor contains the name of the module or group. Table 47 shows the syntax of the name_descriptor.

Table 47: Syntax of name_descriptor

name_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x02
descriptor_length	1	
for (i=0; i<N;i++) {		
text_char	1	Name of the module, e.g. "index"
}		
}		

Semantics of the name_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the name_descriptor it is set to 0x02.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

text_char: This is an 8-bit field. A string of "char" fields specifies the name of the module or group. Text information is coded using the character sets and methods described in annex A of EN 300 468 [2].

10.2.4 Info descriptor

The info_descriptor contains a description in plain text. Table 48 shows the syntax of the info_descriptor.

Table 48: Syntax of info_descriptor

info_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x03
descriptor_length	1	
ISO_639_language_code	3	
for (i=0; i<N;i++) {		
text_char	1	Description of the module or group
}		
}		

Semantics of the info_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the info_descriptor it is set to 0x03.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

ISO_639_language_code: This 3-byte field identifies the language of the following text field.
The ISO_639_language_code contains a 3-character code as specified by ISO 639-2 [15]. Each character is coded into 8 bits according to ISO 8859-1 [14] and inserted in order into the 3-byte field.

text_char: This is an 8-bit field. A string of "char" fields specifies the text description of the module. Text information is coded using the character sets and methods described in annex A of EN 300 468 [2].

10.2.5 Module link descriptor

The module_link_descriptor contains the information about which modules are to be linked to get a complete piece of data out of the data carousel. It also informs the decoder on the order of the linked modules. Table 49 shows the syntax of the module_link_descriptor.

Table 49: Syntax of module_link_descriptor

module_link_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x04
descriptor_length	1	
Position	1	
module_id	2	
}		

Semantics of the module_link_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the module_link_descriptor it is set to 0x04.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

position: This is an 8-bit field identifying the position of this module in the chain. The value of 0x00 shall indicate the first module of the list. The value of 0x01 indicates an intermediate module in the list and the value of 0x02 indicates the last module of the list.

module_id: This is a 16-bit field that identifies the next module in the list. This field shall be ignored for the last value in the list.

10.2.6 CRC32 descriptor

The CRC32_descriptor indicates the calculation of a CRC32 over a complete module. Table 50 shows the syntax of the CRC32_descriptor.

Table 50: Syntax of CRC32_descriptor

CRC32_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x05
descriptor_length	1	
CRC_32	4	
}		

Semantics of the CRC32_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the CRC32_descriptor it is set to 0x05.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

CRC_32: This is a 32-bit field which contains the CRC calculated over this module, which shall be calculated according to annex B of the MPEG-2 Systems ISO/IEC 13818-1 [1].

10.2.7 Location descriptor

The location_descriptor contains the location of the PID where blocks, modules or groups can be found containing data of the carousel. Table 51 shows the syntax of the location_descriptor.

Table 51: Syntax of location_descriptor

location_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x06
descriptor_length	1	
location_tag	1	
}		

Semantics of the location_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the location_descriptor it is set to 0x06.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

location_tag: This 8-bit field has the same value as the component_tag field in the stream identifier descriptor.

10.2.8 Estimated download time descriptor

The est_download_time_descriptor contains an integer estimating the download time for a module or group. Table 52 shows the syntax of the est_download_time_descriptor.

Table 52: Syntax of est_download_time_descriptor

est_download_time_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x07
descriptor_length	1	
est_download_time	4	
}		

Semantics of the est_download_time_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the est_download_time_descriptor it is set to 0x07.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

est_download_time: This 32-bit field gives the estimated download time of data in seconds.

10.2.9 Group link descriptor

The group_link_descriptor contains the information about which group descriptions are to be linked to describe a single larger group. This is necessary when the description of modules in a group exceeds the maximum size of a single DownloadInfoIndication message and has to be spread across a number of such messages. It also informs the decoder on the order of the linked group descriptions. This is not strictly necessary since the order of linking is not important. It is purely to provide a means to identify all the group descriptions that are to be linked. Table 54 shows the syntax of the group_link_descriptor.

Table 53: Syntax of group_link_descriptor

group_link_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x08
descriptor_length	1	
Position	1	
group_id	4	
}		

Semantics of the group_link_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the group_link_descriptor it is set to 0x08.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

position: This is an 8-bit field identifying the position of this group description in the chain. The value of 0x00 shall indicate the first group description of the list. The value of 0x01 indicates an intermediate group description in the list and the value of 0x02 indicates the last group description of the list.

group_id: This is a 32-bit field that identifies the next group description in the list. This field shall be ignored for the last value in the list.

10.2.10 Private descriptor

Table 54: Syntax of private_descriptor

Private_descriptor () {	Number of bytes	Value
Descriptor_tag	1	Table 55
Descriptor_length	1	
for (i=0;i<N;i++) {		
descriptor_byte	1	
}		
}		

Semantics of the private descriptor:

descriptor tag: This 8-bit field identifies the descriptor. For the private descriptor values can be used in the range 0x80-0xFF as defined in table 55.

descriptor length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

descriptor_byte: Data bytes of the private descriptor.

10.2.11 Compressed module descriptor

The presence of the compressed_module_descriptor indicates that the data in the module has the "zlib" structure as defined in RFC 1950 [16]. Table 55 shows the syntax of the compressed_module_descriptor.

Table 55: Syntax of compressed_module_descriptor

compressed_module_descriptor(){	Number of bytes	Value
descriptor_tag	1	0x09
descriptor_length	1	
compression_method	1	
original_size	4	
}		

Semantics of the compressed_module_descriptor:

descriptor_tag: This 8-bit field identifies the descriptor. For the compressed_module_descriptor it is set to 0x09.

descriptor_length: This 8-bit field specifies the number of bytes of the descriptor immediately following this field.

compression_method: This is an 8-bit field identifying the compression method used. This identification follows the definition of zlib structure of RFC 1950 [16].

original size: This is a 32-bit field indicating the size in bytes of the module prior to compression.

10.3 PSI and SI specifications

The data broadcast service shall indicate the use of a data carousel by including one or more data_broadcast_descriptors in SI (see EN 300 468 [2]). Each descriptor shall be associated with a particular stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map section for the stream that is used as a data stream.

10.3.1 Data_broadcast_descriptor

The data_broadcast_descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0006 to indicate a DVB data carousel (see TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor (if present in the PSI program map section) for the stream that is used to broadcast the data carousel.

selector_length: This field shall be set to 0x10.

selector_byte: The selector bytes shall convey the data_carousel_info structure which is defined as follows.

Table 56: Syntax for the data_carousel_info_structure

Syntax	Number of bits	Mnemonic
data_carousel_info () {		
carousel_type_id	2	bslbf
Reserved	6	bslbf
transaction_id	32	uimsbf
time_out_value_DSI	32	uimsbf
time_out_value_DII	32	uimsbf
Reserved	2	bslbf
leak_rate	22	bslbf
}		

The semantics of the data_carousel_info structure are as follows:

carousel_type_id: This 2-bit field indicates which kind of data carousel is used. The coding of the bits is as in table 57.

Table 57: Carousel_type_id values

Carousel_type_id values	
00	reserved
01	one layer carousel
10	two layer carousel
11	reserved

reserved: This is a 6-bit field that shall be set to "111111".

transaction_id: This 32-bit field shall have the same value as the transactionId value of the top-level DownloadServerInitiate message or DownloadInfoIndication message. The value of 0xFFFFFFFF shall be used to indicate to receivers that any received DownloadServerInitiate message (in the case of a two layer carousel) or DownloadInfoIndication message (in the case of a one layer carousel) on the associated stream is valid.

time_out_value_DSI: This 32-bit field indicates the recommended time out period in milliseconds that receivers should use to time out the acquisition of the DownloadServerInitiate message. The value of 0xFFFFFFFF shall be used to indicate to receivers that there is no recommended time-out value.

time_out_value_DII: This 32-bit field indicates the recommended time out period in milliseconds that receivers should use to time out the acquisition of the DownloadInfoIndication message. The value of 0xFFFFFFFF shall be used to indicate to receivers that there is no recommended time-out value.

reserved: This is a 2-bit field that shall be set to "11".

leak_rate: This is a 22-bit field that shall indicate the leak rate Rx_n of the data carousel decoder model that is applied by the service (see clause 10). The leak rate is encoded as a 22-bit positive integer. The value of the leak_rate is expressed in units of 50 byte/s.

10.3.2 Stream type

The presence of a data carousel in a service shall be indicated in the program map table of that service by setting the stream type of the stream that contains the data carousel to the value of 0x0B (see ISO/IEC 13818-1 [1]) or an user defined value.

11 Object carousels

11.1 Scope of the object carousels

The object carousel specification has been added in order to support data broadcast services that require the periodic broadcasting of DSM-CC U-U objects through DVB compliant broadcast networks, specifically as defined by DVB SIS (see ETS 300 802 [10]).

Data broadcast according to the DVB object carousel specification is transmitted according to the DSM-CC object carousel and DSM-CC data carousel specification which are defined in MPEG-2 DSM-CC (see ISO/IEC 13818-6 [5]).

11.2 Data transport specification

The specification of DVB object carousels is based on the DSM-CC object carousel specification (see ISO/IEC 13818-6 [5]). A DVB object carousel represents a particular service domain that consists of a collection of DSM-CC U-U objects within a DVB network. The service domain has a service gateway that presents a graph of service and object names to receivers.

The unique identification of the service gateway in broadcast networks is done by means of carousel Network Service Access Point (NSAP) address as defined in DSM-CC (see ISO/IEC 13818-6 [5]). This address contains a network specific part that shall make the address unique within the network environment used. The carousel NSAP address is used to refer to the object carousel from another service domain. For DVB system environments, the syntax and semantics of the carousel NSAP address are defined below.

11.2.1 Carousel NSAP address

The carousel NSAP address has a structure as defined in figure 22 (see ISO/IEC 13818-6 [5]).

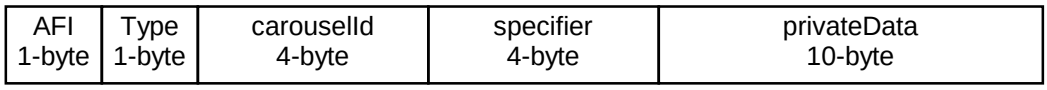

Figure 22: Format of carousel NSAP address

The semantics of the Authority and Format Identifier (AFI), Type, carouselId, and specifier are defined in (see ISO/IEC 13818-6 [5]). In particular:

AFI: This 8-bit field shall be set to the value of 0x00 to indicate the usage of the NSAP format for private use.

Type: This 8-bit field shall be set to 0x00 to indicate the use of the NSAP address for object carousels.

carouselId: This 32-bit field shall be set to the identifier of the object carousel, i.e. the carouselId field.

specifier: This 32-bit field shall convey the specifierType field (set to the value of 0x01) and the Organizational Unique Identifier (OUI) code as defined in DSM-CC (see ISO/IEC 13818-6 [5]). The OUI code shall be set to a value that has been allocated to DVB by the IEEE 802 registration authority.

privateData: This field shall convey the dvb_service_location structure which is defined in table 58.

Table 58: Syntax for DVB_service_location structure

Syntax	Number of bits	Mnemonic
DVB_service_location() {		
transport_stream_id	16	uimsbf
org_network_id	16	uimsbf
service_id	16	uimsbf
Reserved	32	bslbf
}		

The semantics of the dvb_service_location structure are as follows:

transport_stream_id: This is a 16-bit field that identifies the Transport Stream on which the carousel is broadcast.

org_network_id: This 16-bit field that identifies the network_id of the delivery system from which the carousel originates.

service_id: This 16-bit field gives the service identifier of the service that contains the object carousel. The service_id is the same as the program_number in the associated program_map_section.

11.3 Descriptors

NOTE: The descriptors defined for DVB data carousels in clause 2 are usable in DVB object carousels as well.

11.3.1 PSI and SI specifications

The data broadcast service shall indicate the use of a DVB object carousel by including one or more data_broadcast_descriptors in SI (see EN 300 468 [2]). Each descriptor shall point to one DVB object carousel and shall be associated to a particular stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map section for the stream that is used as a data stream.
Each data_broadcast_descriptor allows for the start up of the higher layer protocols based on a language criterion using a list of object names.

DVB object carousels can be implemented using multiple data broadcast services. Data broadcast services may publish that they are part of a particular DVB object carousel by including the carousel_identifier_descriptor as defined by DSM-CC (see ISO/IEC 13818-6 [5]) in the first descriptor loop of the program map table.

Further, DVB object carousels use the concept of Taps (see ISO/IEC 13818-6 [5]) to identify the streams on which the objects are broadcast. The association between Taps and the streams of the data service may be done by either using the association_tag descriptor defined in (see ISO/IEC 13818-6 [5]) or the stream_identifier_descriptor in EN 300 468 [2]. In the latter case, it is assumed that the component_tag field of the stream_identifier descriptor is the least significant byte of the referenced association_tag value which has the most significant byte set to 0x00.

Finally, stream objects within U-U object carousels can be bound to elementary streams of the data broadcasting service itself, to elementary streams of other DVB services, or to complete DVB services. If the stream object is bound to elementary streams of other DVB services, or to complete DVB services the program map table of the data broadcast service shall include the deferred_association_tags_descriptor in the first descriptor loop.

The deferred_association_tags_descriptor is described in clause 10.3.3.

11.3.2 Data_broadcast_descriptor

The data_broadcast_descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0007 to indicate a DVB object carousel (see TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor (if present in the PSI program map table) for the stream that is used to broadcast the object carousel.

selector_length: This field shall be set to length in bytes of the following selector field.

selector_byte: The selector bytes shall convey the object_carousel_info structure which is defined in table 59.

Table 59: Syntax for object_carousel_info structure

Syntax	Number of bits	Mnemonic
object_carousel_info () {		
Carousel_type_id	2	bslbf
Reserved	6	bslbf
Transaction_id	32	uimsbf
time_out_value_DSI	32	uimsbf
time_out_value_DII	32	uimsbf
Reserved	2	bslbf
leak_rate	22	uimsbf
for (I=0;i<N1;i++) {		
ISO_639_language_code	24	bslbf
object_name_length	8	uimsbf
for (j=0;j<N2;j++) {		
object_name_char	8	uimsbf
}		
}		
}		

The semantics of the object_carousel_info structure are as follows:

carousel_type_id: This 2-bit field shall be set to "10" indicating a two-layer carousel.

reserved: This is a 6-bit field that shall be set to "111111".

transaction_id: This 32-bit field shall have the same value as the transactionId value of the DownloadServerInitiate message that carries the object reference of the service gateway. The value of 0xFFFFFFFF shall be used to indicate to receivers that any received DownloadServerInitiate message on the associated stream is valid.

time_out_value_DSI: This 32-bit field indicates the recommended time out period in milliseconds that receivers should use to time out the acquisition of the DownloadServerInitiate message. The value of 0xFFFFFFFF shall be used to indicate to receivers that there is no recommended time-out value.

time_out_value_DII: This 32-bit field indicates the recommended time out period in milliseconds that receivers should use to time out the acquisition of the DownloadInfoIndication message. The value of 0xFFFFFFFF shall be used to indicate to receivers that there is no recommended time-out value.

reserved: This is a 2-bit field that shall be set to "11".

leak_rate: This is a 22-bit field that shall indicate the leak rate Rx_n of the data carousel decoder model that is applied by the service (see clause 10). The leak rate is encoded as a 22-bit positive integer. The value of the leak_rate is expressed in units of 50 byte/s.

ISO_639_language_code: This 24-bit field contains the ISO 639-2 [15] three character language code that is used to select the object necessary to start up the higher layer protocols.

object_name_length: This 8-bit field specifies the number of bytes that follow the object_name_length field for describing characters of the object name.

object_name_char: This is an 8-bit field. A string of object_name_char fields specify the name of the object to be used to start up the higher layer protocols. Text information is coded using the character sets and methods described in annex A of EN 300 468 [2].

11.3.3 Deferred_association_tags_descriptor

The syntax and semantics of the deferred_association_tags_descriptor() in DVB compliant networks are described in table 60.

Table 60: Deferred_association_tags_descriptor

Syntax	Number of bits	Mnemonic
deferred_association_tags_descriptor() {		
descriptor_tag	8	uimsbf
descriptor_length	8	uimsbf
association_tags_loop_length	8	uimsbf
for (i=0;i<N1;i++) {		
association_tag	16	uimsbf
}		
transport_stream_id	16	uimsbf
program_number	16	uimsbf
for (i=0;i<N2;i++) {		
private_data_byte	8	uimsbf
}		
}		

The semantics of the Deferred_association_tags_descriptor are as follows:

descriptor_tag: This field is an 8-bit field. It shall have the decimal value of 21.

descriptor_length: This 8-bit field specifies the length of the descriptor in bytes.

association_tags_loop_length: This 8-bit field defines the length in bytes of the loop of association tags that follows this field.

association_tag: This 16-bit field contains the association_tag that is associated with either a stream that is not part of this data broadcast service or another DVB service.

transport_stream_id: This 16-bit field indicates the Transport Stream in which the service resides that is associated with the enlisted association tags.

program_number: This 16-bit field shall be set to the service_id of the service that is associated with enlisted association tags.

private_data_byte: This field shall contain the deferred_service_location structure which is defined in table 61.

Table 61: Syntax for deferred_service_location structure

Syntax	Number of bits	Mnemonic
deferred_service_location() {		
org_network_id	16	uimsbf
for (i=0; i<N, i++) {		
private_data_byte	8	uimsbf
}		
}		

The semantics of the deferred_service_location structure are as follows:

org_network_id: This 16-bit field that identifies the network_id of the delivery system from which the service originates.

private_data_byte: This 8-bit field is not specified by the present document.

11.3.4 Stream type

The presence of an object carousel in a service shall be indicated in the program map table of that service by setting the stream type of the stream that contains the data carousel to the value of 0x0B (see ISO/IEC 13818-1 [1]) or an user defined value.

12 Higher protocols based on asynchronous data streams

12.1 Data transport specification

Frames of higher protocols are encapsulated in PES packets which are compliant to the PES packets used for asynchronous data streaming. For the mapping of frames into PES packets the following rules apply:

- Multiple frames can be inserted in one PES packet.
- A frame might be spread over several PES packets.
- Stuffing bits shall be used for frames that are not byte-aligned. These bits shall be located at the end of the frame and set to zero.

12.2 PSI and SI specifications

The data broadcast service shall indicate the transmission of asynchronous data frames by including one or more data broadcast descriptors in SI (see EN 300 468 [2] and TR 101 162 [3]).

Each descriptor shall be associated with a stream via a component_tag identifier. In particular, the value of the component_tag field shall be identical to the value of the component_tag field of a stream_identifier_descriptor (see EN 300 468 [2]) that may be present in the PSI program map table for the stream that is used to transmit the frames.

12.2.1 Data_broadcast_descriptor

The data broadcast descriptor is used in the following way:

data_broadcast_id: This field shall be set to 0x0009 to indicate the use of the transmission of higher protocols via asynchronous data streams (see also TR 101 162 [3]).

component_tag: This field shall have the same value as a component_tag field of a stream_identifier_descriptor that may be present in the PSI program map section for the stream on which the data is broadcast.

selector_length: This field is an 8-bit field specifying the total number of selector_bytes.

selector_byte: the selector bytes shall convey the higher_protocol_asynchronous_data_info structure which is defined in table 62.

Table 62: Syntax for higher_protocol_asynchronous_data_info structure

Syntax	Number of bits	Mnemonic
higher_protocol_asynchronous_data_info () {		
higher_protocol_id	4	uimsbf
reserved	4	bslbf
for (i=0; i<N;i++){		
private_data_byte	8	bslbf
}		
}		

The semantics of the higher_protocol_asynchronous_data_info structure are as follows:

higher_protocol_id: This 4-bit field shall indicate the protocol that is used on top of the asynchronous data streaming. The coding shall be according to table 63.

Table 63: Coding of the higher_protocol_id

higher_protocol_id	higher protocol
0x00	reserved
0x01	dGNSS data [17]
0x02	TPEG [18]
0x03 to 0x0F	reserved

private_data_byte: This is an 8-bit field, the value of which is privately defined.

12.2.2 Stream type

The presence of a multiprotocol data stream in a service shall be indicated in the program map section of that service by setting the stream type of that stream to the value of 0x06 or a user private value.

13 Decoder models

The decoder model description is common to the data streaming, multiprotocol encapsulation, data carousel, and object carousel specifications.

The data service decoder model is a conceptual model for decoding data streams. The decoder model is used to specify the delivery of the bits in time. The decoder model does not specify the operation or behaviour of a real decoder implementation and implementations which do not follow the architecture or timing of this model are not precluded.

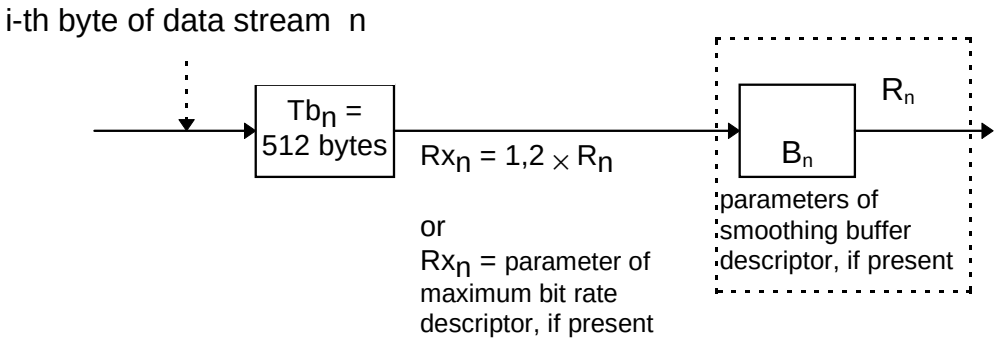

Figure 23: Data service decoder model

Figure 23 shows the structure of the data service decoder model for a single data stream n, which is similar to the Transport System Target Decoder (T-STD) model of ISO/IEC 13818-1 [1]. The symbols Tb_n, B_n, Rx_n, and R_n are defined as follows:

- Tb_n is the transport buffer for data stream n;
- B_n is the main buffer for data stream n;
- Rx_n is the rate at which data is removed from Tb_n; and
- R_n is the rate at which data is removed from B_n.

Complete TS packets containing data from the data stream n are inserted in the transport buffer for stream n, Tb_n. All bytes that enter the buffer Tb_n are removed at rate Rx_n specified below. Bytes which are part of a PES packet, a section, or the contents of these containers are delivered to the main buffer B_n. Other bytes are not, and may be used to control the system. Duplicate TS packets are not delivered to B_n. All bytes that enter the buffer B_n are removed at rate R_n specified below.

The transport buffer Tb_n is 512 bytes.

The transport buffer leak rate Rx_n, the size of the buffer $B_{n,}$ and the leak rate R_n are specific to a particular service. The service may indicate the values for $Rx_{n,}$ B_n and R_n by means of the MPEG-2 maximum_bit_rate_descriptor and the smoothing_buffer_descriptor (see ISO/IEC 13818-1 [1]). If used, the descriptors shall be included in the SDT or EIT as well as in the PMT of the service.

The maximum_bit_rate field of the maximum_bit_rate descriptor shall indicate the value that is applied for Rx_n.

The sb_size field of the smoothing_buffer_descriptor shall contain the value of B_n. The sb_leak_rate field shall contain the value of R_n.

If the maximum_bit_rate_descriptor is not included in SI and PSI, but the smoothing_buf-fer_descriptor is included, then $Rx_n = 1{,}2R_n$.

If the smoothing_buffer_descriptor is not included in SI and PSI, but the maximum_bit_rate_descriptor is included, then the two buffer model becomes a single buffer model that consists of the Transport buffer Tb_n with a leak rate Rx_n.

If neither of the descriptors is included in SI and PSI, then the buffer model is not applicable. In this case, the delivery of the bits is service specific.

Annex A (informative): Registration of private data broadcast systems

TR 101 162 [3] will be extended to include the allocation of the values for the data_broadcast_id. For each data stream in a multiplex the data_broadcast_id identifies the data broadcast profile or private system being used.

Seven values (see table A.1) have been reserved for the different profiles defined in the present document. There is a wide range of values (0x100-0xFFFF) that can be used for the registration of private systems. TR 101 162 [3], which is frequently updated, gives a list of all registered data_broadcast_id.

The registration of a data broadcast solution is highly recommended since it allows for a minimum of interoperability. It helps decoders to identify data streams they can support and prevents them from trying to acquire data streams they do not comply with.

Organizations who register private implementations are invited, but not obliged, to publish the specifications of their systems in order to allow manufacturers to build compatible equipment.

Registrations can be obtained from the:

DVB Project Office
c/o European Broadcasting Union
17a Ancienne Route
CH-1218 Grand Saconnex (GE)

Table A.1: Allocation of data_broadcast_id values

Data Broadcast specification	data_broadcast_id
reserved for future use	0x0000
bit pipe	0x0001
asynchronous data stream	0x0002
synchronous data stream	0x0003
synchronized data stream	0x0004
multiprotocol encapsulation	0x0005
data carousel	0x0006
object carousel	0x0007
DVB ATM streams	0x0008
reserved for future use by DVB	0x0009 to 0x00EF
reserved for DVB MHP	0x00F0 to 0x00FF
reserved for registration	0x0100 to 0xFFFE
reserved for future use	0xFFFF

Annex B (normative): Simulcasting of IP/MAC streams

- Group destination addresses: If the same address within an IP/MAC platform is signalled for more than one DVB location, the data in the respective IP/MAC streams must be the same. The multicast IP/MAC address mapping shall be according to RFC 1112 [7] for IPv4 and RFC 2464 [20] for IPv6.

- Unicast address destination: If the same address within an IP/MAC platform is signalled for more than one DVB location, the broadcast/network operator need not duplicate the data on these IP/MAC streams. If they are located on the same transport stream, a receiver should listen to all of them, as they may have different content.

Annex C (normative): Minimum repetition rates for the INT

Minimal INT repetition rates are defined as:

- 10 s on cable and satellite networks;
- 30 s on terrestrial networks.

Annex D (informative): IP/MAC Platform ID values:

Table D.1: Allocation of platform_id values

Platform_id	Owner
0x000000	Reserved by DVB.
0x000001 to 0xFFEFFF	Reserved for registration. These platform_id values are globally unique.
0xFFF000 to 0xFFFFFE	Managed by the network operator, and may be used for IP/MAC Platforms supporting services only within a single DVB network. These platform_id values are unique within a network_id only.
0xFFFFFF	Reserved by DVB.

These values are specified in TR 101 162 [3].

Annex E (informative): Bibliography

- ETSI TS 101 812: "Digital Video Broadcasting (DVB); Multimedia Home Platform (MHP) Specification 1.0.3".

History

Document history		
V1.1.1	October 1997	Publication as TS 101 192 (Withdrawn)
V1.1.1	December 1997	Publication
V1.2.1	December 1997	Publication as TS 101 192 (Withdrawn)
V1.2.1	June 1999	Publication
V1.3.1	May 2003	Publication
V1.4.1	June 2004	One-step Approval Procedure OAP 20041022: 2004-06-23 to 2004-10-22
V1.4.1	November 2004	Publication