Draft EN 301 192 V1.2.1 (1999-01)

European Standard (Telecommunications series)

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

Reference
REN/JTC-DVB-88 (b1c00ioo.PDF)

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

ETSI

Postal address
F-06921 Sophia Antipolis Cedex - FRANCE

Office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - 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

Internet
secretariat@etsi.fr
Individual copies of this ETSI deliverable
can be downloaded from
http://www.etsi.org
If you find errors in the present document, send your
comment to: editor@etsi.fr

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 SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available **free of charge** from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

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 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 the Joint Technical Committee (JTC) of the European Broadcasting Union (EBU), Comité Européen de Normalization ELECtrotechnique (CENELEC) and the European Telecommunications Standards Institute (ETSI), and is now submitted for the ETSI standards One-step Approval Procedure.

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

Digital Video Broadcasting (DVB) Project

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.

Proposed national transposition dates	
Date of latest announcement of this EN (doa):	3 months after ETSI publication
Date of latest publication of new National Standard or endorsement of this EN (dop/e):	6 months after doa
Date of withdrawal of any conflicting National Standard (dow):	6 months after doa

1 Scope

The present document is designed to be used in conjunction with EN 300 468 [2] and ETR 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]).

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]).

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, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.
- A non-specific reference to an ETS shall also be taken to refer to later versions published as an EN with the same number.

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

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

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

[4] ETR 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] 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", Stanford University.

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

[11] ISO/IEC 8802-1 and 2: "Information technology; Telecommunications and information exchange between systems; Local and metropolitan area networks; Specific requirements;
Part 1: Overview of Local Area Network Standards
Part 2: Logical link control".

[12] ETS 300 743: "Digital Video Broadcasting (DVB); Subtitling systems".

[13] ISO 8859: "Information processing - 8-bit single-byte coded graphic character sets, Latin alphabets".

[14] ISO 639-2: "Codes for the representation of names of languages - Part 2: Alpha-3 code3".

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

3 Abbreviations

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

AFI	Authority and Format Identifier
BCD	Binary Coded Decimal
bslbf	bit string, left bit first
CRC	Cyclic Redundancy Code
DAVIC	Digital Audio VIsual Council
DDB	DownloadDataBlock
DII	DownloadInfoIndication
DSI	DownloadServerInitiate
DSM-CC	DSM-CC data carousel specification
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
EIT	Event Information Table
ETR	ETSI Technical Report
ETS	European Telecommunication Standard
gi	GroupInfoByte
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers (USA)
IP	Internet Protocol
ISO	International Organization for Standardization
LLC	Logical Link Control
MAC	Media Access Control
mi	ModuleInfoBytes
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Pictures Expert Group
NSAP	Network Service Access Point
OUI	Organizational Unique Identifier
PCR	Program Clock Reference
PES	Program Elementary Stream
PID	Packet IDentifier
PMT	Program Map Table
PSI	Program Specific Information
PTS	Presentation Time Stamps
RFC	Request For Comment
rpchof	remainder polynomial coefficients, highest order first
RS	Reed Solomon
SDT	Service Description Table
SI	Service Information
SIS	Systems for Interactive Services
SNAP	SubNetwork Attachment Point
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 ETR 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 ETR 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 an 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	No. 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 ETR 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 ETS 300 743 [12])
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 an 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 an 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 us (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 ETR 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 an 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	No. 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.

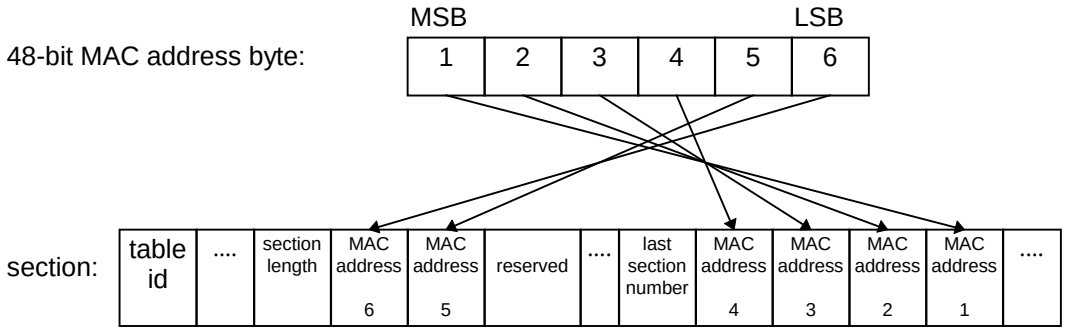


Figure 1: Mapping of MAC address bytes to section fields

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 subclause (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 [11] Logical Link Control (LLC) and ISO/IEC 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 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 ETR 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 ETR 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	No. 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]. 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 a 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 Data carousels

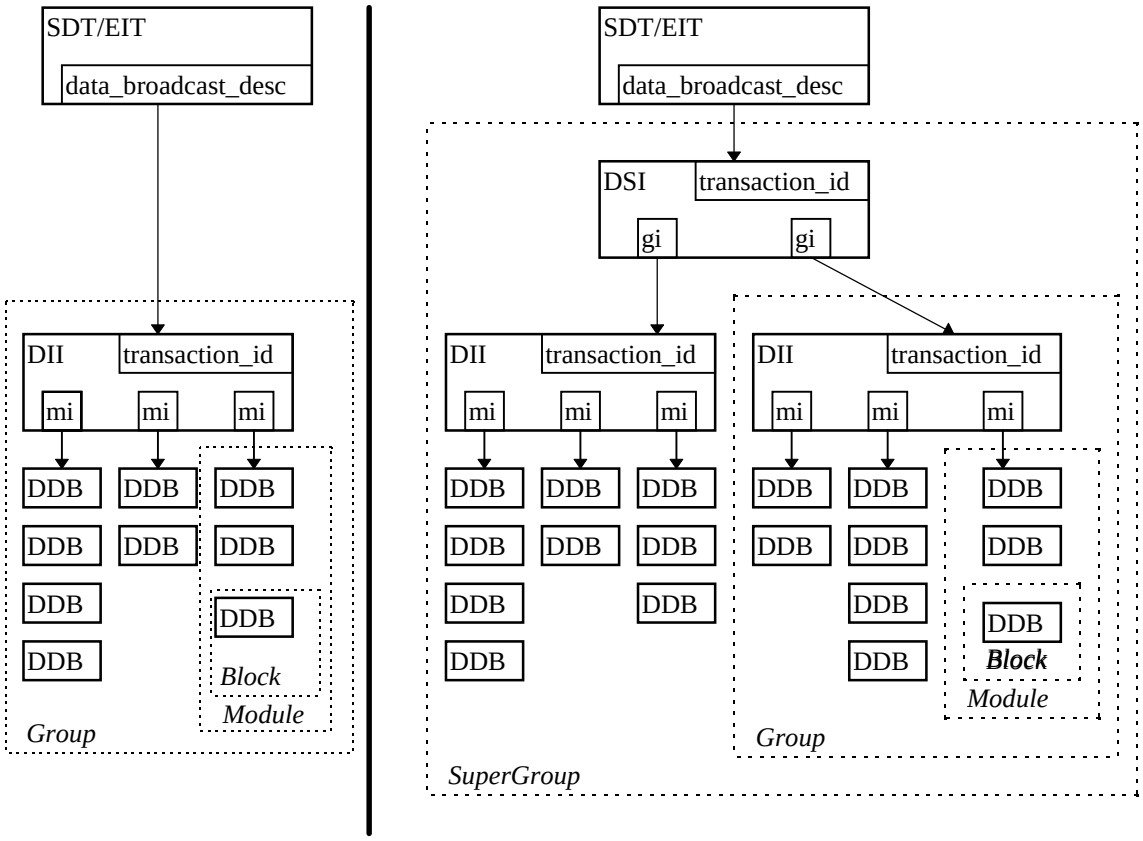
8.1 Data transport specification

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 (DDB) 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.

8.1.1 Structure of DVB data carousel



DSI: DownloadServerInitiate
gi: GroupInfoBytes
DII: DownloadInfoIndication
mi: ModuleInfoBytes
DDB: DownloadDataBlock
⟶ : Location reference (transactionId, optional componentTag)

Figure 2: Structure of the DVB data carousel

DVB data carousels can have one or two layers of control information as illustrated in Figure 2. 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 (mi) 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 is 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 (gi) 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 2 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:

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

- a transaction/module identifier, i.e. an 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 loacated 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 0x0000-0x0001.

- For DownloadInfoIndication messages the 2 least significant bytes of the transactionId shall be in the range 0x0002-0xFFFF.

For the one-layer carousel:

- For DownloadInfoIndication messages the 2 least significant bytes of the transactionId shall be in the range 0x0000-0x0001.

8.1.2 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 below:

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 9.

Table 9: GroupInfoIndication structure

Syntax	Num. 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.

8.1.3 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.

8.1.4 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]).

8.1.5 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.

8.2 Descriptors

8.2.1 Descriptor identification and location

Table 10 contains the descriptors that are defined by the DVB data carousel specifications. These descriptors have their own private descriptor_tag space which implies that they can not be used outside the scope of DVB data carousels.

Table 10: 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

Table 11: Allocation of tag values for DVB data carousel private descriptors

Private descriptor tag in DVB data carousels	Value
0x00 - 0x09	currently allocated by DVB in Table 10
0x0A - 0x7F	reserved for future use by DVB
0x80 - 0xFF	private descriptors

8.2.2 Type descriptor

The type_descriptor contains the type of the module or group as a sequence of characters. Table 12 shows the syntax of the type_descriptor.

Table 12: Syntax of type_descriptor

Type_descriptor(){	No. 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 [8] and [9].

8.2.3 Name descriptor

The name_descriptor contains the name of the module or group. Table 13 shows the syntax of the name_descriptor.

Table 13: Syntax of name_descriptor

name_descriptor(){	No. 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].

8.2.4 Info descriptor

The info_descriptor contains a description in plain text. Table 14 shows the syntax of the info_descriptor.

Table 14: Syntax of info_descriptor

info_descriptor(){	No. 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 [14]. Each character is coded into 8 bits according to ISO 8859-1 [13] 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].

8.2.5 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 15 shows the syntax of the module_link_descriptor.

Table 15: Syntax of module_link_descriptor

module_link_descriptor(){	No. 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.

8.2.6 CRC32 descriptor

The CRC32_descriptor indicates the calculation of a CRC32 over a complete module. Table 16 shows the syntax of the CRC32_descriptor.

Table 16: Syntax of CRC32_descriptor

CRC32_descriptor(){	No. 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 an 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].

8.2.7 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 17 shows the syntax of the location_descriptor.

Table 17: Syntax of location_descriptor

location_descriptor(){	No. 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.

8.2.8 Estimated download time descriptor

The est_download_time_descriptor contains an integer estimating the download time for a module or group. Table 18 shows the syntax of the est_download_time_descriptor.

Table 18: Syntax of est_download_time_descriptor

est_download_time_descriptor(){	No. 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.

8.2.9 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 19 shows the syntax of the group_link_descriptor.

Table 19: Syntax of group_link_descriptor

group_link_descriptor(){	No. 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.

8.2.10 Private descriptor

Table 20: Syntax of private_descriptor

Private_descriptor () {	No. of bytes	Value
Descriptor_tag	1	Table 11
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 11.

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.

8.2.11 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 [15]. Table 21 shows the syntax of the compressed_module_descriptor.

Table 21: Syntax of compressed_module_descriptor

compressed_module_descriptor(){	No. 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 [15].

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

8.3 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.

8.3.1 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 ETR 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 22: Syntax for the data_carousel_info_structure

Syntax	No. 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 follows:

Table 23: 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.

8.3.2 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.

9 Object carousels

9.1 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]).

9.2 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.

9.2.1 Carousel NSAP address

The carousel NSAP address has a structure as defined below (see ISO/IEC 13818-6 [5]):

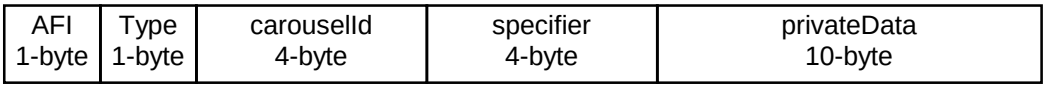


Figure 3: Format of carousel NSAP address

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 24.

Table 24: Syntax for DVB_service_location structure

Syntax	No. 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.

9.3 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 subclause 9.3.2.

9.3.1 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 ETR 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 as follows.

Table 25: Syntax for object_carousel_info structure

Syntax	No. 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 [14] 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 a 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].

9.3.2 Deferred_association_tags_descriptor

The syntax and semantics of the deferred_association_tags_descriptor() in DVB compliant networks are described in Table 26.

Table 26: Deferred_association_tags_descriptor

Syntax	No. 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
}		
}		

descriptor_tag: this field is an 8-bit field. It shall have the value of 0x15.

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 27.

Table 27: Syntax for deferred_service_location structure

Syntax	No. 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.

9.3.3 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.

10 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.

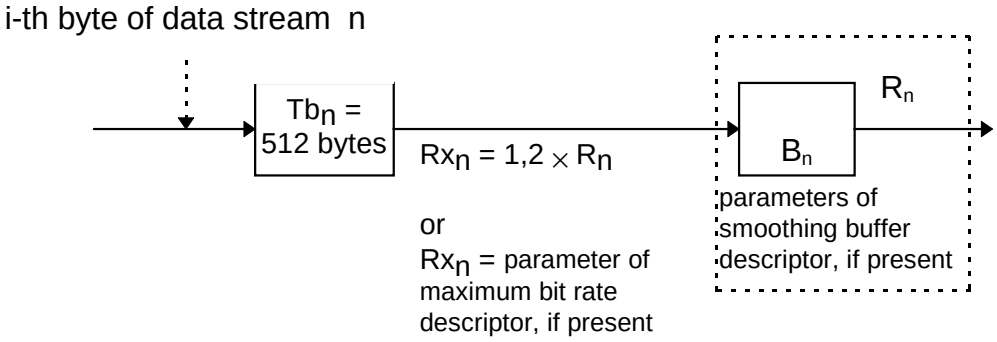


Figure 4: Data service decoder model

Figure 4 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_de-scriptor 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 are 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

ETR 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. ETR 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

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	0x0009-0x00FF
reserved for registration	0x0100-0xFFFE
reserved for future use	0xFFFF

History

Document history			
V1.1.1	August 1997	One-step Approval Procedure	OAP 9748: 1997-08-01 to 1997-11-28
V1.1.1	December 1997	Publication	
V1.2.1	January 1999	One-step Approval Procedure	OAP 9922: 1999-01-29 to 1999-05-28