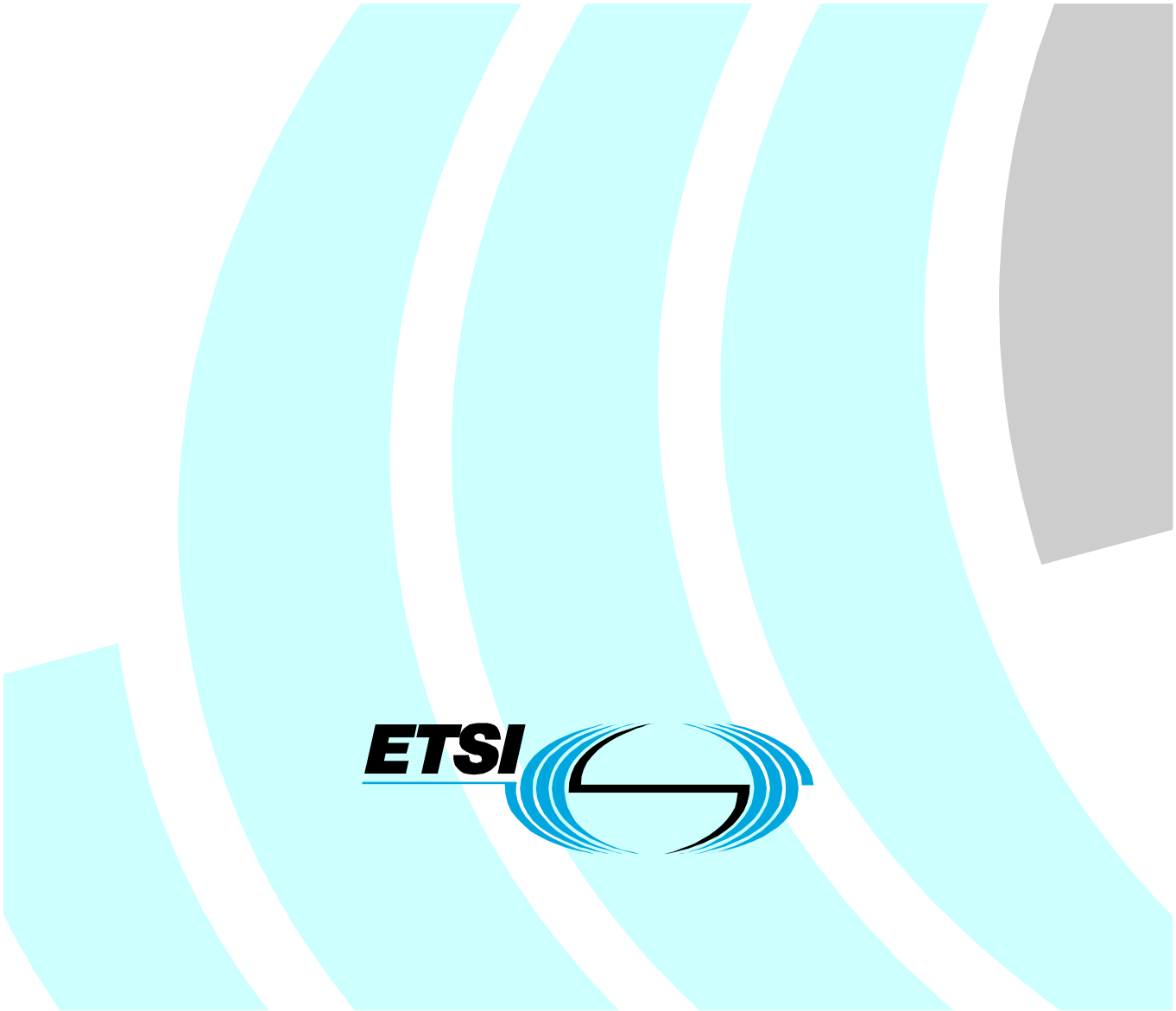
Draft ETSI TS 1XX XXX V1.0.3 (2004-mm)

Technical Specification

Digital Audio Broadcasting (DAB); Transportation Specification for DAB Electronic Programme Guide (EPG)



Reference
<Workitem>

Keywords
audio, broadcasting, DAB, digital, EPG

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

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status.
Information on the current status of this and other ETSI documents is available at
http://portal.etsi.org/tb/status/status.asp

If you find errors in the present document, please send your comment to one of the following services:
http://portal.etsi.org/chaircor/ETSI_support.asp

Copyright Notification

Contents

Intellectual Property Rights

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

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

Foreword

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

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

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

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 [2], for DAB (see note) 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: DAB is a registered trademark owned by one of the Eureka Project 147 partners.

Introduction

This specification defines how the EPG data will be transported, compressed and profiled such that a good user experience can be achieved using limited broadcast capacity. Using a combination of EPG profiles it is possible that a range of features could be supported in receivers, including:

- The display of schedules at varying levels of detail for programmes from a range of services
- The display of schedules, with programmes and events ordered into particular groups
- Navigation and selection of services and programmes
- Searching through current and future programme listings.

- Timed recording of programmes
- Timed recording of groups of programmes
- Automatic recording of groups of programmes and themed or similar programming
- Accurate timed recording of programmes and events using PNUM signalling.

1. Scope

The present document defines how the XML schema data model for an Electronic Programme Guide (EPG) for Eureka-147 Digital Audio Broadcasting (DAB) [1] should be compressed, profiled and broadcast. Within the present document the term "DAB" is used to refer to the Eureka-147 Digital Audio Broadcasting standard [3].

2. References

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

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

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

[1] ETSI TS 102 818 "XML Specification for DAB Electronic Programme Guide (EPG)" , version 1.1.5 (DRAFT), August 2004

[2] ETSI EN 301 234: "Digital Audio Broadcasting (DAB); Multimedia Object Transfer (MOT) Protocol", version 2.1.1 (DRAFT), August 2004

[3] ETSI EN 300 401: "Radio broadcasting systems; Digital Audio Broadcasting (DAB) to mobile, portable and fixed receivers"

[4] ISO 10646-1: "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane".

3. Definitions, symbols and abbreviations

3.1 Definitions

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

Conditional Access (CA): mechanism by which the user access to service components can be restricted

data service: service which comprises a non-audio primary service component and optionally secondary service components

ensemble: transmitted signal, comprising a set of regularly and closely-spaced orthogonal carriers

NOTE: The ensemble is the entity that is received and processed. In general, it contains audio and data services.

Ensemble Identifier (EId): unique 16-bit code, allocated to an ensemble and intended to allow unambiguous worldwide identification of that ensemble

Extended Programme Associated Data (X-PAD): extended part of the PAD carried towards the end of the DAB audio frame, immediately before the Scale Factor Cyclic Redundancy Check (CRC). Its length is variable

Programme Associated Data (PAD): information that is related to the audio data in terms of contents and synchronization

NOTE: The PAD field is located at the end of the DAB audio frame.

secondary service component: In the case where a service contains more than the primary service component, the additional service components are secondary service components.

service: In the present document the term "service" is used to refer to a "radio station" such as BBC Radio 4 or Oneword. In strict DAB terms this is actually a service component of a service.

service component: part of a service which carries either audio (including PAD) or data

NOTE: The service components of a given service are linked together by the Multiplex Configuration Information. Each service component is carried either in a sub-channel or in the Fast Information Data Channel.

Service Identifier (SId): 16- or 32-bit code used to identify a particular service

3.2 Abbreviations

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

CA	Conditional Access
CRID	Content Reference ID
DAB	Digital Audio Broadcasting
DNS	Domain Name Server
DRM	Digital Radio Mondiale
ECC	Ensemble Country Code
EId	Ensemble Identifier
EPG	Electronic Programme Guide
GI	Group Information
IANA	Internet Assigned Numbers Authority
ISO	International Organization for Standardization
MIME	Multipurpose Internet Mail Extensions
MOT	Multimedia Object Transfer
PAD	Programme Associated Data
PI	Programme Information
PNG	Portable Network Graphics
SCIdS	Service Component Identifier within the Service
SDARS	Satellite Digital Audio Radios
SI	Service Information
SId	Service Identifier
SMS	Short Messaging Service
UATy	User Application Type
URI	Uniform Resource Identifier
URL	Uniform Resource Location
UTC	Co-ordinated Universal Time
WAP	Wireless Access Protocol
WBMP	Wireless Bitmap
WWW	World Wide Web
XML	Extensible Markup Language
X-PAD	eXtended Programme Associated Data

4. Encoding

There will be no transmission of the raw EPG XML data [1]. The XML specification should still be used to construct the valid information; this binary specification is then used to encode this information in a compact binary form.

The binary encoding described here uses a tag-length-value encoding. Each element or attribute is encoded using a unique tag value, a length value (indicating the length of the data contained within this element or attribute) and the actual data value/s. This enables receivers to easily skip elements that are not wanted or were undefined.

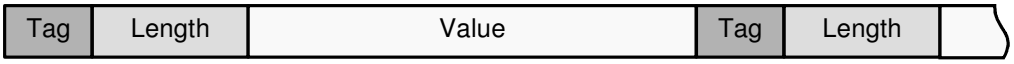


Figure 1 – tag-length-value encoding scheme

XML elements are all encoded in these binary structures as described later in this section. Attributes are coded in a similar way (see section 4.4). The hierarchical nature of the EPG XML is generally preserved in these binary structures, but the structure is not necessarily identical. Various common data types have been assigned efficient binary encodings as described in section 4.7. For an example of a binary encoded XML object, please see Annex C (informative): Binary encoding example.

Note that although the length of certain data types can be worked out from their encoding, there shall still be a length field in the attribute encoding (see section 4.4).

4.1 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.

Mnemonic	Description
Uimsb	Unsigned integer, most significant bit first

Table 1 - Data type mnemonics for syntax specification

4.2 Binary objects

The basic binary objects defined by this specification are defined in Table 2. Each binary object carries a single top level element and shall be carried within a single MOT object.

Syntax	Size	Type
binary_object() { **top_level_element()**		

Table 2 - Structure of a binary object

top_level_element: A top level element as defined in section 4.3.1.

4.3 Elements

All elements use basically the same encoding, as defined here.

Syntax	Size	Type
element() {		
element_tag	8 bits	uimsbf
element_length	8 bits	uimsbf
If (element_length == 0xFE) {		
extended_element_length	16 bits	uimsbf
}		
If (element_length == 0xFF) {		
extended_element_length	24 bits	uimsbf
}		
for (i=0; I<element_length *or* extended_element_length; i++) {		
element_data_byte	8 bits	uimsbf
}		
}		

Table 3 - Structure of an element

element_tag: This byte identifies the element. The tag uniquely identifies the element - i.e. there is a one to one mapping between a tag and an element. If new elements are required in the future then they will use new tag values. The possible values are defined in Annex D (normative): Element tags. Elements with tags that are not defined here are reserved for future use; the tags and their associated content shall not be processed by receivers.

element_length: This field indicates the number of data bytes contained in this element, i.e. the number of bytes that follow the length byte up to the end of the element. The range of this is `0x00` to `0xFD` (i.e. 0 to 253). If this value is either `0xFE` or `0xFF` then the additional *extended_element_length* field defines the element length.

extended_element_length: When used, this field indicates the number of data bytes contained in this element, i.e. the number of bytes that follow the last extended length byte up to the end of the element.

element_data_byte: These bytes contain the element's attributes, CDATA (i.e. string data) and child elements. They shall be encoded in the following order:

1. Attributes
2. Child elements
3. CDATA content

4.3.1 Top-level elements

There are two top-level elements defined in this specification; *epg* and *serviceInformation*. A top-level element shall be carried within a binary object (see section 4.2) and it shall be the only element (apart from its nested children) in that object. The possible values of the *element_tag* for top-level elements are defined in Table 4. Top-level elements with tags that are not defined here are reserved for future use; these tags and their associated content shall not be processed by receivers.

Element	Tag
epg	0x02
serviceInformation	0x03

Table 4 - Top-level element tags

As well as the appropriate elements defined by the EPG XML specification the top-level elements may also, optionally, contain a string token table (see section 4.9) and a default dabID (see section 4.10). If present, these elements shall be the first elements to occur in the top-level element after the attributes.

A top-level element is encoded in the same way as a normal element (see section 4.3) with the exception that the *element_data_bytes* shall be encoded in the following order:

1. Attributes
2. String token table (if present)
3. Default dabID (if present)
4. Child elements
5. CDATA content

4.4 Attributes

All attributes use basically the same encoding, as defined here.

Syntax	Size	Type
attribute() {		
attribute_tag	8 bits	uimsbf
attribute_length	8 bits	uimsbf
if (attribute_length == 0xFE) {		
extended_attribute_length	16 bits	uimsbf
}		
if (attribute_length == 0xFF) {		
extended_attribute_length	24 bits	uimsbf
}		

for (i=0; i<attribute_length *or* extended_attribute_length; i++) {		
attribute_data_byte	8 bits	uimsbf
}		
}		

Table 5 - Structure of an attribute

attribute_tag: This byte uniquely identifies the attribute within the parent element. The possible values are defined in Annex E (normative): Attribute tags.

Attributes with tags that are not defined here are reserved for future use and shall not be processed by receivers.

attribute_length: This field indicates the number of data bytes contained in this attribute, i.e. the number of bytes that follow the length byte up to the end of the attribute. The range of this is `0x00` to `0xFD` (i.e. 0 to 253). If this value is either `0xFE` or `0xFF` then the additional *extended_attribute_length* field defines the attribute length.

extended_attribute_length: When used, this field indicates the number of data bytes contained in this attribute, i.e. the number of bytes that follow the last extended length byte up to the end of the attribute.

attribute_data_byte: These bytes contain either a string (section 4.5.1) or an enumerated data value (see section 4.6) or a common data type (section 4.7).

4.4.1 Default attributes

Where an attribute has a default value there is no need for it to be present in the binary encoding as the receiver shall always automatically use the default value.

4.5 CDATA and strings

All CDATA or text strings apart from textual attributes shall be encoded as detailed in Table 6.

Syntax	Size	Type
cdata() {		
cdata_tag	8 bits	uimsbf
cdata_length	8 bits	uimsbf
If (cdata_length == 0xFE) {		
extended_cdata_length	16 bits	uimsbf
}		
If (cdata_length == 0xFF) {		

extended_cdata_length	24 bits	uimsbf
}		
for (i=0; i<cdata_length *or* extended_cdata_length; i++) {		
cdata_data_byte	8 bits	uimsbf
}		
}		

Table 6 - Structure of a CDATA element

cdata_tag: This shall always be `0x01`.

cdata_length: This field indicates the number of data bytes contained in this string. The range of this is `0x00` to `0xFD` (i.e. 0 to 253). If this value is either `0xFE` or `0xFF` then the additional *extended_cdata_length* field defines the attribute length.

extended_cdata_length: When used, this field indicates the number of data bytes contained in this string.

cdata_data_byte: These bytes contain the characters for this CDATA element.

Note: Attributes with text values shall not be encoded in this format and instead are encoded as an attribute (see section 4.4) with the *attribute_data_bytes* being the character data bytes only.

4.5.1 Character sets

All CDATA and other strings shall use ISO 10646-1 [4] with UTF-8 encoding. The characters `0xE000` to `0xF8FF` shall be ignored (i.e., the terminal shall neither try to display these characters nor shall it display any replacement characters).

4.6 Enumerated data values

Those attributes that are enumerated types are encoded by a single byte, the value of this byte is specific to that particular attribute and is defined in the appropriate table contained in Annex F (normative): Enumerated types.

4.7 Common data types

Various common data types within the EPG specification [1] have been identified and the specific encodings to be used are defined in this section.

4.7.1 Date and time

All elements defined as timePointType or basicTimePointType are encoded as follows.

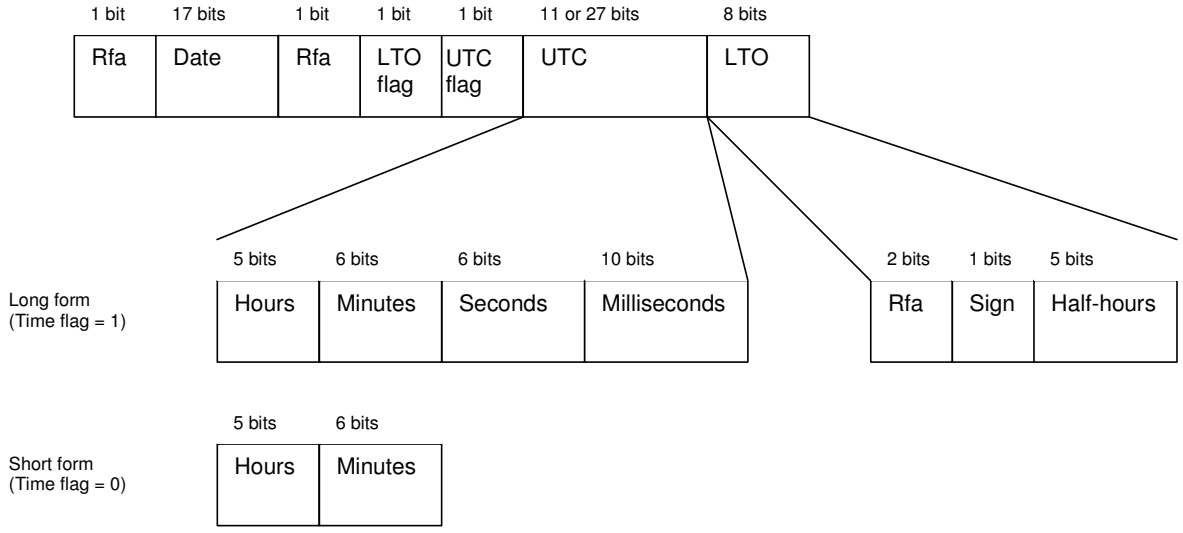


Figure 2 - Date and time encoding (LTO flag == 1)

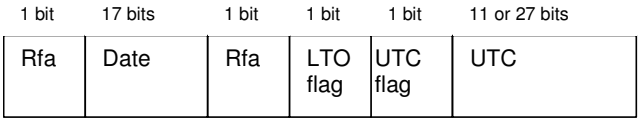


Figure 3 - Date and time encoding (LTO flag == 0)

Rfa: This 1-bit field shall be reserved for future additions. The bit shall be set to zero for the currently specified definition of this field.

NOTE: Receivers shall ignore this bit.

Date: This 17-bit binary number shall define the current date according to the Modified Julian Date (MJD) coding strategy [3]. This number increments daily at 0000 UTC and extends over the range 0-99999. As an example MJD 50000 corresponds to October 10th, 1995.

Rfa: This 1-bit field shall be reserved for future additions. The bit shall be set to zero for the currently specified definition of this field.

NOTE: Receivers shall ignore this bit.

LTO flag: This 1-bit field indicates whether the LTO field (see below) is present or not, as follows:

0: LTO not present, time is in UTC

1: LTO present, local time is UTC plus LTO

UTC flag: This 1-bit field indicates whether the UTC (see below) takes the short form or the long form, as follows:

0: UTC short form

1: UTC long form

UTC (Co-ordinated Universal Time): Two forms are available depending upon the state of the UTC flag. They are defined as follows:

Short form: This 11-bit field contains two sub-fields, coded as unsigned binary numbers. The first sub-field is a 5-bit field which shall define the hours and the other sub-field is a 6-bit field which shall define the minutes.

Long form: In addition to the hours and minutes fields defined in the short form, this 27-bit field shall contain two further sub-fields, both of which shall be encoded as unsigned binary numbers. The first is a 6-bit field which shall define the seconds and the other is a 10-bit field which shall define the milliseconds.

LTO (Local Time Offset): This 8-bit field shall give the Local Time Offset (LTO) for the time given. It is only present if the LTO flag is set to 1. The first two bits are reserved for future additions, they shall be set to zero for the currently specified definition and shall be ignored by receivers. The next bit shall give the sense of the LTO, as follows:

0: positive offset

1: negative offset

The final 5 bits define the offset in multiples of half-hours in the range –12 hours to +12 hours.

For example, a programme broadcast at 05:00 in the UK during Daylight Savings time would have a UTC of 04:00 and an LTO of +1 hour.

4.7.2 Duration

All elements defined as durationType and basicDurationType are encoded as a 16-bit unsigned integer, representing the duration in seconds from 0 to 65,535 (just over 18 hours).

4.7.3 CRIDs

All elements defined as CRIDType are encoded as a string attribute (see section 4.4).

4.7.4 Short CRIDs

All elements defined as shortCRIDType are encoded as a 24-bit unsigned integer.

4.7.5 Genres

All elements that are defined as genreType shall be encoded as follows. Only the href attribute is encoded (the CS and the levels) together with the (optional) type attribute. The name and definition elements are not encoded.

Genre href

The href attribute of the genre element shall be encoded as follows:

4 bits	4 bits	0 or 8 bits	0 or 8 bits	0 or 8 bits
Rfu	CS	Level 1	Level 2	Level 3

Figure 4 - Genre href encoding

Rfu: This 4-bit field shall be reserved for future use of the remainder of the structure. The four bits shall be set to zero for the currently specified definition of this field.

CS field: This 4-bit field shall indicate which classification scheme (CS, e.g. the 1 of “1.2.3.4”) this genre is a member of, as follows:

0: Undefined. Genres with this CS shall be ignored.

1 : Intention CS

2 : Format CS

3: Content CS

4: Intended audience CS

5: Origination CS

6: Content alert CS

7: Media type CS

8: Atmosphere CS

9-15: Undefined. Genres with this CS shall be ignored.

Level 1 field: This 8-bit field shall indicate the genre value for the first (i.e. highest) level after the CS (e.g. the 2 of “1.2.3.4”). If this value is not present then this and subsequent genre bytes should not be present.

Level 2 field: This 8-bit field shall indicate the genre value for the second level after the CS (e.g. the 3 of “1.2.3.4”). If this value is not present then this and subsequent genre bytes should not be present.

Level 3 field: This 8-bit field shall indicate the genre value for the third level after the CS (e.g. the 4 of “1.2.3.4”). If this value is not present then this and subsequent genre bytes should not be present.

Genre type: The type attribute of the genre element shall be encoded as an enumerated attribute as described in section 4.6.

4.7.6 dabID

All elements defined as dabIDType are encoded as follows:

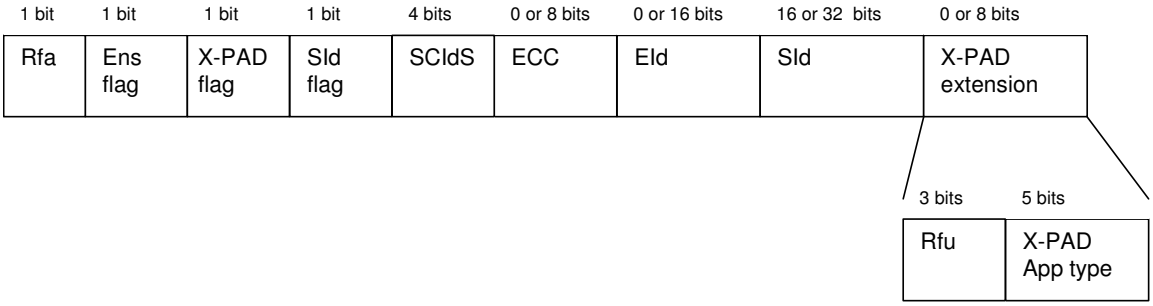


Figure 5 – dabID encoding

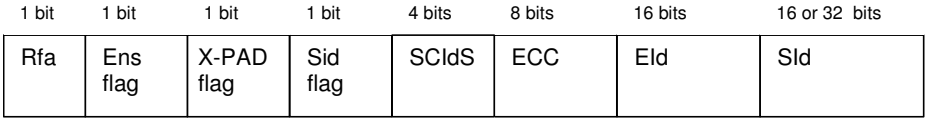


Figure 6 – example dabID encoding (Ens flag ==1 and X-PAD flag == 0)

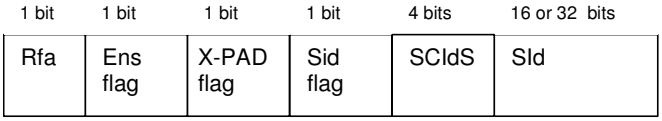


Figure 7 – example dabID encoding (Ens flag ==0 and X-PAD flag == 0)

Rfa: This 1-bit field shall be reserved for future additions. This bit shall be set to zero for the currently specified definition of this field.

NOTE: Receivers shall ignore this bit.

Ens flag: This 1-bit flag shall indicate whether the ECC and EId are contained within the dabID, as follows:

0: ECC and EId are not present. The service that is referenced within the dabID is transmitted on the same ensemble as this EPG service.

1: ECC and EId are present.

X-PAD flag: This 1-bit flag shall indicate whether the addressed component is carried in an X-PAD channel, as follows:

0: Is not carried in an X-PAD channel;X-PAD extension is not present

1: Is carried in an X-PAD channel; X-PAD extension is present

SId flag: This 1-bit flag shall indicate how the SId field is encoded, as follows:

0: SId is encoded as a 16-bit service identifier (i.e.audio service)

1: SId is encoded as a 32-bit service identifier (i.e. data service)

SCIdS: This 4-bit field defines the Service Component Id within the Service (SCIdS).

ECC: This optional 8-bit field defines the Extended Country Code (ECC) of the ensemble on which the service is broadcast and is only present if the Ens flag is set to 1.

EId: This optional 16-bit field defines the Ensemble ID (EId) of the ensemble on which the service is broadcast and is only present if the Ens flag is set to 1.

SId: This 16-bit or 32-bit field (indicated by the SId flag) defines the service identifier.

X-PAD extension: This optional data field is only present if the X-PAD flag is set to 1.

Rfu: This 3-bit field shall be reserved for the future use of the X-PAD App Type (indicated by the XPAD flag). The three bits shall be set to zero for the currently specified definition of this field.

NOTE: Receivers shall check that these bits are zero in order to determine the valid status of the X-PAD App Type field.

X-PAD App Type: This 5-bit field (indicated by the XPAD flag) defines the first X-PAD Application Type (used for X-PAD data applications only).

4.7.7 ensembleID

All elements defined as ensembleIDType are encoded as follows:

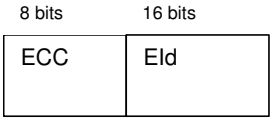


Figure 8 – ensembleID encoding

ECC: This 8-bit field defines the Extended Country Code (ECC) of the ensemble.

EId: This 16-bit field defines the Ensemble ID (EId).

4.7.8 triggerType/Pnum

All elements defined as triggerType are encoded as 4 bytes. See [3] for details on how this is encoded.

4.7.9 URL

All elements defined as urlType are encoded as strings (see section 4.5).

4.7.10 MIME type

All elements defined as mimeType are encoded as strings (see section 4.5).

4.8 Miscellaneous fields

4.8.1 xml:lang

A string shall be used to encode the language code.

4.8.2 index

Used in <memberOf>. Encoded as a 16-bit unsigned integer integer.

4.8.3 version

Used in <programme>, <programmeEvent>, <serviceinformation>, <ensemble>, <service>, <programmeGroups>, <programmeGroup> and <schedule>. Encoded as a 16-bit unsigned integer.

4.8.4 bitrate

Used in <service> and <programme>. Encoded as a 16-bit unsigned integer that should be multiplied by 8 to give the bitrate in kbit/s.

4.8.5 kHz

Used in <frequency>. Encoded as a 24-bit unsigned integer that gives the frequency in kHz.

4.8.6 numOfItems

Used in <programmeGroup>. Encoded as a 16-bit unsigned integer.

4.8.7 width and height

Used in <multimedia>. Each encoded as a 16-bit unsigned integer.

4.9 Token table element

This element is not defined in the XML specification. Frequently recurring strings in the EPG character data ("tokens") can be encoded using a token table. A maximum of 16 tokens are allowed per table. This table defines tags (bytes that can be identified in the character data stream) and their equivalent strings. Whenever a decoder finds a token tag in a character stream it shall replace the tag with its equivalent string. This element can only occur within the two top-level elements (epg and serviceInformation) and, if present, it shall occur before any other elements. This element applies to all character data within the parent top-level element (i.e. epg or serviceInformation) and all children of the parent element. This element shall be encoded as defined in section 4.3, with the following provisos:

element_tag: This shall always be `0x04`.

element_data_byte: These bytes contain a sequence of one or more tokens (see below).

4.9.1 Tokens

Entries in the token table are encoded as a unique tag and its associated string. Token strings shall never include references to other tokens.

Syntax	Size	Type
token() {		
token_tag	8 bits	uimsbf
token_length	8 bits	uimsbf
for (i=0; i<token_ length; i++) {		
token_data_byte	8 bits	uimsbf
}		
}		

Table 7 - Structure of a token

token_ tag: This byte identifies the token. There are 16 possible tag values (these are all non-printing characters);

`0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x06`, `0x07`, `0x08`, `0x0B`, `0x0C`, `0x0E`, `0x0F`, `0x10`, `0x11`, `0x12`, `0x13`

Note that this excludes the values 0x00 (null), 0x09 (tab), 0x0A (linefeed) and 0x0D (carriage return). Each tag shall occur at most once within the token table.

token_length: This field indicates the number of data bytes in the token string. The range of this is `0x00` to `0xFF` (i.e. 0-255).

token_data_byte: The token string.

4.10 Default dabID

This element is not defined in the XML specification. This element can only occur within the top-level element (epg) and, if present, it shall occur after the string token table (if present) and before any other child elements. This element applies to all location elements within the parent top-level element (i.e. epg) and all children of the parent element. If the default dabID element is present, then whenever the default dabID occurs within a location, it does not need to be encoded. Whenever a decoder finds a missing dabID for a location, then it shall use the default dabID. If a dabID is present for a particular location then the decoder shall use this dabID tag, rather than the default dabID value. This element shall be encoded as defined in section 4.3 with the following provisos:

element_tag: This shall always be `0x05`.

element_data_byte: These bytes shall be a dabID as defined in section 4.7.6..

5. Profiling

5.1 Profiles

There are two different profiles for each of the three EPG "data types" (programme information, group information and service information), a **'Basic'** profile and an **'Advanced'** profile.

5.1.1 Basic Profile

The target receivers for the **'Basic'** profile are simple/embedded receivers. These receivers typically have in the order of 25 kBytes or less of available memory for the EPG decoder code and for data storage.

In order to maximize the available broadcast bandwidth and storage capacity a simple binary encoding mechanism is utilised (see section 4). This profile does not permit the use of GZIP(deflate) compression of any of the '**Basic**' profile objects or of the MOT directory of the carousel in which it is broadcast.

The attributes and elements from the XML Specification that can be used in the '**Basic**' profile are restricted. A list of the permitted attributes and elements is included in Annex A (normative): Profiling tables.

5.1.2 Advanced Profile

Any remaining attributes or elements from the EPG XML Specification are broadcast within the **'Advanced'** profile. Again, the data is binary encoded, but this profile also allows for the optional application of GZIP (deflate) compression to the encoded data.

5.2 Fragmentation of the Profile data into objects

The EPG data is fragmented into objects based on the data type and profile. Note that an EPG may describe services that are not transmitted on the same ensemble as the EPG service.

For examples of the fragmentation of the data, please see Annex B (informative): Profiling/fragmenting examples.

Note that the "advanced" data shall be carried as '**Advanced**' profile objects with a link to the '**Basic**' profile information. Note that the '**Basic**' and '**Advanced**' profile data for a specific service shall be contained within a single carousel. An appropriate decoder shall then internally combine the sets of information from the different profiles to present a consistent EPG. Consequently, when providing an '**Advanced**' profile EPG service there shall also be an associated '**Basic**' profile service; decoders wishing to decode an advanced service shall also decode the associated basic service.

For details of the scheme to combine profile data, please see section 5.3.

5.2.1 Service Information (SI)

The '**Basic'** profile service information for all of the services on a single ensemble described by this EPG service shall be contained within one object. If the EPG service contains data for more than one ensemble, then there shall be one '**Basic'** profile service information object per ensemble.

If there is additional service information for any of these services, then they shall be carried within additional objects as '**Advanced'** profile service information objects and can be additionally compressed with GZIP. The grouping of this additional service information into objects is flexible; the advanced data can be grouped in any way. For instance, all the additional service information data for each ensemble can be contained within individual objects, or contained within a single object.

5.2.2 Programme Information (PI)

For programme information in the '**Basic**' profile there shall be one object for a single service for a period of one day. The total number of '**Basic**' profile programme information objects shall therefore be the number of services described, multiplied by the number of days covered by the EPG.

One day is defined as all programmes carried on one or more services that are billed to start at or between 00:00:00 (local time) and 23:59:59 (local time) on a particular date.

All programmes in the '**Basic**' profile PI object shall be sorted chronologically by start time.

NOTE: Where a programme contains multiple time elements, these individual times shall be sorted chronologically. The individual programme shall be sorted within the object based on the start time of its first time element.

The grouping of all other programme information (i.e. the greater detail carried by the '**Advanced**' profile programme information) is flexible; the advanced data can be grouped by ensemble, service or by day. Receivers shall be capable of supporting all methods of grouping the advanced programme information data.

NOTE: Programme Information objects should not be removed from the broadcast channel until the billed stop times for all of the programmes contained within those objects have expired (i.e. the current local time is later than the latest billed stop time contained within the programme information data).

5.2.3 Group Information (GI)

The '**Basic'** profile group information for all of the services on a single ensemble described by this EPG service shall be contained within one object. If the EPG service contains data for more than one ensemble, then there shall be one '**Basic'** profile group information object per ensemble.

If there is additional group information for any of these services, then they shall be carried within additional objects as '**Advanced'** profile group information objects and can be additionally compressed with GZIP. There shall be one '**Advanced'** profile object for all of the services on a single ensemble described by this EPG service.

5.3 Scheme to combine the profile data

The '**Basic**' and '**Advanced**' profile data is derived from a *master* XML file that conforms to the EPG schemas defined in [1]. These derived files are well formed XML documents that are encoded separately using the binary encoding format referred to in section 4.

The format of the basic profile XML shall match the form of the master XML but with only the basic profile elements and attributes included. Each element shall have the same nesting and order as in the master document.

The advanced profile data shall match the form of the original XML, with the following data removed:

- attributes and elements that are included in the basic profile
- text/CDATA that are included in the basic profile
- elements that are empty as a result of these removals

There shall be no duplication of attributes and text across the basic and advanced documents apart from those required by the receiver to merge the two documents, as detailed in Table 8, Table 9 and Table 10.

For an EPG that contains no 'advanced' profile data, the basic profile document will be the same as the master document.

5.4 Attributes required for merging

To enable an advanced receiver to combine the basic and advanced files some attributes and tags shall be duplicated in both the basic and advanced files. The elements/attributes that need to be present in both are:

5.4.1 Service Information

Element	Attribute
serviceInformation	version
serviceInformation.ensemble	Id
serviceInformation.ensemble.service.serviceID	Id

Table 8 – service information merging attributes

5.4.2 Programme Information

Element	Attribute
epg.schedule	version
programme	shortId

Table 9 – schedule information merging attributes

5.4.3 Group Information

Element	Attribute
epg.programmeGroups	version
epg.programmeGroup	shortId

Table 10 – group information merging attributes

A receiver should not attempt to merge data unless the IDs and versions match in the basic and advanced data. If these do not match, then only the basic profile data shall be used.

6. Transportation

6.1 Transport mechanism

MOT in Directory Mode [2] will be employed as the method for transporting the EPG data.

The mapping of the EPG data onto MOT objects is described in section 5.2.

Additionally, the MOT Directory shall not be compressed and the header information within the MOT directory shall be sorted in ascending order of the `ContentName`, signalled by the MOT Directory extension parameter `SortedHeaderInformation` as detailed within the MOT specification [2]**.**

6.2 Maximum Object Size

Each 'basic' profile MOT object shall have a maximum size of 8 Kilobytes (8,192 bytes), and a maximum size for the MOT directory of 8 Kilobytes (8,192 bytes). The size of each of the 'Advanced' profile objects is not restricted.

If the size of the MOT directory object exceeds 8 Kilobytes then individual services shall be moved to an alternative carousel, until the size of the directory object is at or below the maximum size.

The 'basic' and 'advanced' profile data for a specific service shall be contained within a single carousel.

6.3 Maximum Channel Size

For any carousels containing EPG objects, the data rate for MOT transported in either packet mode or PAD is limited to a maximum of 64 kbps. For packet mode transport the overall size of the subchannel including the EPG is limited to a maximum of 128 kbps.

6.4 MOT Parameters

In order to provide useful tracking information for receivers to efficiently download and cache data, MOT parameters will be used.

MOT parameters that are to be applied to individual MOT objects are carried within the MOT header information of each directory entry in the MOT directory. A summary of the MOT parameters for individual objects that apply to the EPG specification are given in Table 11, and are specified in detail by the following subclauses.

The MOT functionality "caching" is optional for both UA provider and receiver [2].

If the MOT parameters `ProfileSubset`, `CAInfo`, `ContentName` and/or `UniqueBodyVersion` are used then these parameters shall be sorted in this order and placed at the beginning of the MOT parameter list of the MOT header**.**

Note that other parameters may be defined within the context of a specific profile definition. Any parameters that are encountered that are not understood by a given receiver profile shall be ignored.

The MOT parameters detailed in Table 11 will be used to identify the content of the individual objects.

Parameter	Parameter Id	Specified in	Mandatory for UA provider	Mandatory for Receiver	Occurrence
ProfileSubset	0x21	MOT	No (but the parameter shall be used for all 'Advanced' profile objects)	Yes	Single
ContentName	0x0C	MOT	Yes	Yes	Single
CompressionType	0x11	MOT	No (but the parameter shall be used for all objects compressed on MOT transport level)	Yes for "advanced" profile receivers ('basic' profile objects shall not be compressed)	Single
CAInfo	0x23	MOT	No (but the parameter shall be used for all objects encrypted on MOT level)	Yes (non CA capable receivers shall discard encrypted objects)	Single
ScopeStart	0x25	EPG	See section 6.4.6	No	Single
ScopeEnd	0x26	EPG	See section 6.4.7	No	Single
ScopeID	0x27	EPG	Yes	No	Single

Table 11 – mot parameters used to identify individual objects

6.4.1 MOT Header Core

The ContentType parameter indicates the main category of the body's content. The ContentSubType parameter indicates the exact type of the body's content depending on the value of the field [2].

The parameters ContentType / ContentSubType identify whether the EPG data is schedule, service or group information. A list of permitted values for ContentType / ContentSubtype for EPG specific data is listed in Table 12. The ContentType of all EPG-specific data is the "application" value (7).

Please therefore note that these ContentSubType values are only unique within the (EPG) application. Other applications could use the same values for other content types.

ContentType/ContentSubType value	Description
7 / 0	Object contains Service Information
7 / 1	Object contains Programme Information
7 / 2	Object contains Group Information

Table 12 – epg content type / subtype values

6.4.2 ProfileSubset

Where a carousel contains objects for more than one profile, additional handling may be applied by the MOT decoder if it knows which profile a given object is used by. The `ProfileSubset` parameter identifies the profile for which the object is relevant. The possible `ProfileSubset` values are those defined for the profile IDs in Table 13. If the `ProfileSubset` parameter is not specified for an object in the carousel, the receiver shall assume that the object is relevant to all profiles supported by the EPG user application that are indicated by the application signalling. The `ProfileSubset` parameter is used as specified in the MOT specification [2].

Note: Basic objects are required by both basic and advanced profile receivers, and so it not necessary to specify the `ProfileSubset` parameter for basic objects.

6.4.3 ContentName

This mandatory MOT parameter uniquely identifies the object within the MOT carousel and within the EPG service.

The `ContentName` parameter is used as specified in the MOT specification [2].

Note that the `ContentName` itself should be kept to a minimum, as it contains no useful information other than a unique identifier.

6.4.4 CompressionType

The `CompressionType` parameter is used as specified in the MOT specification [2].

No objects containing ‘basic’ profile data shall be compressed.

The objects containing ‘advanced’ profile information may include large amounts of raw text. Consequently, GZIP (deflate) compression can subsequently be applied to the binary encoded objects to increase the compression. This data will only be used by the more advanced receivers, which shall have such a capability to de-compress this data. An EPG decoder shall support a maximum window size of 32kB for GZIP.

6.4.5 CAInfo

This parameter is used if conditional access on the MOT level is applied to the MOT data. The `CAInfo` parameter is used as specified in the MOT specification [2].

If this parameter is present a non CA-capable device shall discard this MOT object.

6.4.6 ScopeStart

This parameter is used for Programme Information EPG objects only; it shall not be used for Service Information or Group Information objects. For Programme Information objects, this parameter shall be mandatory and is used to indicate the billed start date and time (in service local time) of the first programme covered by EPG data contained within this Programme Information object. This shall be encoded as defined in section 4.7.1, with the following restriction that only short-form UTC with an optional LTO shall be used. The start time shall be rounded down to the nearest minute.

6.4.7 ScopeEnd

This parameter is used for Programme Information EPG objects only; it shall not be used for Service Information or Group Information objects. For Programme Information objects, this parameter is used to indicate the billed end date and time (in service local time) of the last programme covered by EPG data contained within this Programme Information object. This shall be encoded as defined in section 4.7.1, with the following restriction that only short-form UTC with an optional LTO shall be used. The end time shall be rounded down to the nearest minute.

Note: This parameter shall be mandatory for Programme Information (PI) objects that are **not** continguous, i.e. where there is not another PI object that starts immediately after the end of the current one. In the case where a PI object exists in the carousel with a ScopeStart equal to the ScopeEnd of the current object then the ScopeEnd may be omitted for this current object.

6.4.8 ScopeID

If the object contains *Service Information* or *Group Information*, then this parameter indicates the Ensemble Country Code and the Ensemble ID of the ensemble for which the object contains data (coded as specified in section 4.7.7).

If the object contains *Programme Information*, then this parameter indicates the *dabID* of the service for which the object contains data, as specified in section 4.7.6. If the object contains schedule data for more than one service then this parameter shall consist of a continuous sequence of the relevant *dabIDs*.

This parameter is encoded as a sequence of bytes representing the *dabID*, or sequence of *dabIDs*, as defined in section 4.7.6.

6.5 Transportation of Other Objects

The MOT carousel may also contain other content. These objects shall be transported as detailed within the MOT specification [2] and using the appropriate signaling as detailed within that specification.

These additional objects may only be transported in the same MOT carousel as the basic profile data on the condition that the MOT directory will not exceed the maximum size permitted for the 'basic' profile data (see section 6.2). If the MOT directory exceeds the maximum permitted size then these additional objects shall be transported in an additional carousel.

The EPG-specific parameters (`ScopeStart`, `ScopeEnd` and `ScopeID`) shall not be used for these objects.

7. Signalling

7.1 FIG 0/13 (Application Type) signalling

The use of the EPG application within a DAB data channel shall be indicated by the use of FIG0/13 with a UserApplicationType value of 7.

The data is profiled into '**Basic**' and '**Advanced**' profile EPG objects, as described in section 5.

The user application data field for the EPG user application is a sequence of 1-byte values, each being a `ProfileID`, indicating which profile(s) of the EPG service are carried there. If there is more than one `ProfileID`, then the list shall be sorted in ascending order with the lowest ProfileID first. The list of ProfileIds is closed with a 0x00 at the end. The `ProfileID`s are defined in Table 13. Any remaining values for the `ProfileID` are reserved for future use and shall not be processed by receivers:

Profile ID	Description
0x00	Reserved
0x01	Basic profile
0x02	Advanced profile
0x03 .. 0xFF	Reserved

Table 13 – profile IDs

7.2 EPG service referencing signalling

To enable an EPG decoder to quickly determine whether a particular service component has an associated EPG, and if it does, where that EPG is located, each service component that has an associated EPG, shall reference the data channel that carries its EPG using the appropriate FIG signalling (as defined in ETSI EN 300 401 [3]).

Note that this EPG service data channel could be transmitted on another ensemble.

7.3 FIG0/9 and FIG 0/10 (Reference time) signalling

The provision of a correctly broadcast reference time within FIG 0/10 and local time offset in FIG 0/9 is a mandatory requirement of this User Application. For further details on these parameters, read [3].

7.4 FIG 0/16 (PNUM) signalling

The provision of a correctly signaled Programme Number within FIG 0/16 is required in order to use the programme 'trigger' attribute within the DAB EPG specification. For further details on this parameter, read [3].

8. Annex A (normative): Profiling tables

8.1 Elements & attributes that are transmitted in the 'Basic' profile

The elements and attributes below are those that form the 'Basic' profile.

Key:

- O : Optional
- R : Required
- R1 : Required only if the parent is not empty
- R2 : Required only if the actual value is not the same as the default value
- R3: Not required if the data is transported within an XPAD channel

8.1.1 Service Information

Element	Attribute	Required ?
serviceInformation		R
	system	R2
serviceInformation.ensemble		R
	id	R
serviceInformation.ensemble.shortName		R
	xml:lang	R2
serviceInformation.ensemble.mediumName		R
	xml:lang	R2
serviceInformation.ensemble.frequency		R
	type	R
	kHz	R
serviceInformation.ensemble.mediaDescription		O
serviceInformation.ensemble.mediaDescription.multimedia		O
	type	O
	mimeValue	O
	xml:lang	R2
	url	R1
	width	O
	height	O
serviceInformation.ensemble.service		R
	format	R2
	bitrate	O
serviceInformation.ensemble.service.serviceID.		R
	id	R
	type	R2
serviceInformation.ensemble.service.shortName		R
	xml:lang	R2
serviceInformation.ensemble.service.mediumName		R
	xml:lang	R2
serviceInformation.ensemble.service.mediaDescription		O
serviceInformation.ensemble.service.mediaDescription.multimedia		O
	type	O
	mimeValue	O
	xml:lang	R2
	url	R1
	width	O
	height	O

Table 14 – Basic profile service information

8.1.2 Programme Information

Element	Attribute	Required ?
epg		R
epg.schedule		R
epg.schedule.scope	 startTime stopTime	O R1 R1
epg.schedule.scope.serviceScope	 id	O R1
epg.schedule.programme	 short id recommendation broadcast bitrate	R R R2 R2 R2
epg.schedule.programme.mediumName	 xml:lang	R R2
epg.schedule.programme.longName	 xml:lang	O R2
epg.schedule.programme.location epg.schedule.programme.location.time epg.schedule.programme. location.bearer	 time duration id trigger	R R R R R3 R3 O
epg.schedule.programme.mediaDescription epg.schedule.programme.mediaDescription. ShortDescription	 xml:lang	O O R2
epg.schedule.programme.genre	 href type	O R1 R2
epg.schedule.programme.memberof	 short id index	O R1 O

Table 15 – Basic profile schedule information

8.1.3 Group Information

Element	Attribute	Required ?
Epg		R
epg.programmeGroups		R
epg.programmeGroups.programmeGroup	 short id type numofItems	R R O O
epg.programmeGroups.programmeGroup.mediumName	 xml:lang	R R2
epg.programmeGroups.programmeGroup.longName	 xml:lang	O R2
epg.programmeGroups.programmeGroup.genre	 href type	O R1 R2
epg.programmeGroups.programmeGroup.memberof	 short id index	O R1 O

Table 16 – Basic profile group information

8.2 Elements & attributes that are transmitted in the 'Advanced' profile

The '**Advanced**' profile consists of any valid elements and attributes from the EPG XML specification that are not included within the '**Basic**' profile.

9. Annex B (informative): Profiling/fragmenting examples

9.1 Profiling/fragmenting example 1

Example, using two ensemblees, each with eight services (advanced information grouped by service and by ensemble):

Ensemble A	Ensemble B
Service 1	Service 1
Service 2	Service 2
Service 3	Service 3
Service 4	Service 4
Service 5	Service 5
Service 6	Service 6
Service 7	Service 7
Service 8	Service 8

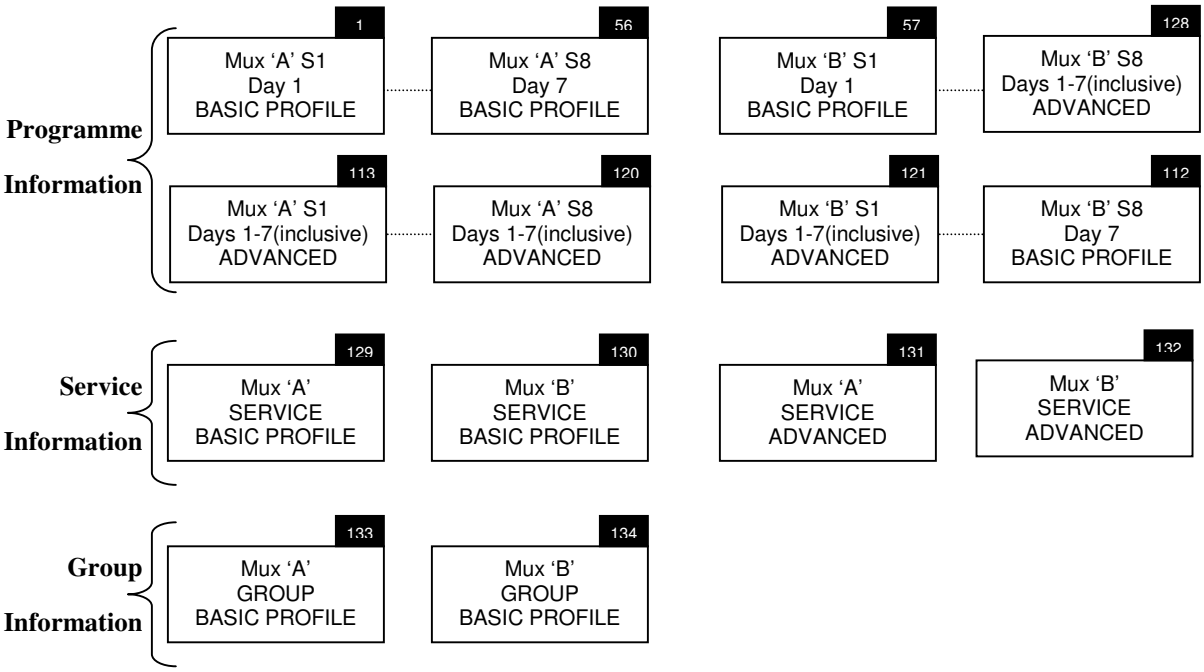
Table 17 – Profiling example 1

Ensemble 'A' broadcasts an EPG service consisting of programme information for ALL services on BOTH ensembles.

Consequently, the number of objects generated is:

- 2x Service Information objects (basic detail).
- 2x Service Information objects (advanced detail)
- 2x Group Information objects (basic detail)
- 56x Programme Information objects for all services on mux A (basic detail)
- 8x Programme Information objects for all services on mux A (advanced detail)
- 56x Programme Information objects for all services on mux B (basic detail)
- 8x Programme Information objects for all services on mux B (advanced detail)

A total of 134 objects.


Programme
Information
1
Mux 'A' S1
Day 1
BASIC PROFILE
56
Mux 'A' S8
Day 7
BASIC PROFILE
57
Mux 'B' S1
Day 1
BASIC PROFILE
128
Mux 'B' S8
Days 1-7(inclusive)
ADVANCED
113
Mux 'A' S1
Days 1-7(inclusive)
ADVANCED
120
Mux 'A' S8
Days 1-7(inclusive)
ADVANCED
121
Mux 'B' S1
Days 1-7(inclusive)
ADVANCED
112
Mux 'B' S8
Day 7
BASIC PROFILE
Service
Information
129
Mux 'A'
SERVICE
BASIC PROFILE
130
Mux 'B'
SERVICE
BASIC PROFILE
131
Mux 'A'
SERVICE
ADVANCED
132
Mux 'B'
SERVICE
ADVANCED
Group
Information
133
Mux 'A'
GROUP
BASIC PROFILE
134
Mux 'B'
GROUP
BASIC PROFILE

9.2 Profiling/fragmenting example 2

Example, using a single ensemble, consisting of multiple EPG channels (advanced information grouped by service):

Service Provider	Service	EPG Channel
Service Provider A	Service 1	EPG 1
Service Provider A	Service 2	EPG 1
Service Provider A	Service 3	EPG 1
Service Provider A	Service 4	EPG 1
Service Provider B	Service 5	EPG 2
Service Provider B	Service 6	EPG 2
Service Provider C	Service 7	EPG 3
Service Provider D	Service 8	EPG 4
Service Provider E	Service 9	EPG 5

Table 18 - – Profiling example 2

Five service providers, A-E, have services on the same ensemble. They have a different number of services each; one has 4 services, one has 2 services and three have only a single service.

Service providers broadcast one EPG for all their services in packet mode. So there are five EPG services, one including four services, one including two services and three including one programme service. EPGs for multiple programme services are transmitted in packet mode, EPGs for single programme services are transmitted in PAD.

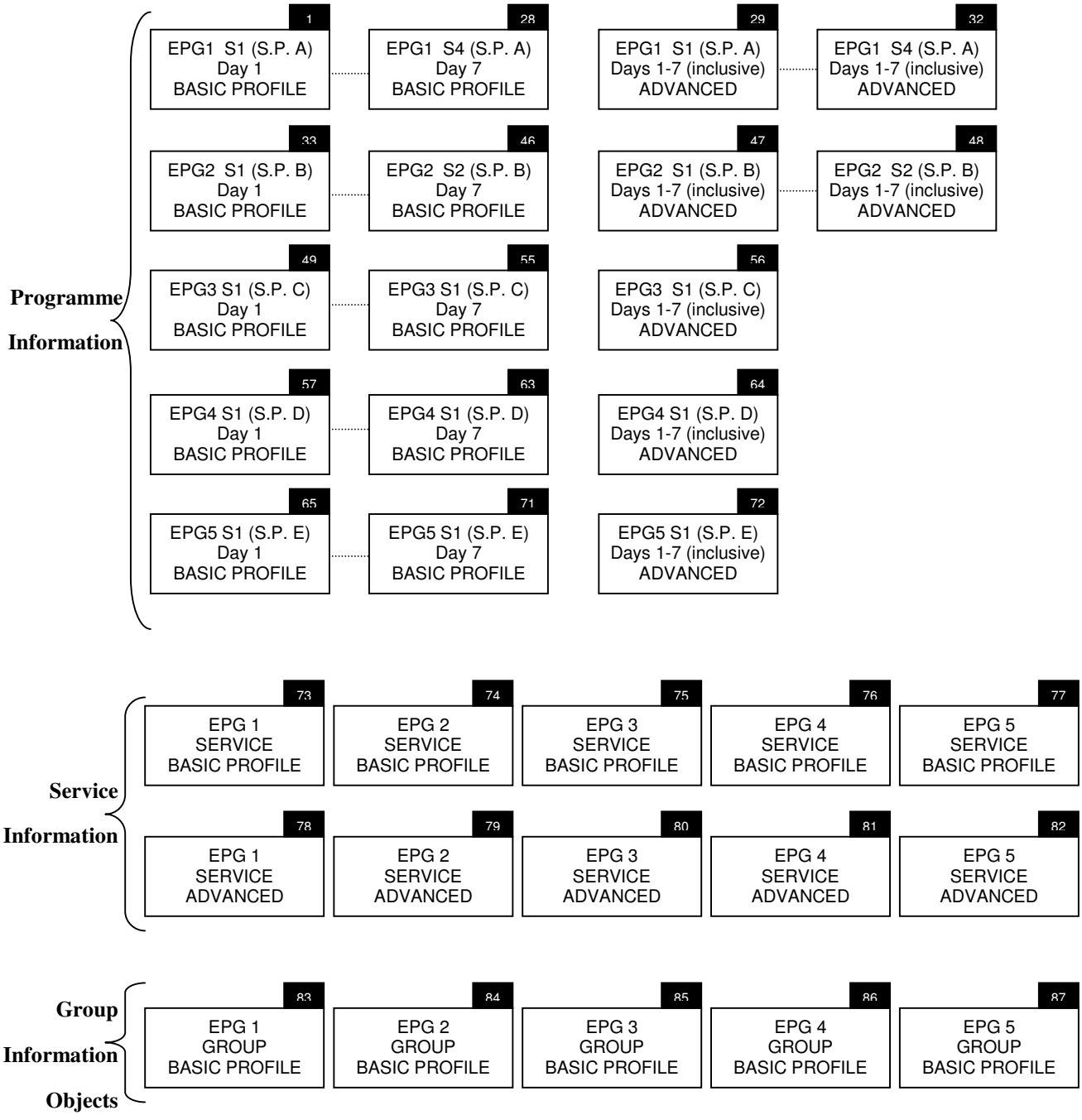
No EPG data from other ensembles is incorporated.

Consequently, the number of objects generated is:

- 5x Service Information objects (basic detail).
- 5x Service Information objects (advanced detail)
- 5x Group Information objects (basic detail)
- 28x Programme Information objects for all services on EPG 1 (basic detail)
- 4x Programme Information objects for all services on EPG 1 (advanced detail)
- 14x Programme Information objects for all services on EPG 2 (basic detail)
- 2x Programme Information objects for all services on EPG 2 (advanced detail)
- 7x Programme Information objects for all services on EPG 3 (basic detail)
- 1x Programme Information object for all services on EPG 3 (advanced detail)
- 7x Programme Information objects for all services on EPG 4 (basic detail)

- 1x Programme Information object for all services on EPG 4 (advanced detail)
- 7x Programme Information objects for all services on EPG 5 (basic detail)
- 1x Programme Information object for all services on EPG 5 (advanced detail)

A total of 87 objects.



10. Annex C (informative): Binary encoding example

```
<epg xmlns:epg="http://www.worlddab.org/schemas/epg" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.worlddab.org/schemas/epg epgSchedule_11.xsd" system="DAB">
   <schedule version="1" creationTime="2001-02-28T00:00:00" originator="BBC">
      <scope startTime="2003-12-18T17:00:00" stopTime="2003-12-18T18:00:00">
         <serviceScope id="e1.ce15.e1c224.0"/>
      </scope>
      <programme shortId="16442449">
         <epg:mediumName>PM</epg:mediumName>
         <epg:location>
            <epg:time time="2003-12-18T17:00:00" duration="PT1H0M0S"/>
            <epg:bearer id="e1.ce15.e1c224.0"/>
         </epg:location>
      </programme>
    </schedule>
</epg>
```

These are the bytes for the encoded binary object:

Bytes	Description
02	**<epg>**
3F	Length = 63 bytes
21	**<schedule>**
3D	Length = 61 bytes
24	**<scope>**
17	Length = 23
80	**<startTime>**
04	Length = 4
19 DF E8 80	2003-12-18T17:00:00 *(short form local time)*
81	**<stopTime>**
04	Length = 4
19 DF E9 00	2003-12-18T18:00:00
25	**<serviceScope>**
09	Length = 9
80	ID attribute
07	Length = 7
40 E1 CE 15 E1 C2 24	e1.ce15.e1c224.0
1C	**<programme>**
22	Length = 34
81	Shortid attribute
03	Length = 3
FA E4 51	16442449
11	**<mediumName>**
02	Length = 2
50 4D	PM
19	**<location>**
17	Length = 23
2C	<time>
0A	Length = 10
80	Time attribute
04	Length = 4
19 DF E8 80	2003-12-18T17:00:00 *(short form local time)*
81	Duration attribute
02	Length = 2
0E 10	3600
2D	**<bearer>**
09	Length = 9
80	ID attribute
07	Length = 7
40 E1 CE 15 E1 C2 24	e1.ce15.e1c224.0

Table 19 – Binary encoded example

11. Annex D (normative): Element tags

Note: All element tags, apart from the top-level and other special elements, are within the range 0x10 to 0x7E. All attribute tags are within the range 0x80 to 0xFF.

The tag 0x7F is reserved and will never have a value defined here. It may therefore be used in receivers for internal use (e.g. to signify invalid elements).

Child element	Possible parent elements	Tag	Defined in section
epg	Top-level	0x02	4.3.1
serviceInformation	Top-level	0x03	4.3.1
tokenTableElement	epg, serviceInformation	0x04	4.9
defaultdabIDElement	epg	0x05	4.10
shortName	programmeGroup, ensemble, service, programme, programmeEvent	0x10	4.3
mediumName	programmeGroup, ensemble, service, programme, programmeEvent	0x11	4.3
longName	programmeGroup, ensemble, service, programme, programmeEvent	0x12	4.3
mediaDescription	programmeGroup, ensemble, service, programme, programmeEvent	0x13	4.3
genre	programmeGroup, service, programme, programmeEvent	0x14	4.3
CA	ensemble, service, programme, programmeEvent	0x15	4.3
keywords	programmeGroup, ensemble, service, programme, programmeEvent	0x16	4.3
memberOf	programmeGroup, programme, programmeEvent	0x17	4.3
link	programmeGroup, ensemble, service, programme, programmeEvent	0x18	4.3
location	programme, programmeEvent	0x19	4.3
shortDescription	CA, mediaDescription	0x1A	4.3
longDescription	CA, mediaDescription	0x1B	4.3
programme	epg, schedule	0x1C	4.3
programmeGroups	epg	0x20	4.3
schedule	epg	0x21	4.3
alternateSource	epg	0x22	4.3
programmeGroup	programmeGroups	0x23	4.3
scope	schedule	0x24	4.3
serviceScope	scope	0x25	4.3
ensemble	serviceInformation	0x26	4.3
frequency	ensemble	0x27	4.3
service	ensemble	0x28	4.3
serviceID	service	0x29	4.3
epgLanguage	service	0x2A	4.3
multimedia	mediaDescription	0x2B	4.3
time	location	0x2C	4.3
bearer	location	0x2D	4.3
programmeEvent	programme	0x2E	4.3

Table 20 - Element tags

12. Annex E (normative): Attribute tags

The encoding of attributes is defined in section 4.4. Additional encoding rules sections are referenced below.

Element	Attribute	Tag	Defined in section
epg[1]	system	0x80	4.6
programmeGroups	version	0x80	4.8.3
	creationTime	0x81	4.7.1
	originator	0x82	4.5
programmeGroup	id	0x80	4.7.3
	shortId	0x81	4.7.4
	version	0x82	4.8.3
	type	0x83	4.6
	numOfItems	0x84	4.8.6
schedule	version	0x80	4.8.3
	creationTime	0x81	4.7.1
	originator	0x82	4.5
scope	startTime	0x80	4.7.1
	stopTime	0x81	4.7.1
serviceScope	id	0x80	4.7.6
alternateSource	protocol	0x80	4.6
	type	0x81	4.6
	url	0x82	4.7.9

Table 21 - epgSchedule attribute tags

Element	Attribute	Tag	Defined in section
serviceInformation	version	0x80	4.8.3
	creationTime	0x81	4.7.1
	originator	0x82	4.5
	serviceProvider	0x83	4.5
	system	0x84	4.6
ensemble	id	0x80	4.7.6
	version	0x81	4.8.3
frequency	type	0x80	4.6
	kHz	0x81	4.5
service	version	0x80	4.8.3
	format	0x81	4.6
	bitrate	0x82	4.8.4
serviceID	id	0x80	4.7.6
	type	0x81	4.6

Table 22 - epgSI attribute tags

[1] Note that this scheme does not encode any of the schema information (e.g. the “xmlns” attribute).

Element	Attribute	Tag	Defined in section
CA	type	0x80	4.6
keywords	xml:lang	0x80	4.8.1
multimedia	mimeValue	0x80	4.7.10
	xml:lang	0x81	4.8.1
	url	0x82	4.7.9
	type	0x83	4.6
	width	0x84	4.8.7
	height	0x85	4.8.7
time	time	0x80	4.7.1
	duration	0x81	4.7.2
	actualTime	0x82	4.7.1
	actualDuration	0x83	4.7.2
bearer	id	0x80	4.7.6
	trigger	0x81	4.7.8
memberOf	id	0x80	4.7.3
	shortId	0x81	4.7.4
	index	0x82	4.8.2
dabLanguage	xml:lang	0x80	4.8.1
link	url	0x80	4.7.9
	mimeValue	0x81	4.7.10
	xml:lang	0x82	4.8.1
	description	0x83	4.5
	expiryTime	0x84	4.7.1
programme	id	0x80	4.7.3
	shortId	0x81	4.7.4
	version	0x82	4.8.3
	recommendation	0x83	4.6
	broadcast	0x84	4.6
	bitrate	0x85	4.8.4
	xml:lang	0x86	4.8.1
programmeEvent	id	0x80	4.7.3
	shortId	0x81	4.7.4
	version	0x82	4.8.3
	recommendation	0x83	4.6
	broadcast	0x84	4.6
shortName	xml:lang	0x80	4.8.1
mediumName	xml:lang	0x80	4.8.1
longName	xml:lang	0x80	4.8.1
shortDescription	xml:lang	0x80	4.8.1
longDescription	xml:lang	0x80	4.8.1
genre	href	0x80	4.7.5
	type	0x81	4.6

Table 23 - epgDataTypes attribute tags

13. Annex F (normative): Enumerated types

The default attribute, if present, is shown in italics and always has the value `0x01`.

Note: If an attribute to be encoded has the default value then there is no need for it to be encoded.

Attribute	Value	Tag
epg system	*DAB*	*0x01*

Table 24 – epg system

Attribute	Value	Tag
programmeGroup type	series	0x02
	show	0x03
	programConcept	0x04
	magazine	0x05
	programCompilation	0x06
	otherCollection	0x07
	otherChoice	0x08
	topic	0x09

Table 25 – programmeGroup type

Attribute	Value	Tag
alternateSource protocol	*URL*	*0x01*
	DAB	0x02

Table 26 – alternateSource protocol

Attribute	Value	Tag
alternateSource type	*identical*	*0x01*
	more	0x02
	less	0x03
	similar	0x04

Table 27 - alternateSource type

Attribute	Value	Tag
frequency type	*primary*	*0x01*
	alternative	0x02

Table 28 - frequency type

Attribute	Value	Tag
service format	*Audio*	*0x01*
	DLS	0x02
	MOTSlideshow	0x03
	MOTBWS	0x04
	TPEG	0x05
	DGPS	0x06

	proprietary	0x07

Table 29 – service format

Attribute	Value	Tag
serviceID type	*primary*	*0x01*
	secondary	0x02

Table 30 - serviceID type

Attribute	Value	Tag
CA type	*none*	*0x01*
	unspecified	0x02

Table 31 - CA type

Attribute	Value	Tag
broadcast type	*on-air*	*0x01*
	off-air	0x02

Table 32 - broadcast type

Attribute	Value	Tag
recommendation type	*no*	*0x01*
	yes	0x02

Table 33 - recommendation type

Attribute	Value	Tag
system type	*DAB*	*0x01*

Table 34 - system type

Attribute	Value	Tag
multimedia type	logo_unrestricted	0x02
	logo_mono_square	0x03
	logo_colour_square	0x04
	logo_mono_rectangle	0x05
	logo_colour_rectangle	0x06

Table 35 - multimedia type

Attribute	Value	Tag
genre type	*main*	*0x01*
	secondary	0x02
	other	0x03

Table 36 - genre type

14. History

Document history		
V1.0.3	September 2004	Publication