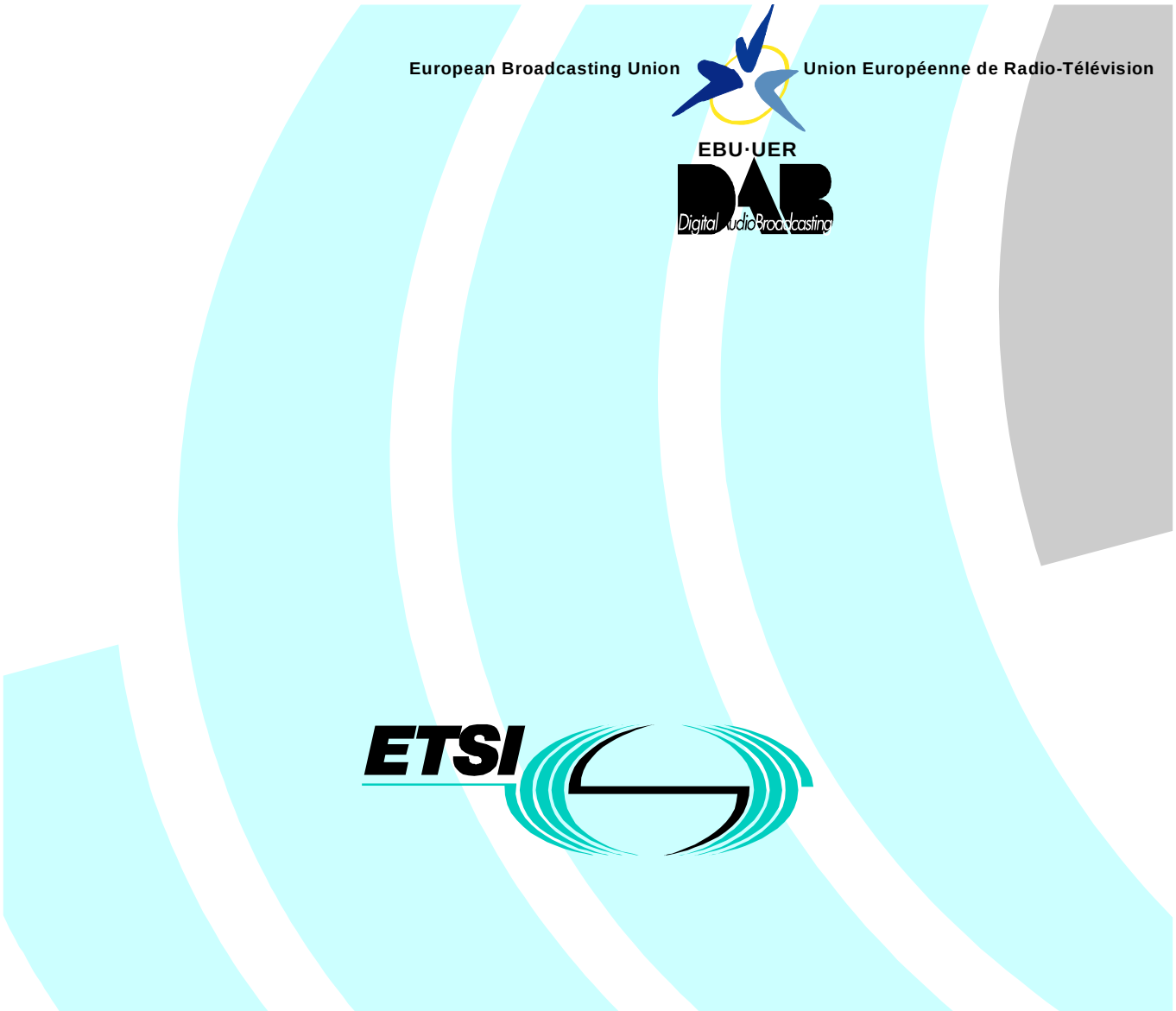
ETSI TS 101 759 V1.1.1 (2000-09)

Technical Specification

Digital Audio Broadcasting (DAB); Data Broadcasting - Transparent Data Channel



Reference
DTS/JTC-DAB-25

Keywords
audio, broadcasting, DAB, data, digital,
packet mode

ETSI

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

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

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

Contents

Intellectual Property Rights

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

Foreword

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

NOTE 1: 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 Audio Broadcasting (DAB) Eureka Project 147

The Eureka Project 147 was established in 1987, with funding from the European Commission, to develop a system for the broadcasting of audio and data to fixed, portable or mobile receivers. Their work resulted in the publication of European Standard, EN 300 401 [1], for DAB (see note 2) which now has worldwide acceptance. The members of the Eureka Project 147 are drawn from broadcasting organizations and telecommunication providers together with companies from the professional and consumer electronics industry.

NOTE 2: DAB is a registered trademark owned by one of the Eureka Project 147 partners.

Introduction

The Transparent Data Channel (TDC) specification provides DAB with the ability to transport simple data streams using a variety DAB data channels. In a simple stream, data bytes are processed sequentially by the receiver in the order in which they are received but there is no implied beginning or end to the data. Any "framing" of data within the stream is determined purely by the application protocol that is carried by the stream.

Three forms of the Transparent Data Channel are specified here:

- TDC in a packet mode service component;
- TDC in a stream mode service component;
- TDC in an audio service component carried as X-PAD.

The Transparent Data Channel complements the MOT specification by allowing DAB to transport data in the form of *streams* as well as *files*. It is intended that applications that rely on simple stream transport, such as the TPEG and DGPS protocols, can be supported in DAB simply by specifying the use of the Transparent Data Channel for data transport.

1 Scope

The present document specifies how to transport data transparently within the Digital Audio Broadcasting (DAB) system forming a Transparent Data Channel and gives guidelines for its use.

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] ETSI EN 300 401: "Radio Broadcasting Systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers".

[2] ETSI TS 101 756: "Digital Audio Broadcasting (DAB); Registered Tables".

[3] ETSI TR 101 496: "Digital Audio Broadcasting (DAB) Guidelines and rules for implementation and operation".

[4] ETSI EN 301 234: "Digital Audio Broadcasting (DAB); Multimedia Object Transfer (MOT) protocol".

3 Abbreviations and syntax specification

3.1 Abbreviations

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

CCITT	International Consultative Committee for Telegraph and Telephone
CRC	Cyclic Redundancy Check
DAB	Digital Audio Broadcasting
FIC	Fast Information Channel
F-PAD	Fixed Programme Associated Data
MOT	Multimedia Object Transfer
MSC	Main Service Channel
PAD	Programme Associated Data
TCP	Transmission Control Protocol
TDC	Transparent Data Channel
TPEG	Transport Protocol Experts Group
X-PAD	Extended Programme Associated Data

3.2 Syntax specification

The specifications of syntax that appear in the present document are written using a form of pseudo-code that is similar to the procedural language 'C'; this provides for easy specification of loops and conditional data structures. Within these specifications, the type of individual data fields is expressed using the mnemonics given in table 1-1.

Table 1-1: Data type mnemonics for syntax specification

Mnemonic	Description
uimsbf	Unsigned integer, most significant bit first
bslbf	Bit string, left bit first
rpchof	Remainder polynomial coefficients, high order first

4 Transparent Data Channel specification

The MOT protocol allows transportation of objects – this is achieved either by using simple MOT transport for non-ordered data objects or by using a data carousel for a managed set of data files. As a complement to this, the Transparent Data Channel allows simple stream data to be delivered to support stream based applications (for example TPEG).

The TDC specification described below is based on the mechanisms that are provided by [1] for data delivery:

- Packet mode service component.
- Data in an audio service Component using X-PAD.

The implementation of these basic mechanisms is explained in more details in [3].

4.1 TDC in a packet mode service component

TDC data carried in a packet mode service component may be organized either with or without the MSC Data Group structure defined by EN 300 401 [1].

The use of data groups for establishing a transparent data channel provides:

- Error control by using data group repetition.
- The ability to guarantee that "chunks" of data from the stream are delivered to the receiver (or lost) as a coherent sets of data.
- Optional addressing of end users.

Where the features above are not required, not using data groups for the TDC provides:

- Low overhead.
- Low latency.
- Simple processing requirements.

The two forms of the TDC in a packet mode service component are specified in the following paragraphs.

4.1.1 TDC in a packet mode service component without data groups

Packet mode sub-channels use a simple packet protocol to allow a number of separate data services to be multiplexed into a single DAB sub-channel. Carriage of simple stream data in such a channel is implemented by sending packets containing the data *when there is data to send*. The structure of DAB packets is shown in table 2-1.

Table 2-1: Structure of a DAB packet

Syntax	Size	Type
DAB_packet() {		
packet_length	**2 bits**	**uimsbf**
continuity_index	**2 bits**	**uimsbf**
first_ flag	**1 bit**	**bslbf**
last_flag	**1 bit**	**bslbf**
packet_address	**10 bits**	**uimsbf**
command_flag	**1 bit**	**bslbf**
useful_data_length	**7 bits**	**uimsbf**
for (i=0;i<useful_data_length;i++) {		
packet_data_byte	**8 bits**	**uimsbf**
}		
for (i=0;i<N;i++) {		
padding_byte	**8 bits**	**uimsbf**
}		
packet_CRC	**16 bits**	**rpchof**
}		

packet_length: This field can take one of four values to indicate the overall size of the packet in multiples of 24 bytes. The maximum length of the payload of each packet is equal to the overall packet length minus the 5 bytes of packet overhead. The overall packet length may be determined from the *packet_length* field by the formula (*packet_length* + 1) * 24, thus a *packet_length* of 0 indicates an overall packet length of 24 bytes and a *packet_length* of 3 indicates an overall packet length of 96 bytes.

continuity_index: The *continuity_index* is incremented by one for each successive packet that is sent **for a particular *packet_address***.

first_flag: The *first_flag* is used to indicate (when set to 1) that the packet is the first packet in a series of packets that are to be assembled into a DAB "Data Group".

last_flag: The *last_flag* is used to indicate (when set to 1) that the packet is the last packet in a series of packets that are to be assembled into a DAB "Data Group".

packet_address: The *packet_address* is a 10 bit number that identifies packets for a particular data service component within the multiplex of components for a packet mode sub-channel.

useful_data_length: The *useful_data_length* field identifies the number of bytes within the payload of the packet that are to be considered to be carrying useful data.

packet_data_byte: These fields carry the useful payload for each individual packet.

padding_byte: The *padding_bytes* are used to fill the remainder of the packet after the *packet_data_bytes* and shall be set to 0.

packet_CRC: The *packet_CRC* field is calculated, according to the CCITT CRC-16 polynomial, over the entire packet, with the CRC register initialized to all 1s and the resulting CRC inverted.

Within a packet mode sub-channel, packets may be assembled into larger, variable length datagrams known as "Data Groups". The assembly of packets into Data Groups is controlled by the first and last flags, as shown in table 2-2.

Table 2-2: Interpretation of first and last flags

first_flag	last_flag	Interpretation
0	0	The packet is an intermediate packet in a series of packets
0	1	The packet is the last packet in a series of packets
1	0	The packet is the first packet in a series of packets
1	1	The packet is the one and only packet

In order to carry stream data in this very simple way, the data is sent in packets with the first_flag and the last_flag both set to zero, thus indicating that the data carried by the packets is simply an intermediate part of a stream with no specific start or end. Marking the packets in this way ensures that no receiver will ever attempt to incorrectly assemble asynchronous stream data into data groups.

The use of Data Groups in TDC depends upon the DG flag in FIG 0/3 (see [1]). When set to '0', Data Groups are used, when set to '1' no Data Groups are used.

4.1.2 TDC in a packet mode service component with Data Groups

In order to carry stream data, the data is put into a Data Group, which in turn is sent in packets.

Table 2-3: Structure of a DAB Data Group

Syntax	Size	Type
DAB_Data_Group() {		
MSC_data_group_header() {		
extension_flag	**1**	**bslbf**
CRC_flag	**1**	**bslbf**
segment_flag	**1**	**bslbf**
user_access_flag	**1**	**bslbf**
data_group_type	**4**	**uimsbf**
continuity_index	**4**	**uimsbf**
repetition_index	**4**	**uimsbf**
if (extension_flag == 1) {		
extension_field	**16**	**bslbf**
}		
}		
session_header() {		
if (segment_flag == 1) {		
last	**1**	**bslbf**
segment_number	**15**	**uimsbf**
}		
if (user_access_flag == 1) {		
user_access_field () {		
rfa	**3**	**bslbf**
tranport_id_flag	**1**	**bslbf**

length_indicator	**4**	**uimsbf**
if (transport_id_flag == 1) {		
transport_id	**16**	**uimsbf**
}		
end_user_address_field() {		
for (n=0;n<length_indicator-2;n++) {		
end_user_address_byte	**8**	**uimsbf**
}		
}		
}		
}		
MSC_data_group_data_field() {		
for (i=0;i<data_group_length;i++) {		
data_group_data_byte	**8 bits**	**uimsbf**
}		
}		
if (CRC_flag==1) {		
MSC_data_group_CRC	**16 bits**	**rpchof**
}		
}		

extension_flag: The *extension_flag* indicates (when set to 1) that the *extension_field* is present.

crc_flag: The crc_flag indicates (when set to 1) that the *MSC_data_group_CRC* is present.

segment_flag: The *segment_flag* indicates (when set to 1) that the *segment_number* and last Flag in the session_header are present.

user_access_flag: The *user_access_flag* indicates (when set to 1) that the *user_access_field* in the *session_header* is present.

data_group_type: For TDC the *data_group_type* shall be set to 0.

continuity_index: The *continuity_index* shall be incremented each time a data group with a content different from that of the preceding data group is transmitted.

repetition_index: The *repetition_index* shall signal the remaining number of repetitions of a data group with the same content. Exceptionally, the code '1111' shall be used to signal that the repetition continues for an undefined period.

extension_field: The *extension_field* is reserved for future use in TDC.

last: The *last* flag shall indicate that the current data group is the last in a sequence of data groups with the same *transport_id* that carry the data for a coherent "chunk" of data from the TDC.

segment_number: The *segment_number* shall indicate the segment number of the current data group when a coherent "chunk" of data (identified by the *transport_id*) is being transmitted that is longer than a single data group.

rfa: Reserved for future additions. Shall be set to '0'.

transport_id_flag: For TDC, the *transport_id_flag* shall indicate (when set to 1) that the *transport_id* field is present.

length_indicator: The *length_indicator* shall indicate the length n in bytes of the transport_id and the *end_user_address_field*.

transport_id: For TDC, the *transport_id* (if needed) shall be identical for all different segments forming a specific "chunk". It shall be changed for each coherent "chunk" of data.

end_user_address_byte: The *end_user_address_bytes* shall be used to distinguish between different TDC data streams for different terminals. It's use depends upon the application (ApplicationId in FIG0/13) and is not subject to this specification.

MSC_data_group_data_byte: The *MSC_data_group_data_bytes* shall contain the data for the transparent data channel.

MSC_data_group_CRC: The *MSC_data_group_CRC* field is calculated, according to the CCITT CRC-16 polynomial, over the *MSC_data_group_header*, the *session_header* and the *MSC_data_group_data_field*, with the CRC register initialized to all 1s and the resulting CRC inverted.

4.2 TDC in a stream mode service component

In a stream mode service component, all the data for the sub-channel is assumed to be part of the stream. This means that a Transparent Data Channel carried in a stream mode service component is an essentially synchronous (i.e. fixed bit rate) stream.

4.3 TDC in an audio service component using X-PAD

As with the TDC in a packet mode service component, the stream of TDC data in X-PAD may be considered to be arbitrarily long and without an identifiable beginning or end. Thus, the only X-PAD application type that is required is the "*Transparent data channel*" application type (with number 23, see [1], subclause 7.4.3) to identify the X-PAD data subfields carrying TDC data.

NOTE: The same transport mechanism (as defined by the present document for TDC in X-PAD) may be applied also for other X-PAD application types. For the application types, see [2].

Two forms of X-PAD are available and the precise interpretation of the X-PAD sub-field for the TDC depends on type of X-PAD that is being used. In "short" X-PAD a single field of either 3 or 4 bytes is present in each audio frame. In "variable length" X-PAD up to four X-PAD sub-fields of up to 48 bytes each can be present in each audio frame. The two cases are described in detail below.

4.3.1 Using short X-PAD for the TDC

Table 2-4 shows the structure of short X-PAD but it should be noted that the byte order of the X-PAD fields is reversed on transmission. See [1] for more details.

Table 2-4: Structure of X-PAD when using short X-PAD

Syntax	Size	Type	Value
X-PAD_field() {			
if (contents_indicator_flag == 1) {			
application_type	**8 bits**	**uimsbf**	
X-PAD_subfield	**24 bits**	**uimsbf**	
} else {			
X-PAD_subfield	**32 bits**	**uimsbf**	
}			
}			

contents_indicator_flag: This flag is identical with the CI flag in F-PAD (see [1]) and is used to indicate the presence of a contents_indicator field in the same DAB audio frame.

application_type: This field determines the intended application for the X-PAD_subfield. The application type for TDC is given in [2].

X-PAD_subfield: This field carries the stream data for the TDC.

If the useful data bytes cannot fill the X-PAD sub-field completely, the rest of the sub-field is filled with 0xff (byte stuffing). To enable a receiver to differentiate between a useful byte with value 0xff from a stuffing byte, each useful byte 0xff is inserted in the X-PAD_subfield as the pair of bytes 0xfe, 0x01, and each useful byte 0xfe is inserted as the pair of bytes 0xfe, 0x00. All other useful bytes are inserted transparently, i.e. without any change.

4.3.2 Using variable length X-PAD for the TDC

Table 2-5 shows the structure of variable length X-PAD but it should be noted that the byte order of the X-PAD fields is reversed on transmission. It also should be noted, that in addition to the table, a complete X-PAD field may comprise a single data subfield (provided the X-PAD field is containing a continuation of the X-PAD data application of the same type as that carried in the last X-PAD data subfield of the previous DAB audio frame). For details see [1] and [3].

Table 2-5: Structure of X-PAD when using variable length X-PAD

Syntax	Size	Type	Value
X-PAD_field() {			
if (contents_indicator_flag == 1) {			
for (i=0;i<N1;i++) {			
X-PAD_subfield_length	**3 bits**	**bslbf**	
application_type	**5 bits**	**uimsbf**	
if (application_type == 31) {			
application_type_extension	**8 bits**	**uimsbf**	
}			
}			
}			
for (i=0;i<N1;i++) {			
for (j=0;i<X-PAD_subfield_length(N1);j++) {			
X-PAD_data_byte	**8 bits**	**uimsbf**	
}			
}			
}			

contents_indicator_flag: This flag is identical to the CI flag in F-PAD (see [1]) and is used to indicate the presence of a contents_indicator field in the same DAB audio frame.

X-PAD_subfield_length: This field determines the length the X-PAD_subfield.

application_type: This field determines the intended application for the X-PAD_subfield. The application type for TDC is given in [2].

application_type_extension: This field determines the intended application for the X-PAD_subfield when the application type is set to 31.

X-PAD_data_byte: These fields carry the data for the X-PAD sub-field.

If the useful data bytes cannot fill the X-PAD sub-field completely, the rest of the sub-field is filled with 0xff (byte stuffing). To enable a receiver to differentiate between a useful byte with value 0xff from a stuffing byte, each useful byte 0xff is inserted in the X-PAD_subfield as the pair of bytes 0xfe, 0x01, and each useful byte 0xfe is inserted as the pair of bytes 0xfe, 0x00. All other useful bytes are inserted transparently, i.e. without any change.

5 Guidelines for the use of the Transparent Data Channel

Simple stream data formats are extremely common, not least because the bulk of Internet communication uses stream transport in the form of TCP/IP. When defining applications to use the DAB Transparent Data Channel, however, it is important to recognize that there are differences between streams implemented using TCP/IP and broadcast streams.

In TCP/IP, a series of acknowledgements and time-outs is used to detect and correct any errors that are introduced into the data for the stream. This means that reliable delivery of the data is guaranteed but this feature relies on the fact that the Internet allows bi-directional communication. In a broadcast environment there is no possibility for a receiver to request re-transmission of data that has been corrupted, and so it is not possible to guarantee reliable delivery of data using the DAB Transparent Data Channel.

Because reliable delivery of data cannot be assumed, applications that use the Transparent Data Channel for data transport **must ensure that they are tolerant to errors** in the data. This applies to corrupted data, loss of data and additional erroneous data.

The TDC specification is based on two mechanisms provided by [1] for data delivery, either delivery in a packet mode service component or in an audio service component using X-PAD. So the specification enables each service provider to whom only one single type of service component is assigned by regulation authorities, to take the appropriate mechanism.

In case of an audio service component using X-PAD the TDC specification offers a single, but very simple delivery mechanism.

In case of a packet mode service component there are three options to be chosen from. Which of the options will be chosen has to be determined by the definition of the application, depending on its requirements in terms of performance and complexity, particularly in terms of storage capacity and error robustness. The options are:

- The TDC in a packet mode component without data groups is a very simple delivery mechanism, as simple as the TDC in an audio service component.

- If the TDC is using data groups in packet mode, but without any session header, TDC provides the possibility to retransmit the same data group one or several times, so that a receiver can cope with bit errors in one or the other data group. It is recommended that the data group is restricted to a size of 8 kB. This ensures, that under normal reception conditions the error probability for a data group can be kept low and that further on, the storage capacity is in line with a more advanced receiver design. The same restriction applies when a session header is used with end user address field only.

- If the TDC is used with data groups including session headers, it is capable to deliver "chunks" of data. As with a multimedia object (see MOT protocol [4]) a chunk is segmented in data groups of equal size. Each segment of a chunk is identified by its segment number. All segments of a particular chunk are identified by the same TransportId.

If an application uses the TDC delivery in packet mode, it should use the method with the lowest complexity that complies with the required performance.

The TDC delivery of chunks is included in the present document for completeness and with the intention to provide future proof mechanisms.

History

Document history		
V1.1.1	September 2000	Publication