Specification for Data Broadcasting

-

Commercial Requirements

DVB DOCUMENT A027

May 1997

DVB Project Office

15 May 1997

Contents

1. Scope

This specification is designed to be used in conjunction with the DVB-SI Standard [2] and the DVB-SI Implementation Guidelines [4].

Furthermore, it is designed to comply with the Commercial Requirements as outlined by the DVB Commercial Module's Ad-hoc Group on Data Broadcasting.

2. Introduction

The DVB System provides a means of delivering MPEG-2 Transport Streams via a variety of transmission media. These Transport Streams 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 this draft specification.

The following is a short description of the application areas and the profiles.

2.1 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 Transport Stream packets [1].

2.2 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 synchronised way through DVB compliant broadcast networks.

Asynchronous data streaming is defined as the streaming of only data without any timing requirements (e.g. RS232 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).

Synchronised 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 synchronisation with other kinds of data streams (e.g. audio, video).

Data broadcast according to the data streaming specification is carried in PES packets which are defined in MPEG-2 Systems [1].

2.3 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 [5], which are compliant with the MPEG-2 private section format [1].

2.4 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 [5]. This specification 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.

2.5 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 [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 [5].

3. Data Piping

3.1 Data transport specification

The data broadcast service shall insert the data to be broadcast directly in the payload of MPEG-2 Transport Stream 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 this specification.

3.2 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 [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 [2] that may be present in the PSI program map section for the stream that is used as a data pipe.

3.2.1 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 [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.

3.2.2 Stream type

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

4. Asynchronous Data Streaming

4.1 Data transport specification

The data broadcast service shall insert the data to be broadcast in PES packets as defined by MPEG-2 Systems [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 [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.

4.2 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 [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 [2] that may be present in the PSI program map section for the stream that is used as a data stream.

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 0x0002 to indicate an asynchronous data stream [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.

4.2.2 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.

5. Synchronous and Synchronised 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. 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 [1].

The synchronous and synchronised 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) or 0xBF (private_stream_2) for synchronous data streams. For synchronised data streams this value shall be set to 0xBD (private_stream_1).

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 below.

Syntax	No. of bits	Mnemonic
PES_data_packet () {		
data_identifier	8	uimsbf
sub_stream_id	8	uimsbf
reserved	4	bslbf
PES_data_packet_header_length	4	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
}		

Table 5.1: Syntax for PES_data_packet structure.

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 5.2 (see also [3,6]):

data_identifier	value
0x00 to 0x0F	reserved for future use
0x10 to 0x1F	reserved for EBU data (see [6])
0x20	DVB subtitling (see [12])
0x21	DVB synchronous data stream
0x22	DVB synchronised data stream
0x23 to 0x7F	reserved for future use
0x80 to 0xFF	User defined

Table 5.2: Coding for data_identifier field.

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.

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.

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.

5.2 PSI and SI specifications

The data broadcast service shall indicate the use of a synchronous or synchronised data stream by including one of more data_broadcast_descriptors in SI [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 [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 0x0003 to indicate a synchronous data stream and to 0x0004 for synchronised data streams [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 a synchronous data stream or a synchronised 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.

6. Multiprotocol Encapsulation

6.1 Data transport specification

Datagrams are encapsulated in datagram_sections which are compliant to the DSMCC_section format for private data [5]. The mapping of the section into MPEG-2 Transport Stream packets is defined in MPEG-2 Systems [1], and is described in Annex D for reason of convenience.

The syntax and semantics of the datagram_section are defined below.

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
}		
}		

Table 6.1: Syntax of datagram_section.

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 [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 6.1 illustrates the mapping of the MAC address bytes in the section fields. Note that the order of the bits in the bytes is not reversed and that the most significant bit of each byte is still transmitted first.

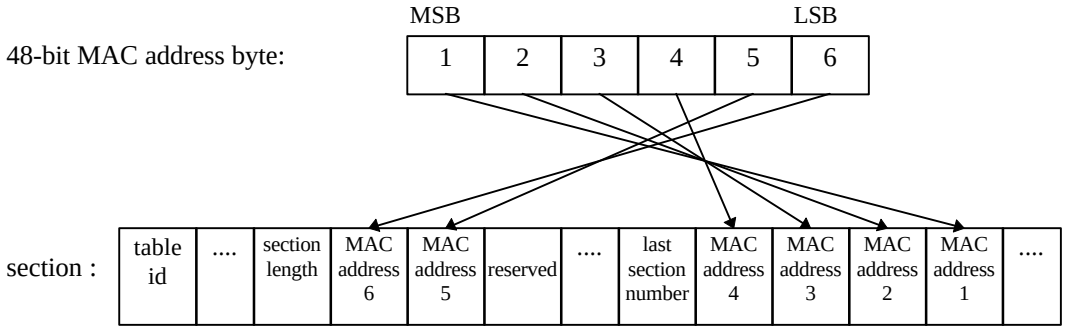


Figure 6.1: Mapping of MAC address bytes to section fields.

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_byte_1 but excludes the checksum or CRC32 field. See Table 6.2. The scrambling method applied is user private.

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

Table 6.2: Coding of the payload_scrambling_control field.

address_scrambling_control: this 2-bit field defines the scrambling mode of MAC address in this section. See Table 6.3. This field enables a dynamic change of MAC addresses. The scrambling method applied is user private.

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

Table 6.3: Coding of the address_scrambling_control field.

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 8802-1a 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.

6.2 PSI and SI specifications

The data broadcast service shall indicate the transmission of datagrams by including one or more data broadcast descriptors in SI [2],[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 [2] that may be present in the PSI program map table for the stream that is used to transmit the datagrams.

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 0x0005 to indicate the use of the multiprotocol encapsulation specification (see also[3]) .

component_tag: this field shall have the same value as a component_tag field of a stream_identifier_descriptor that shall 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 6.4.

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
}		

Table 6.4: Syntax for multiprotocol_encapsulation_info structure.

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 6.5.

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

Table 6.5:Coding of the MAC_address_range field.

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 this specification.

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 6.6..

value	alignment in bits
00	8 (default)
01	32

Table 6.6: Coding of the alignment_indicator field.

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.

6.2.2 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 [5] or a user defined value.

7. Data Carousels

7.1 Data transport specification

The specification of DVB data carousels is based on the DSM-CC data carousel specification [5]. The DSM-CC data carousel specification embodies the cyclic transmission of data to receivers. The data transmitted within the data carousel is organised 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.

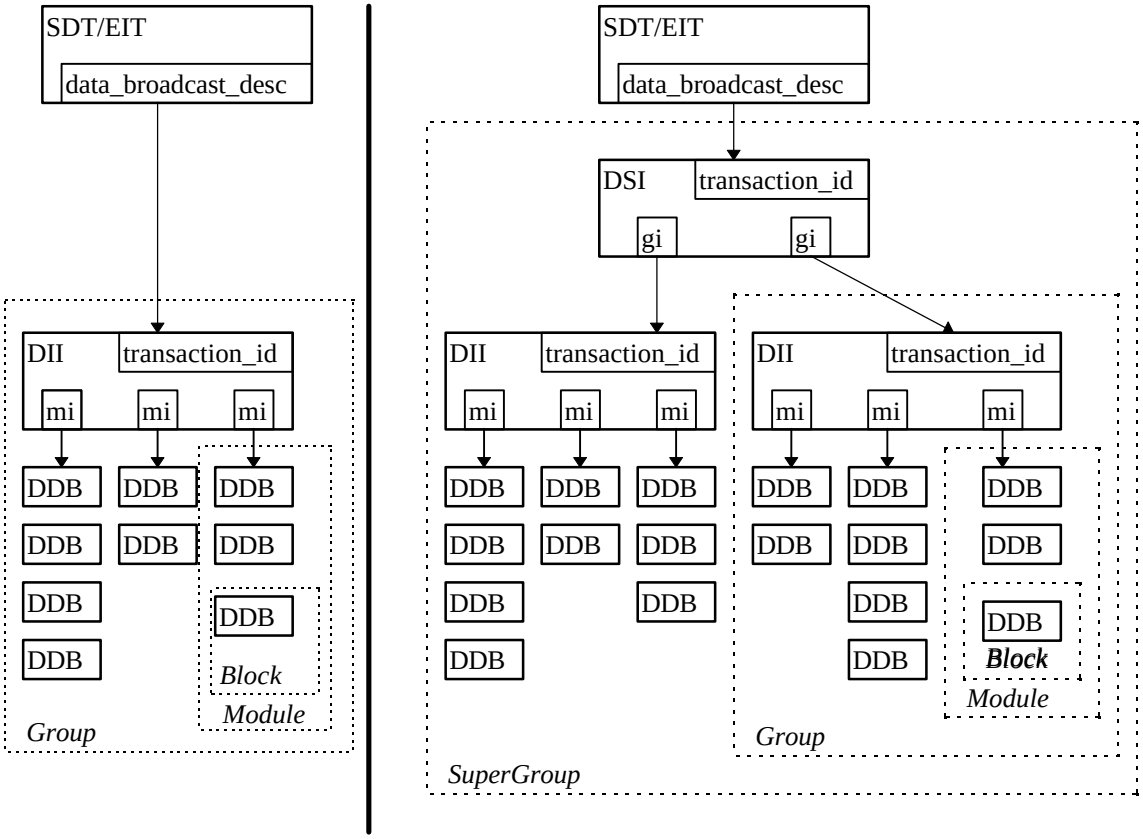
The data carousel specification uses four messages of the DSM-CC Download specification. The data is carried in DownloadDataBlock messages, while the control over the Modules is provided by DownloadInfoIndication, DownloadServerInitiate, and DownloadCancel messages. The Download-ServerInitiate message 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 [5], and are provided for convenience purposes in Annex B.

The use of these messages in DVB data carousels is described below.

7.1.1 Structure of DVB data carousel

one-layer data carousel

two-layer data carousel



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

Figure 7.1 : Structure of the DVB data carousel.

DVB data carousels can have one or two layers of control information as illustrated in Figure 7.1. 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 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 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 7.1 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. an unique identifier of a control message or a module.

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

In order to give information on the time to download data from a carousel, provisions have been made to be included in the data_broadcast_descriptor. Furthermore in the DownloadServerInitiate and DownloadInfoIndication messages parameters for the sizes of modules and blocks have been specified, based on DSM-CC.

Within this specification 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.

All DownloadServerInitiate 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's and module_id's can be used to efficiently filter the data of the data carousel, based on the following semantics:

- For DownloadServerInitiate messages the 2 least significant bytes of the transaction_id shall be in the range 0x0000 - 0x0001.
- DownloadInfoIndication messages the 2 least significant bytes of the transaction_id shall be in the range 0x0002 - 0xFFFF.
- For DownloadCancel messages no restrictions do apply.

7.1.2 DownloadServerInitiate message

The DownloadServerInitiate message (see Annex C) 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 [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 7.1.

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**
}	
}	

Table 7.1 GroupInfoIndication structure

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 16-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 [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.

7.1.3 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 [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 [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.

7.1.4 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 [5].

7.1.5 DownloadCancel

The DownloadCancel message may be used to indicate to the receivers that the data carousel aborts 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 [5].

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

privateDataByte: These fields are user defined.

7.2 Descriptors

7.2.1 Descriptor identification and location

Table 7.2 tabulates the descriptors that are defined by the DVB data carousel specifications. It should be noted that these descriptors have an own private descriptor_tag space which implies that they can not be used outside the scope of DVB data carousels.

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
location	0x06	+	+	Location of data
est_download_time	0x07	+	+	estimated download time
group_link	0x08		+	Links DII messages describing a Group

Table 7.2: Defined descriptors, values, and allowed locations.

7.2.2 Type

The type_descriptor contains the type of the module or Group as a sequence of characters. Table 7.3 shows the syntax of the 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/html”
}		
}		

Table 7.3: Syntax of type_descriptor.

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 following the Media Type specifications RFC 1521 [8] and RFC 1590 [9].

7.2.3 Name

The name_descriptor contains the name of the Module or Group. Table 7.4 shows the syntax of the 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"
}		
}		

Table 7.4: Syntax of name_descriptor.

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. Text information is coded using the character sets and methods described in annex A of ETS 300 468.

7.2.4 Info

The info_descriptor contains a descripton in plain text. Table 7.5 shows the syntax of the 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
}		
}		

Table 7.5: Syntax of info_descriptor.

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. Each character is coded into 8 bits according to ISO 8859-1 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 ETS 300 468.

7.2.5 Module Link

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 7.6 shows the syntax of the module_link_descriptor.

module_link_descriptor(){	No.of bytes	Value
descriptor_tag	1	0x04
descriptor_length	1	
position	1	
module_id	2	
}		

Table 7.6: Syntax of module_link_descriptor.

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.

7.2.6 CRC32

The CRC32_descriptor indicates the calculation of a CRC32 over a complete module. Table 7.7 shows the syntax of the CRC32_descriptor.

CRC32_descriptor(){	No.of bytes	Value
descriptor_tag	1	0x05
descriptor_length	1	
CRC_32	4	
}		

Table 7.7: Syntax of CRC32_descriptor.

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 spec [1].

7.2.7 Location

The location_descriptor contains the location of the PID where Blocks, Modules or Groups can be found containing data of the carousel. Table 7.8 shows the syntax of the location_descriptor.

location_descriptor(){	No.of bytes	Value
descriptor_tag	1	0x06
descriptor_length	1	
location_tag	1	
}		

Table 7.8: Syntax of location_descriptor.

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.

7.2.8 Estimated download time

The est_download_time_descriptor contains an integer estimating the download time for a Module or Group. Table 7.9 shows the syntax of the est_download_time_descriptor.

est_download_time_descriptor(){	No.of bytes	Value
descriptor_tag	1	0x07
descriptor_length	1	
est_download_time	4	
}		

Table 7.9: Syntax of est_download_time_descriptor.

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.

7.2.9 Group Link

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 accross 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 7.6 shows the syntax of the group_link_descriptor.

group_link_descriptor(){	No.of bytes	Value
descriptor_tag	1	0x08
descriptor_length	1	
position	1	
group_id	4	
}		

Table 7.10: Syntax of group_link_descriptor.

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.

7.3 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 [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 [2] that may be present in the PSI program map section for the stream that is used as a data stream.

7.3.1 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 [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.

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
}		

Table 7.10: Syntax for the data_carousel_info_structure

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:

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

Table 7.11: carousel_type_id values

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 layercarousel) 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 section 9). The leak rate is encoded as a 22-bit positive integer. The value of the leak_rate is expressed in units of 50 bytes/second.

7.3.2 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 [1] or an user defined value.

8. Object Carousels

8.1 Scope

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 [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 [5].

8.2 Data transport specification

The specification of DVB object carousels is based on the DSM-CC Object Carousel specification [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 NSAP address as defined in DSM-CC [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.

8.2.1 Carousel NSAP address

The Carousel NSAP address has a structure as defined below [5]:

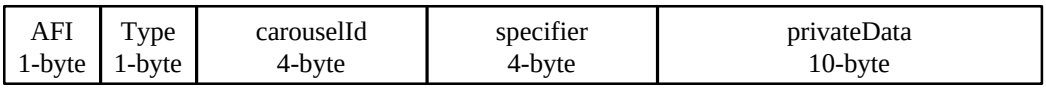


Figure 8.1: Format of Carousel NSAP address

The semantics of the AFI, Type, carouselId, and specifier are defined in [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 OUI code as defined in DSM-CC [5]. The OUI code shall be set to 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 1.

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
}		

Table 8.1: Syntax for DVB_service_location structure.

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 orginates.

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.

8.3 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 [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 [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 service may publish that they are part of a particular DVB object carousel by including the carousel_identifier_descriptor as defined by DSM-CC [5] in the first descriptor loop of the program map table.

Further, DVB object carousels use the concept of Taps [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 [5] or the stream_identifier_descriptor in [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 Section 8.3.2.

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 0x0007 to indicate a DVB object carousel [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 data 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.

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
}		
}		
}		

Table 8.2: Syntax for object_carousel_info structure.

The semantics of the object_carousel_info structure are as follows.

carousel_type_id: This 2-bit field indicates which kind of object carousel is used. The coding of the bits is as follows:

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

Table 8.3: carousel_type_id values

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 Section 9). The leak rate is encoded as a 22-bit positive integer. The value of the leak_rate is expressed in units of 50 bytes/second.

ISO_639_language_code : this 24-bit field contains the ISO 639.2 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 ETS 300 468.

8.3.2 Deferred_association_tags_descriptor

The syntax and semantics of the deferred_association_tags_descriptor() in DVB compliant networks are described below:

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
}		
}		

Table 8.4 deferred_association_tags_descriptor

descriptor_tag : this field is an 8-bit field. It shall have the decimal value of 0x21.

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 below.

Syntax	No. of bits	Mnemonic
deferred_service_location() {		
org_network_id	16	uimsbf
for (i=0; i<N, i++) {		
private_data_byte	8	uimsbf
}		
}		

Table 8.5: Syntax for deferred_service_location structure.

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 orginates.

private_data_byte: this 8-bit field is not specified by this specification.

8.3.3 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 [1] or an user defined value.

9. 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.

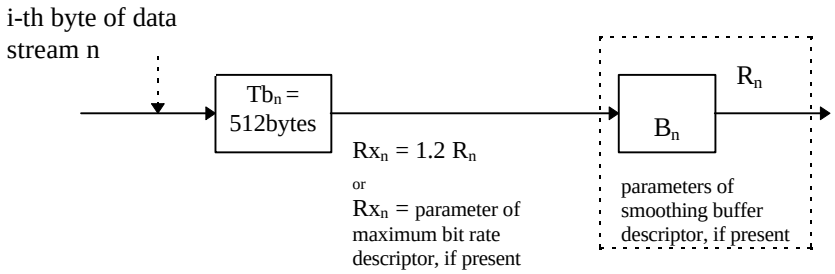


Figure 9.1: Data service decoder model.

Figure 9.1 shows the structure of the data service decoder model for a single data stream n, which is similar to the T-STD model of [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 Transport Stream 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 Transport Stream packets are not delivered to B_n. All bytes that enter the buffer B_n are removed at rate R_n specified below.

For all data streams specified in the data broadcast specification 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 [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_buffer_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 are included in SI and PSI, then the buffer model is not applicable. In this case, the delivery of the bits is service specific.

10. References:

[1] ISO/IEC 13818-1: Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems - International Standard (IS).

[2] ETS 300 468: Digital broadcasting systems for television, sound and data services; Specification for Service Information (SI) in Digital Video Broadcasting (DVB) systems

[3] ETR 162: Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems

[4] ETR 211: Digital broadcasting systems for television; Guidelines on the 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: Extension for Digital Storage Media Command and Control (DSM-CC) - International Standard (IS).

[6] ETS 300 472: Digital broadcasting systems for television, sound and data services; Specification for conveying ITU-R System B Teletext in Digital Video Broadcasting (DVB) systems

[7] RFC 1112, Host extensions for IP multicast, S. Deering, Stanford University, August 1988.

[8] RFC 1521,. N. Borenstein, N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", 09/23/1993.

[9] RFC 1590, J. Postel, "Media Type Registration Procedure", 03/02/1994, (Updates RFC1521)

[10] pr ETS 300 802: Digital Video Broadcasting (DVB); Network independent protocols for interactive services, January 1997

[11] ISO/IEC 8802

[12] pr ETS 300 743: Digital Video Broadcasting (DVB); DVB Subtitling system

[13] ISO 8859 “Information processing - 8-bit single-byte coded graphic character sets, Latin alphabets”

[14] ISO 6392: “Code for the representation of names of languages”

11. Annex A: 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 this standard. There is a wide range of values (0x100 ... 0xFFFF) that can be used for the registration of private systems. ETR 162, which is frequently updated, gives a list of all registered data_broadcast_ids.

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.

Organisations who register private implementations are invited but not obliged to publish the specifications of their systems in order to allow manufactureres 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)

Data Broadcast specification.	**data_broadcast_id**
reserved for future use	0x0000
Bit pipe	0x0001
Asynchronous data stream	0x0002
Synchronous data stream	0x0003
Synchronised data stream	0x0004
Multiprotocol encapsulation	0x0005
Data Carousel	0x0006
Object Carousel	0x0007
reserved for future use by DVB	0x0008-0x00FF
reserved for registration	0x0100-0xFFFE
reserved for future use	0xFFFF

Table A.1: Allocation of data_broadcast_id values

DATA BROADCASTING SERVICES

Commercial Requirements

1. Introduction

The DVB broadcasting specification has been generally designed to support the provision of video, audio and data services. Nevertheless, some additional specification is required in order to cover the large variety of Data Broadcasting Services such as value added services, software downloading, file distribution, broadcast multimedia and other business and consumer services which could be provided by the DVB/MPEG-2 multiplex.

The objective of this document is to specify the commercial requirements of end users, service providers and network operators for Data Broadcasting Services using the DVB system. These requirements represent guidelines for Data Broadcasting Services which should facilitate interoperability and compatibility of different systems, whilst promoting the positive competitive market forces which will accelerate the technological development of such services. It is not the intention of this document to define a standardised DVB end-user terminal for Data Broadcasting Services. However, common protocols and interfaces may need to be specified in addition to those already defined within the DVB.

Security systems for data services depending on customer requirements should be available for normal protection and high confidentiality. In addition, some applications of Data Broadcasting Services may require a return channel.

The receiving equipment for DVB data reception could be a normal IRD extended with the minimum hardware and/or software to support data reception or could be integrated in a single computer add-on device or peripheral.

2. Definition

Data Broadcasting Services are defined as those which are independent of or complementary to TV and radio services or other DVB services as defined elsewhere. The data rates of Data Broadcasting Service might range from a few kbps to the full capacity of the transport stream.

3. Commercial Requirements

Technical specifications need to be defined for the following commercial requirements which cover the interests of end users, service providers and network operators.

Technical recommendations shall not prevent service providers and network operators from taking into account all issues covered by local and national laws, especially those related to the protection of personal data.

1. Any technical specification shall be based on the layered architecture of the ISO/OSI reference model and shall identify the layer it is addressing.

2. Where appropriate, existing international standards and relevant aspects of DVB-MPEG 2 specifications shall be adopted.

3. There shall be no modifications to the existing DVB standards, which are only to be complemented when necessary. These complementary specifications should be based on existing DVB and MPEG 2 systems mechanisms. Where appropriate, other existing international standards and relevant aspects of DVB and MPEG 2 shall be adopted.

4. Only the OSI layers 4 and below shall be considered. The DVB data transmissions should be transparent to OSI layer 5 and above.

5. The DVB Data Broadcasting recommendation must be application-neutral.

6. The specifications must be transmission media independent.

7. The specifications must be media-content independent.

8. The technical recommendation shall neither limit the kind of Data Broadcasting Services nor the respective data rates involved. It shall rather, if necessary, provide recommendations for services within ranges of data rates through the definition of a limited number of profiles.

9. The Technical Module should identify the relevant hardware and software interfaces compatible with the different proposed profiles.

10. DVB data transmission should be able to accommodate transmissions coming from other networks (e.g. PDH, SDH, ATM and ISDN).

11. The technical specifications shall leave open whether the user terminal for Data Broadcasting Services will be combined with a digital IRD or a single computer add-on device or peripheral or any other user terminal or interface.

12. The specification (i.e. data structures and containers) must be future-proof, so that future network or service extensions will not interfere with existing services.

13. The specification shall enable the use of any kind of scrambling and conditional access systems at the transport layer to meet customer requirements for confidentiality.

14. In defining the data container, the use by service providers of other additional scrambling and conditional access systems should not be precluded.

15. For interactive data applications requiring a return channel the "Commercial Requirements for Asymmetric Interactive Services Supporting Broadcast to the Home with Narrow Band Return Channels" of the DVB-ISCM group and the corresponding technical specification of the DVB-SIS group shall be applied accordingly.

16. The recommendations shall allow easy identification and selection of data only channels, using standard DVB mechanisms and if the profiles are used, the specifications shall allow the end-user terminal to identify whether the incoming Data Broadcasting profile is supported.

17. The DVB SI tables, where appropriate, shall be extended to accommodate information on the Data Broadcasting Services available and service operating parameters.

18. It must be possible to implement Data Broadcasting Services in an IRD (at least for some of the profiles) at virtually no extra cost excepting for the physical data ports themselves.

19. Allowance must be made in the specifications for data ports at minimal cost.

20. The proposed technical specifications shall be available as soon as possible and preferably by March 1996.

21. A need has been identified to study recovery from transmission errors at packet level. The methods can be strictly limited to the application level, but can also take into account the information available at the transport level. The DBS ad-hoc group requests the Technical Module to identify the existing elements at the transport level that could be employed to recover transmission errors at the application level.