# ETSI TS 102 371 V3.2.1 (2016-05)


# Digital Audio Broadcasting (DAB); Digital Radio Mondiale (DRM); Transportation and Binary Encoding Specification for Service and Programme Information (SPI)

Reference
RTS/JTC-DAB-77

Keywords
audio, broadcasting, DAB, digital, DRM, 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

# 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 (https://ipr.etsi.org/).

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 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

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, ETSI EN 300 401 [3], for DAB (see note 2) which now has worldwide acceptance.

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

The DAB family of standards is supported by World DAB, an organization with members drawn from broadcasting organizations and telecommunication providers together with companies from the professional and consumer electronics industry.

# Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

# Introduction

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

- Navigation and selection of services and programmes.
- 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.
- Searching through current, future and past programme listings, including on-demand content (Filecast).
- Timed recording of individual programmes, or of groups of programmes and themed or similar programming.

The present document is compatible with the hybrid digital radio SPI, ETSI TS 102 818 (V3.1.1 or later) [1].

# 1 Scope

The present document defines how the XML schema data model for Service and Programme Information (SPI) (ETSI TS 102 818 [1]) should be compressed, profiled and broadcast. Within the present document the term "DAB" is used to refer to the Digital Audio Broadcasting standard (ETSI EN 300 401 [3]) and "DRM" is used to refer to the Digital Radio Mondiale standard (ETSI ES 201 980 [6]).

In respect to previous versions of the present document, hybrid radio provisions have been added to allow a seamless experience for users when consuming radio services delivered by digital radio broadcasting systems (DAB, DRM) or IP or a combination of both. The use of the present document allows content to be created once by the service provider for delivery by both mechanisms and allows manufacturers to implement devices with many common elements.

# 2 References

## 2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are necessary for the application of the present document.

[1] ETSI TS 102 818 (V3.1.1 or later): "Hybrid Digital Radio (DAB, DRM, RadioDNS); XML Specification for Service and Programme Information (SPI)".

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

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

[4] ISO/IEC 10646: "Information technology - Universal Coded Character Set (UCS)".

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

[6] ETSI ES 201 980: "Digital Radio Mondiale (DRM); System specification".

[7] ETSI TS 101 968: "Digital Radio Mondiale (DRM); Data applications directory".

## 2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 102 371 (V1.3.1): "Digital Audio Broadcasting (DAB); Digital Radio Mondiale (DRM); Transportation and Binary Encoding Specification for Electronic Programme Guide (EPG)".

[i.2] ETSI TS 102 818 (V1.5.1): "Digital Audio Broadcasting (DAB); Digital Radio Mondiale (DRM); XML Specification for Electronic Programme Guide (EPG)".

[i.3] ETSI TS 103 177: "Digital Audio Broadcasting (DAB); Filecasting; User application specification".

# 3 Definitions and abbreviations

## 3.1 Definitions

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

**delivery system:** broadcast system, either DAB or DRM, used to deliver the SPI service

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

**entity reference:** group of characters used in text strings as a substitute for a single specific character, e.g. &amp

**filecast channel:** data service component containing media files according to ETSI TS 103 177 [i.3]

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

**service:** "radio station" such as BBC Radio 4 or Heart

**Service Identifier (SId):** 16-bit, 24-bit 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 |
| CDATA | Character DATA |
| CRID | Content Reference ID |
| CS | Classification Scheme |
| DAB | Digital Audio Broadcasting |
| DRM | Digital Radio Mondiale |
| ECC | Extended Country Code |
| EId | Ensemble Identifier |
| EPG | Electronic Programme Guide |
| FAC | Fast Access Channel |
| FIG | Fast Information Group |
| GCC | Global Country Code |
| GI | Group Information |
| GZIP | GnuZIP |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| LTO | Local Time Offset |
| MIME | Multipurpose Internet Mail Extensions |
| MJD | Modified Julian Date |
| MOT | Multimedia Object Transfer |
| PAD | Programme Associated Data |
| PI | Programme Information |
| Rfa | Reserved for future addition |
| Rfu | Reserved for future use |
| SCIdS | Service Component Identifier within the Service |
| SDC | Service Description Channel |
| SI | Service Information |
| SId | Service Identifier |
| SPI | Service and Programme Information |
| UA | User Application |
| UTC | Co-ordinated Universal Time |
| UTF | Unicode Transformation Format |
| XML | eXtensible Markup Language |

X-PAD eXtended - Programme Associated Data

# 4 Encoding

## 4.0 Introduction

The present document provides a method to encode raw SPI XML data, generated as per the hybrid radio SPI XML specification ETSI TS 102 818 [1], into a compact binary format to be broadcast using the MOT protocol [2] using the DAB [3] or DRM [4] systems.

The hybrid SPI XML contains elements and attributes that are not relevant for broadcast use and consequently these are not binary encoded. Similarly, it also represents the data in a slightly modified form to the former broadcast only version ETSI TS 102 818 [i.2], and in order to generate backwards compatible binary that permits devices equipped with an EPG decoder based on the former broadcast only version of the present document ETSI TS 102 371 [i.1], to continue to decode correctly, some additional steps are required to generate the binary.

In the present document the term "delivery system" is used to indicate whether the SPI service is delivered over DAB or DRM. This parameter is used in the encoding of several elements and attributes, but is not itself encoded. Therefore, decoders need to know which system delivered the binary data in order to correctly decode it.

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 devices 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 in clause 4.2. Attributes are coded in a similar way (see clause 4.4). The hierarchical nature of the SPI 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 clause 4.7. For an example of a binary encoded XML object, see annex C.

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 clause 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.

**Table 1: Data type mnemonics for syntax specification**

| Mnemonic | Description |
|---|---|
| Uimsbf | Unsigned integer, most significant bit first |

## 4.2 Binary objects

The basic binary objects defined by the present document are defined in table 2. Each binary object carries a single top level element and shall be carried within a single MOT object.

**Table 2: Structure of a binary object**

| Syntax | Size | Type |
|---|---|---|
| `binary_object() {`<br>`  top_level_element()`<br>`}` | | |

**top_level_element:** A top level element as defined in clause 4.3.1.

# 4.3 Elements

## 4.3.0 General

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

**Table 3: Structure of an element**

| 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 |
| `    }` | | |
| `}` | | |

**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. Elements with tags that are not defined here are reserved for future use; the tags and their associated content shall not be processed by devices.

**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 (including special codings, see clause 4.5).

## 4.3.1 Top-level elements

There are two top-level elements defined in the present document; *epg* and *serviceInformation*. A top-level element shall be carried within a binary object (see clause 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 devices.

**Table 4: Top-level element tags**

| Element | Tag |
|---|---|
| epg | 0x02 |
| serviceInformation | 0x03 |

As well as the appropriate elements defined by the SPI XML specification the top-level elements may also, optionally, contain a string token table (see clause 4.9) and a default language (see clause 4.11). 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 clause 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 language (if present).
4) Child elements.
5) CDATA content.

# 4.4 Attributes

## 4.4.0 General

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

**Table 5: Structure of an attribute**

| 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 |
| `    }` | | |
| `}` | | |

**attribute_tag:** This byte uniquely identifies the attribute **within the parent element**. The possible values are defined in annex E.

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

**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 (also see clause 4.5.1), an enumerated data value (see clause 4.6), a common data type (see clause 4.7) or a specific encoding (see clause 4.8).

NOTE: Any entity references should be expanded.

## 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 device shall always automatically use the default value.

# 4.5 CDATA and strings

## 4.5.0 General

All CDATA or text strings, except textual attributes (see clause 4.4.0) and elements of the georss:doubleList type (see clause 4.7.7) shall be encoded as detailed in table 6.

**Table 6: Structure of a CDATA element**

| 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 |
| } | | |
| } | | |

**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 1: Attributes with text values are encoded as an attribute (see clause 4.4.0) with the *attribute_data_bytes* being the character data bytes only.

NOTE 2: Any entity references should be expanded.

## 4.5.1 Character encoding

All CDATA and other text strings shall use ISO/IEC 10646 [4] with UTF-8 encoding. The ISO/IEC 10646 [4] characters `0xE000` to `0xF8FF` shall not be included within any binary encoded strings.

# 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 table F.1.

# 4.7 Common data types

## 4.7.0 Introduction

Various common data types within the SPI specification (ETSI TS 102 818 [1]) have been identified and the specific encodings to be used are defined in clause 4.7.

## 4.7.1 CRID type

All attributes defined as CRID type are encoded as a string attribute (see clause 4.4.0).

### 4.7.2 Short CRID type

All attributes defined as shortCRID type are encoded as a 24-bit unsigned integer.

### 4.7.3 MIME type

All attributes defined as mimeType are encoded as a string attribute (see clause 4.4.0).

### 4.7.4 timePoint type

All elements defined as timePoint type are encoded as follows:

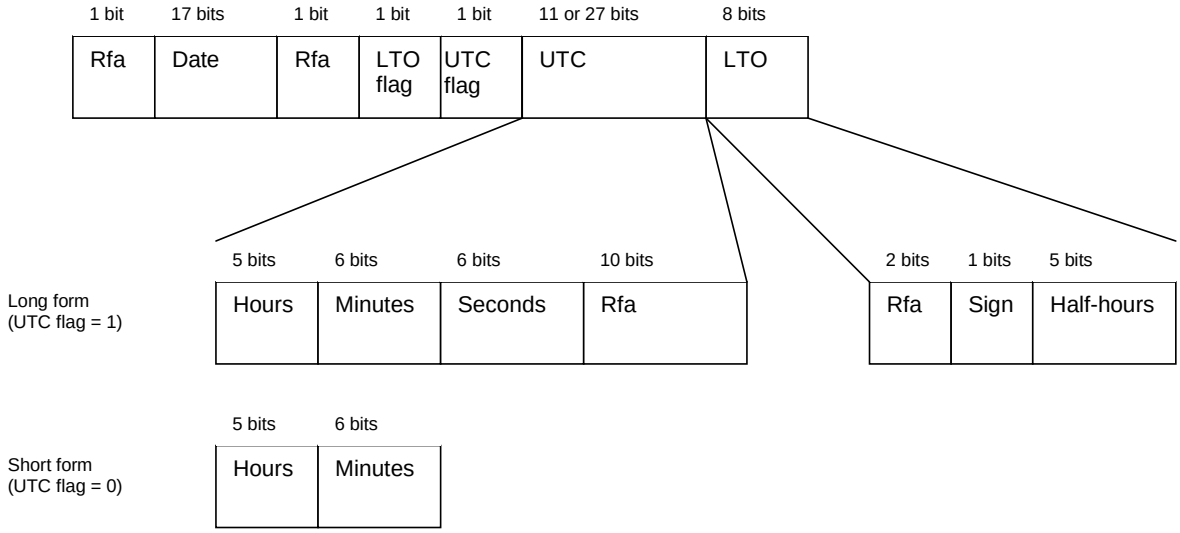



**Figure 2: Date and time encoding (LTO flag == 1)**

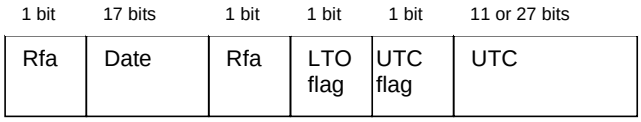



**Figure 3: Date and time encoding (LTO flag == 0)**

**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. Devices shall ignore this bit.

**Date:** This 17-bit unsigned binary number shall define the current date according to the Modified Julian Date (MJD) coding strategy (ETSI EN 300 401 [3]). This number increments daily at 00:00 UTC and extends over the range 0 to 99 999. As an example MJD 50 000 corresponds to October 10$^{th}$, 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. Devices 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, local time is UTC (LTO = 0).
- 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 one further sub-field which shall be encoded as an unsigned binary number. This is a 6-bit field which shall define the seconds. The following 10-bits shall be reserved for future additions. These bits shall be set to zero for the currently specified definition of this field.

**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 devices. 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 0 to 14 hours.

It is important to remember that times specified in XML documents give the local time and the time offset to UTC, whereas the binary encoding specifies the UTC time and the time offset to local time.

EXAMPLE: A programme broadcast at 05:00 in the UK during Daylight Savings time would be specified in the XML document as T05:00:00+01 (i.e. local time and offset to UTC) but when binary encoded would be specified as UTC of 04:00 and an LTO of +1 hour.

## 4.7.5 Duration type

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

## 4.7.6 bearerURI type

### 4.7.6.0 General

The uri domain is used to determine how elements with attributes of bearerURI type shall be encoded.

### 4.7.6.1 dab: domain

Attributes defined as bearerURI type in the dab: domain are encoded using the "id" tag as follows:

| 1 bit | 1 bit | 1 bit | 1 bit | 4 bits | 8 bits | 16 bits | 16 or 32 bits |
|---|---|---|---|---|---|---|---|
| Rfa | Ens flag | X-PAD flag | SId flag | SCIdS | ECC | EId | SId |

**Figure 4: bearerURI encoding for DAB**

**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. Devices shall ignore this bit.

**Ens flag:** This 1-bit flag shall be set to 1 for reasons of backwards compatibility.

**X-PAD flag:** This 1-bit flag shall be set to 0 for reasons of backwards compatibility.

**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 8-bit field carries the Extended Country Code (ECC) . It is the least significant 8-bits of the GCC value.

**EId:** This 16-bit field carries the Ensemble Id (EId).

**SId:** This 16-bit or 32-bit field (indicated by the SId flag) carries the Service Identifier (SId).

#### 4.7.6.2 drm: domain

Attributes defined as bearerURI type in the drm: domain are encoded using the "id" tag as follows:

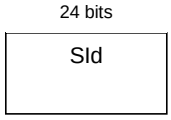



**Figure 5: bearerURI encoding for DRM**

**SId:** This 24-bit field defines the service identifier.

#### 4.7.6.3 http: domain

Attributes defined as bearerURI type in the http: domain are encoded using the "url" tag as a string attribute (see clause 4.4.0).

### 4.7.7 georss:doubleList type

All elements defined as georss:doubleList type are encoded as follows:

Each coordinate pair is coded as two 24-bit signed integers: the first coordinate of the pair is the latitude value which is coded as a 24-bit signed integer, representing the value of latitude multiplied by 92 000 (i.e. in 1/92 000 of a degree, ca. 2,4 m); the second coordinate of the pair is the longitude value which is coded as a 24-bit signed integer, representing the value of longitude multiplied by 46 000 (i.e. in 1/46 000 of a degree, ca. 2,4 m).

## 4.8 Miscellaneous fields

### 4.8.1 xml:lang

All attributes defined as xml:lang are encoded as a string attribute (see clause 4.4.0).

### 4.8.2 index

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

### 4.8.3 version

Used in **programme**, **programmeEvent**, **serviceinformation**, **service**, **programmeGroups**, **programmeGroup** and **schedule**. Encoded as a 16-bit unsigned integer. This value shall be incremented by one, every time a change is made.

### 4.8.4 numOfItems

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

### 4.8.5 width and height

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

# 4.9 Token table element

## 4.9.0 General

This element is not defined in the XML specification. Frequently recurring strings in the SPI 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 clause 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).

The use of the token table can provide good gains in transmission efficiency, provided the encoding of the token table is done well. Decoding advantages are achieved when the token table is ordered such that the string length of tokens increases token by token (i.e. the shortest string is assigned the lowest tag value).

## 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. Every token shall represent a unique string. All defined tokens shall be referenced within the encoded data.

**Table 7: Structure of a token**

| 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 |
| `    }` | | |
| `}` | | |

**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: The values 0x00 (null), 0x09 (tab), 0x0A (linefeed) and 0x0D (carriage return) are excluded.

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 to 255).

**token_data_byte:** The token string.

# 4.10 Void

## 4.11 Default language

This element is not defined in the XML specification. This element can only occur within the top-level elements **serviceInformation** and **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 elements within the parent top-level element and all children of the parent element. If the **default language** element is present, then whenever an **xml:lang** attribute with the same value as the **default language** occurs within an element, it does not need to be encoded. Whenever a decoder finds a missing **xml:lang** attribute for an element, then it shall use the **default language** value. If an **xml:lang** is present for a particular element then the decoder shall use that **xml:lang** value rather than the **default language** value. This element shall be encoded as defined in clause 4.3 with the following provisos:

**element_tag:** This shall always be `0x06`.

**element_data_byte:** These bytes shall be an xml:lang as defined in clause 4.8.1.

## 4.12 Genre element

The **genre** element shall be encoded as follows.

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

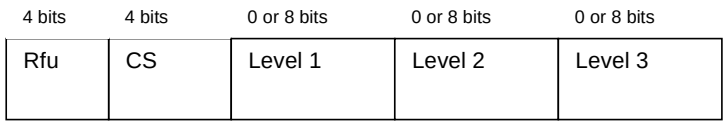



**Figure 6: Genre href encoding**

**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 to 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.

The **type** attribute of the **genre** element shall be encoded as an enumerated attribute as described in clause 4.6.

## 4.13 location element

The **location** element shall be encoded as follows:

- **location** elements containing only **time** or **relativeTime** elements shall always be encoded;
- **location** elements containing **bearer** elements shall only be encoded if at least one **bearer** element has an **id** attribute with the same domain as the delivery system.

## 4.14 onDemand element

The **onDemand** element shall be encoded as follows:

- **onDemand** elements shall only be encoded if at least one **bearer** element has an **id** attribute with the same domain as the delivery system, or with an http: domain.

## 4.15 bearer element

Of the attributes of the bearer element, only the **id** attribute shall be encoded.

When the domain of the **id** attribute matches the delivery system, it shall be encoded according to clause 4.7.6.

As a child element of **onDemand**, when the domain of the **id** attribute is http:, it shall be encoded according to clause 4.7.6, using the **url** attribute tag in place of the **id** attribute tag.

NOTE: **bearer** elements that do not meet the criteria above are not encoded.

## 4.16 serviceScope element

When the domain of the **id** attribute matches the delivery system, it shall be encoded according to clause 4.7.6.

NOTE: **serviceScope** elements that do not meet the criteria above are not encoded.

## 4.17 serviceInformation element

### 4.17.0 Introduction

The structure of the XML now places two container elements **services** and **serviceGroups** as child elements of **serviceInformation**. For a broadcast compliant XML file, the **serviceGroups** element may contain a **serviceGroup** element that represents the former **ensemble** element and the **services** element contains one or more **service** elements. In order to provide backwards compatibility to devices that decode the broadcast only version, the **serviceInformation** element shall be encoded as follows:

- The **serviceInformation** element shall be encoded.
- If the delivery system is DAB, the **ensemble** element shall be encoded according to clause 4.17.1.
- Each **service** element contained in the **services** element shall be encoded; only the **bearer** elements with a domain that matches the delivery system shall be encoded.

NOTE 1: If the delivery system is DAB, the **ensemble** element is required and the **service** element(s) are encoded as children of **ensemble**.

NOTE 2: If the delivery system is DRM, the **ensemble** element is not used and the **service** element(s) are encoded as children of **serviceInformation**.

### 4.17.1 DAB ensemble element encoding

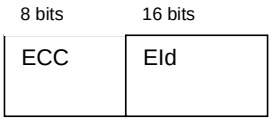
The **ensemble** element is no longer defined in the XML specification but for compatibility reasons, it is still required in the binary encoding when the delivery system is DAB. Two methods of providing the minimum necessary information are possible:

- the encoder configuration information holds the ECC and EId of the ensemble and an **id** attribute to identify a **serviceGroup** element in the SI file that carries the other ensemble information;
- the encoder configuration information holds the ECC and EId of the ensemble and, at a minimum, the **shortName** and **mediumName** elements for the ensemble.

The **ensemble** element shall be encoded as follows:

- The **ensemble** element tag shall be encoded.

The **id** attribute shall encoded as follows:




NOTE: **ECC:** This 8-bit field defines the Extended Country Code (ECC) of the ensemble.
**EId:** This 16-bit field defines the Ensemble Id (EId).

**Figure 7: id attribute encoding**

- For the first method, all child elements of the **serviceGroup** element identified by the **id** attribute in the encoder configuration shall be encoded except the **genre** and **geolocation** elements (if present).
- For the second method, the **shortName**, **mediumName** and any additional child elements present in the encoder configuration shall be encoded.

## 4.18 Unencoded elements

The following hybrid XML elements are not encoded for broadcast delivery: **services**, **serviceProvider**, **serviceGroups**, **serviceGroup** (except as the "ensemble" element, see clause 4.17.1), **serviceGroupMember**.

# 5 Profiling

## 5.1 Profiles

### 5.1.0 Introduction

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

### 5.1.1 Basic profile

The target devices for the "**Basic**" profile are simple/embedded devices. These devices typically have in the order of 25 kbytes or less of available memory for the SPI decoder code and for data storage.

In order to maximize the available broadcast bandwidth and storage capacity a simple binary encoding mechanism is utilized (see clause 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.

### 5.1.2 Advanced profile

Any remaining attributes or elements from the SPI 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

### 5.2.0 Introduction

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

For examples of the fragmentation of the data, please see annex B.

Note that the **"Advanced"** data shall be carried as **"Advanced"** profile objects with a link to the **"Basic"** 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 SPI. Consequently, when providing an **"Advanced"** profile SPI service there shall also be an associated **"Basic"** profile service; decoders wishing to decode an **"Advanced"** profile service shall also decode the associated **"Basic"** profile service.

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

### 5.2.1 Service Information (SI)

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

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/channel** 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 shall be one object per single service per 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 SPI.

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 within the **"Basic"** profile PI object shall be sorted chronologically by start time.

Where a programme contains multiple time elements, these individual time elements shall be sorted chronologically. The individual programme shall be sorted within the PI 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/channel, service or by day. Devices 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/channel described by this SPI service shall be contained within one object. If the SPI service contains data for more than one ensemble/channel, then there shall be one **"Basic"** profile group information object per ensemble/channel.

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.

## 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 SPI schemas defined in ETSI TS 102 818 [1]. These derived files are well-formed XML documents that are encoded separately using the binary encoding format referred to in clause 4.

The format of the **"Basic"** profile binary 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 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 device to merge the two documents, as detailed in tables 8, 9 and 10.

For an SPI that contains no **"Advanced"** profile data, the **"Basic"** profile document will be the same as the master document.

## 5.4 Attributes required for merging

### 5.4.0 Introduction

To enable an advanced device 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

**Table 8: Service information merging attributes**

| Delivery system | Element | Attribute |
|---|---|---|
| DAB, DRM | serviceInformation | version |
| DAB | serviceInformation.ensemble | id |
| DAB | serviceInformation.ensemble.service.bearer | id |
| DRM | serviceInformation.service.bearer | id |

### 5.4.2 Programme Information

**Table 9: Programme information merging attributes**

| Element | Attribute |
|---|---|
| epg.schedule | version |
| programme | shortId |

### 5.4.3 Group Information

**Table 10: Group information merging attributes**

| Element | Attribute |
|---|---|
| epg.programmeGroups | version |
| epg.programmeGroup | shortId |

A device should not attempt to merge data unless the specified attributes 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 (ETSI EN 301 234 [2]) will be employed as the method for transporting the SPI data.

The mapping of the SPI data onto MOT objects is described in clause 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 (ETSI EN 301 234 [2]).

## 6.2 Maximum object size

Each "**Basic**" profile MOT object shall have a maximum size of 16 Kbytes (16 384 bytes), and a maximum size for the MOT directory of 8 Kbytes (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 Kbytes, then individual services shall be moved to an alternative carousel until the size of the directory object is at or below the maximum size.

If the size of any of the "**Basic**" profile SPI objects exceeds 16 Kbytes (16 384 bytes), then the level of detail within the affected object should be reduced until the size of the 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 SPI 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 SPI is limited to a maximum of 128 kbps.

## 6.4 MOT parameters

### 6.4.0 General

In order to provide useful tracking information for devices 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 SPI specification are given in table 11, and are specified in detail by the following clauses.

The MOT functionality "caching" is optional for both UA provider and device.

If the MOT parameters `ProfileSubset`, `CAInfo`, `ContentName` and/or `UniqueBodyVersion` are used then these parameters should 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 device profile shall be ignored.

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

**Table 11: MOT parameters used to identify individual objects**

| Parameter | Parameter Id | Specified in | Mandatory for UA provider | Mandatory for device | 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 devices ("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 devices shall discard encrypted objects) | Single |
| ScopeStart | 0x25 | SPI | See clause 6.4.6 | No | Single |
| ScopeEnd | 0x26 | SPI | See clause 6.4.7 | No | Single |
| ScopeID | 0x27 | SPI | Yes | No | Single |

## 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 (ETSI EN 301 234 [2]).

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

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

**Table 12: SPI content type/subtype values**

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

## 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 device shall assume that the object is relevant to all profiles supported by the SPI user application that are indicated by the application signalling. The `ProfileSubset` parameter is used as specified in the MOT specification (ETSI EN 301 234 [2]).

NOTE: Basic objects are required by both basic and advanced profile devices, 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 SPI service.

The `ContentName` parameter is used as specified in the MOT specification (ETSI EN 301 234 [2]).

NOTE: 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 (ETSI EN 301 234 [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 devices, which shall have such a capability to de-compress this data. An SPI decoder shall support a maximum window size of 32 kBytes 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 (ETSI EN 301 234 [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 SPI 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 SPI data contained within this Programme Information object. This shall be encoded as defined in clause 4.7.4, 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 SPI 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 SPI data contained within this Programme Information object. This shall be encoded as defined in clause 4.7.4, 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.

This parameter shall be mandatory for Programme Information (PI) objects that are **not** contiguous, 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 *identifier* of the ensemble/channel for which the object contains data, coded as follows:

- When the delivery system is DAB, *ScopeID* is coded as a 24-bit number in the form: <ecc>.<eid>, where <ecc> is the 8-bit Extended Country Code (ECC) and <eid> is the 16-bit Ensemble Identifier (EId) of the ensemble.
- When the delivery system is DRM, *ScopeID* is coded as a 24-bit number in the form: <sid> , where <sid> is the 24-bit Service Identifier (SId) of one of the services in the DRM channel.

If the object contains *Programme Information*, then this parameter indicates the *bearerURI* of the service for which the object contains data, as specified in clause 4.7.6 (i.e. without the domain prefix).

## 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 (ETSI EN 301 234 [2]) and using the appropriate signalling 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 clause 6.2). If the MOT directory exceeds the maximum permitted size then these additional objects shall be transported in an additional carousel.

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

# 7 Signalling

## 7.1 DAB

### 7.1.1 FIG 0/13 (application type) signalling

The use of the SPI application within a DAB data channel shall be indicated by the use of FIG0/13 with a UserApplicationType value of "EPG", see ETSI TS 101 756 [5].

The data is profiled into **"Basic"** and **"Advanced"** profile SPI objects, as described in clause 5.

The user application data field for the SPI user application is a sequence of 1-byte values, each being a `ProfileID`, indicating which profile(s) of the SPI 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 `ProfileID`s are defined in table 13. Any remaining values for the `ProfileID` are reserved for future use and shall not be processed by devices.

If user application specific parameters other than the list of profiles are carried within the user application data field, then the data field shall commence with the list of `ProfileID`s, followed by a single 0x00 after the end of the list of profiles and before the other user application specific parameters.

NOTE: The decoder can easily extract the list of profiles from the user application data field. The list of profiles either ends at the end of the user application data field (no other user application specific parameters) or at a delimiting 0x00 (other user application specific parameters follow after the 0x00), whichever comes first.

**Table 13: Profile IDs**

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

### 7.1.2 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, see ETSI EN 300 401 [3].

## 7.2 DRM

### 7.2.1 Data entity type 5 (data application) signalling

The use of the SPI application within a DRM channel shall be indicated by the use of data entity type 5, see ETSI ES 201 980 [6], with an application domain value of "DAB", see ETSI TS 101 968 [7], and with a UserApplicationType value of "EPG", see ETSI TS 101 756 [5].

The signalling of profile data is as described for DAB, see clause 7.1.1.

### 7.2.2 Data entity type 8 (reference time) signalling

The provision of a correctly broadcast reference time within SDC data entity type 8 is a mandatory requirement of this User Application, see ETSI ES 201 980 [6].

### 7.2.3 FAC data application signalling

The use of the SPI application within a DRM channel may be indicated by the use of the FAC Application identifier, see ETSI ES 201 980 [6], with an application identifier of "Electronic Programme Guide", see ETSI TS 101 968 [7].

# Annex A (normative): Profiling tables

## A.1 Elements and attributes that are transmitted in the "Basic" profile

### A.1.0 General

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.

### A.1.1 Service Information (SI)

**Table A.1: Basic profile service information (DAB)**

| Element | Attribute | Required? |
|---|---|---|
| serviceInformation | | R |
| | version | O |
| serviceInformation.ensemble | | R |
| | id | R |
| serviceInformation.ensemble.shortName | | R |
| | xml:lang | R2 |
| serviceInformation.ensemble.mediumName | | R |
| | xml:lang | R2 |
| 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 |
| serviceInformation.ensemble.service.bearer | | R |
| | id | R |
| 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 |
| serviceInformation.ensemble.service.radiodns | | O |
| | fqdn | R1 |
| | serviceIdentifier | R1 |

**Table A.2: Basic profile service information (DRM)**

| Element | Attribute | Required? |
|---|---|---|
| serviceInformation | | R |
| | version | O |
| serviceInformation.service | | R |
| serviceInformation.service.bearer | | R |
| | id | R |
| serviceInformation.service.shortName | | R |
| | xml:lang | R2 |
| serviceInformation.service.mediumName | | R |
| | xml:lang | R2 |
| serviceInformation.service.mediaDescription | | O |
| serviceInformation.service.mediaDescription.multimedia | | O |
| | type | O |
| | mimeValue | O |
| | xml:lang | R2 |
| | url | R1 |
| | width | O |
| | height | O |
| serviceInformation.service.radiodns | | O |
| | fqdn | R1 |
| | serviceIdentifier | R1 |

## A.1.2 Programme Information (PI)

**Table A.3: Basic profile programme information**

| Element | Attribute | Required? |
|---|---|---|
| epg | | R |
| epg.schedule | | R |
| | version | O |
| epg.schedule.scope | | O |
| | startTime | R1 |
| | stopTime | R1 |
| epg.schedule.scope.serviceScope | | O |
| | id | R1 |
| epg.schedule.programme | | R |
| | shortId | R |
| | recommendation | R2 |
| | broadcast | R2 |
| epg.schedule.programme.mediumName | | R |
| | xml:lang | R2 |
| epg.schedule.programme.longName | | O |
| | xml:lang | R2 |
| epg.schedule.programme.location | | R |
| epg.schedule.programme.location.time | | R |
| | time | R |
| | duration | R |
| epg.schedule.programme. location.bearer | | R2 |
| | id | R2 |
| epg.schedule.programme.mediaDescription | | O |
| epg.schedule.programme.mediaDescription.shortDescription | | O |
| | xml:lang | R2 |
| epg.schedule.programme.genre | | O |
| | href | R1 |
| | type | R2 |
| epg.schedule.programme.memberof | | O |
| | shortId | R1 |
| | index | O |

### A.1.3 Group Information (GI)

**Table A.4: Basic profile group information**

| Element | Attribute | Required? |
|---|---|---|
| epg | | R |
| epg.programmeGroups | | R |
| | version | O |
| epg.programmeGroups.programmeGroup | | R |
| | shortId | R |
| | type | O |
| | numofItems | O |
| epg.programmeGroups.programmeGroup.mediumName | | R |
| | xml:lang | R2 |
| epg.programmeGroups.programmeGroup.longName | | O |
| | xml:lang | R2 |
| epg.programmeGroups.programmeGroup.genre | | O |
| | href | R1 |
| | type | R2 |
| epg.programmeGroups.programmeGroup.memberof | | O |
| | shortId | R1 |
| | index | O |

## A.2 Elements and attributes that are transmitted in the "Advanced" profile

The "Advanced" profile consists of any valid elements and attributes from the SPI XML specification.

NOTE: A broadcaster, for their particular SPI service, may choose to move elements and attributes from the Basic profile to the Advanced profile to increase efficiency.

# Annex B (informative): Profiling examples

## B.1 Profiling/fragmenting example 1

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

**Table B.1: Profiling example 1**

| 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 |

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

Consequently, the number of objects generated is:

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

**A total of 134 objects.**

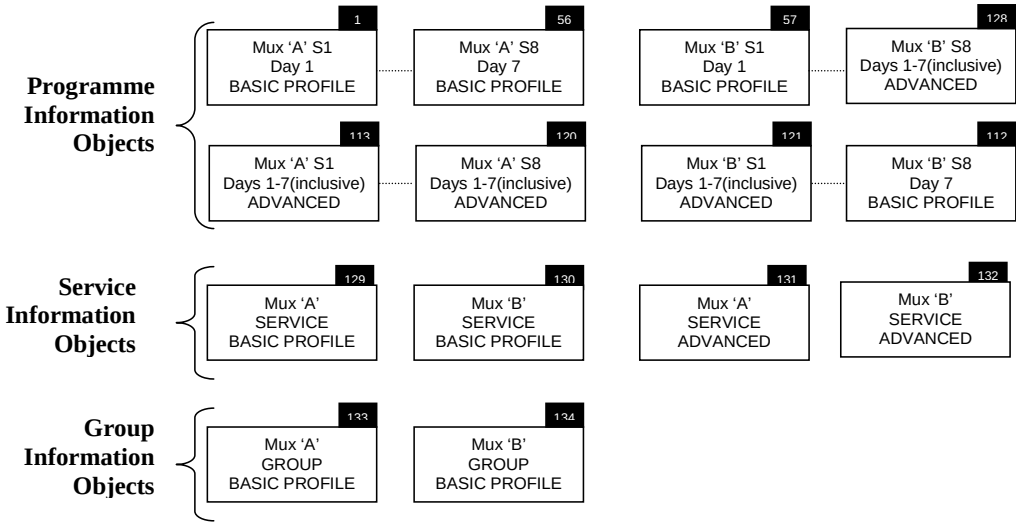



**Figure B.1: Overview of objects**

# B.2 Profiling/fragmenting example 2

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

**Table B.2: Profiling example 2**

| Service Provider | Service | SPI channel |
|---|---|---|
| Service Provider A | Service 1 | SPI 1 |
| Service Provider A | Service 2 | SPI 1 |
| Service Provider A | Service 3 | SPI 1 |
| Service Provider A | Service 4 | SPI 1 |
| Service Provider B | Service 5 | SPI 2 |
| Service Provider B | Service 6 | SPI 2 |
| Service Provider C | Service 7 | SPI 3 |
| Service Provider D | Service 8 | SPI 4 |
| Service Provider E | Service 9 | SPI 5 |

Five service providers, A to 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 SPI for all their services in packet mode. So there are five SPI services, one including four services, one including two services and three including one programme service. SPIs for multiple programme services are transmitted in packet mode, SPIs for single programme services are transmitted in PAD.

No SPI data from other ensembles is incorporated.

Consequently, the number of objects generated is:

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

**A total of 87 objects.**

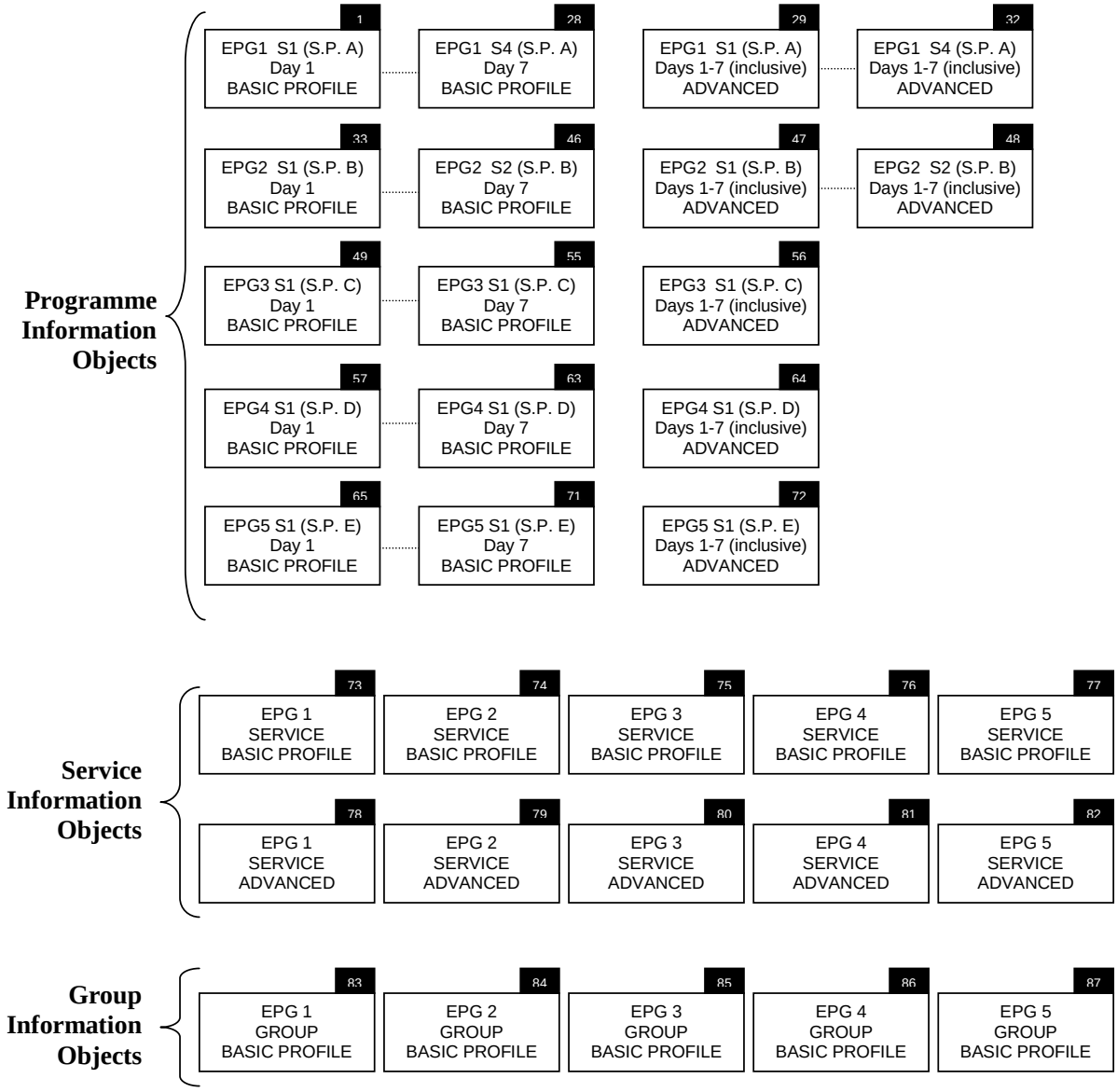



**Figure B.2: Overview of objects**

# Annex C (informative): Binary encoding example

```
<?xml version="1.0" encoding="UTF-8"?>
<epg xmlns="http://www.worlddab.org/schemas/spi/31" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.worlddab.org/schemas/spi/31 spi_31.xsd">
    <schedule version="1">
        <scope startTime="2003-12-18T17:00:00Z" stopTime="2003-12-18T18:00:00Z">
            <serviceScope id="dab:ce1.ce15.c224.0"/>
        </scope>
        <programme shortId="16442449" id="crid://bbc.co.uk/4969758988">
            <mediumName>PM</mediumName>
            <location>
                <time time="2003-12-18T17:00:00Z" duration="PT1H"/>
            </location>
        </programme>
    </schedule>
</epg>
```

These are the bytes for the encoded binary object:

**Table C.1: Binary encoded example**

| Bytes | Description |
|---|---|
| 02 | **<epg>** |
| 52 | length = 82 bytes |
| 21 | **<schedule>** |
| 50 | length = 80 bytes |
| 24 | **<scope>** |
| 16 | length = 22 |
| 80 | **<startTime>** |
| 04 | length = 4 |
| 33 BF C4 40 | 2003-12-18T17:00:00 *(short form)* |
| 81 | **<stopTime>** |
| 04 | length = 4 |
| 33 BF C4 80 | 2003-12-18T18:00:00 *(short form)* |
| 25 | **<serviceScope>** |
| 08 | length = 8 |
| 80 | id attribute |
| 06 | length = 6 |
| 40 E1 CE 15 C2 24 | e1.ce15.c224.0 |
| 1C | **<programme>** |
| 36 | length = 54 |
| 81 | shortId attribute |
| 03 | length = 3 |
| FA E4 51 | 16442449 |
| 80 | id attribute |
| 1B | length=27 |
| 63 72 69 64 3A 2F 2F 62 62 | crid://bbc.co.uk/4969758988 |
| 63 2E 63 6F 2E 75 6B 2F 34 | |
| 39 36 39 37 35 38 39 38 38 | |
| 11 | **<mediumName>** |
| 04 | length = 4 |
| 01 | CDATA |
| 02 | length = 2 |
| 50 4D | PM |
| 19 | **<location>** |
| 0C | length = 12 |
| 2C | **<time>** |
| 0A | length = 10 |
| 80 | time attribute |
| 04 | length = 4 |
| 33 BF C4 40 | 2003-12-18T17:00:00 *(short form)* |
| 81 | duration attribute |
| 02 | length = 2 |
| 0E 10 | 3600 |

# 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 devices for internal use (e.g. to signify invalid elements).

**Table D.1: Element tags**

| Child element | Possible parent elements | Tag | Defined in clause |
|---|---|---|---|
| epg | Top-level | 0x02 | 4.3.1 |
| serviceInformation | Top-level | 0x03 | 4.3.1 |
| tokenTable | epg, serviceInformation | 0x04 | 4.9 |
| - | Not used | 0x05 | N/A |
| defaultLanguage | epg, serviceInformation | 0x06 | 4.11 |
| 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 |
| - | Not used | 0x15 | N/A |
| 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 | mediaDescription | 0x1A | 4.3 |
| longDescription | mediaDescription | 0x1B | 4.3 |
| programme | schedule | 0x1C | 4.3 |
| programmeGroups | epg | 0x20 | 4.3 |
| schedule | epg | 0x21 | 4.3 |
| - | Not used | 0x22 | N/A |
| programmeGroup | programmeGroups | 0x23 | 4.3 |
| scope | schedule | 0x24 | 4.3 |
| serviceScope | scope | 0x25 | 4.3 |
| ensemble | serviceInformation | 0x26 | 4.3 |
| - | Not used | 0x27 | N/A |
| service | ensemble, serviceInformation | 0x28 | 4.3 |
| bearer (see note 2) | service | 0x29 | 4.3 |
| - | Not used | 0x2A | N/A |
| multimedia | mediaDescription | 0x2B | 4.3 |
| time | location | 0x2C | 4.3 |
| bearer | location, onDemand | 0x2D | 4.3 |
| programmeEvent | programme | 0x2E | 4.3 |
| relativeTime | location | 0x2F | 4.3 |
| - | Not used | 0x30 | N/A |
| radiodns | service | 0x31 | 4.3 |
| geolocation | service, bearer | 0x32 | 4.3 |
| country | geolocation | 0x33 | 4.3 |
| point | geolocation | 0x34 | 4.3 |
| polygon | geolocation | 0x35 | 4.3 |
| onDemand | programme, programmeEvent | 0x36 | 4.3 |
| presentationTime | onDemand | 0x37 | 4.3 |
| acquisitionTime | onDemand | 0x38 | 4.3 |
| NOTE 1: Deprecated elements are marked "Not used" for reasons of backwards compatibility. | | | |

| Child element | Possible parent elements | Tag | Defined in clause |
|---|---|---|---|
| NOTE 2: In earlier versions of ETSI TS 102 371 [i.1], this child element was called serviceID. | | | |

# Annex E (normative): Attribute tags

The encoding of attributes is defined in clause 4.4. Additional encoding rules clauses are referenced below.

**Table E.1: Common attribute tags**

| Element | Attribute | Tag | Defined in clause |
|---|---|---|---|
| genre | href | 0x80 | 4.12 |
| | type | 0x81 | 4.6 |
| keywords | xml:lang | 0x80 | 4.8.1 |
| link | uri | 0x80 | 4.4.0 |
| | mimeValue | 0x81 | 4.7.3 |
| | xml:lang | 0x82 | 4.8.1 |
| | description | 0x83 | 4.4.0 |
| | expiryTime | 0x84 | 4.7.4 |
| 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 |
| multimedia | mimeValue | 0x80 | 4.7.3 |
| | xml:lang | 0x81 | 4.8.1 |
| | url | 0x82 | 4.4.0 |
| | type | 0x83 | 4.6 |
| | width | 0x84 | 4.8.5 |
| | height | 0x85 | 4.8.5 |
| bearer | id | 0x80 | 4.7.6 |
| | Not used | 0x81 | N/A |
| | url (see note 2) | 0x82 | 4.4.0 |
| geolocation | xml:id | 0x80 | 4.4.0 |
| | ref | 0x81 | 4.4.0 |
| NOTE 1: Deprecated attributes are marked "Not used" for reasons of backwards compatibility. | | | |
| NOTE 2: The url tag is used in place of the id tag for bearers in the http: domain, see clause 4.7.6. | | | |

**Table E.2: SI attribute tags**

| Element | Attribute | Tag | Defined in clause |
|---|---|---|---|
| serviceInformation (see note 1) | version | 0x80 | 4.8.3 |
| | creationTime | 0x81 | 4.7.4 |
| | originator | 0x82 | 4.4.0 |
| | serviceProvider | 0x83 | 4.4.0 |
| | Not used | 0x84 | N/A |
| ensemble | id | 0x80 | 4.17.1 |
| | Not used | 0x81 | N/A |
| service | version | 0x80 | 4.8.3 |
| | Not used | 0x81 | N/A |
| | Not used | 0x82 | N/A |
| | Not used | 0x83 | N/A |
| | Not used | 0x84 | N/A |
| radiodns | fqdn | 0x80 | 4.4.0 |
| | serviceIdentifier | 0x81 | 4.4.0 |
| NOTE 1: This scheme does not encode any of the schema information (e.g. the "xmlns" attribute). | | | |
| NOTE 2: Deprecated attributes are marked "Not used" for reasons of backwards compatibility. | | | |

**Table E.3: PI and GI attribute tags**

| Element | Attribute | Tag | Defined in clause |
|---|---|---|---|
| epg (see note 1) | Not used | 0x80 | N/A |
| programmeGroups | version | 0x80 | 4.8.3 |
| | creationTime | 0x81 | 4.7.4 |
| | originator | 0x82 | 4.4.0 |
| programmeGroup | id | 0x80 | 4.7.1 |
| | shortId | 0x81 | 4.7.2 |
| | version | 0x82 | 4.8.3 |
| | type | 0x83 | 4.6 |
| | numOfItems | 0x84 | 4.8.4 |
| schedule | version | 0x80 | 4.8.3 |
| | creationTime | 0x81 | 4.7.4 |
| | originator | 0x82 | 4.4.0 |
| scope | startTime | 0x80 | 4.7.4 |
| | stopTime | 0x81 | 4.7.4 |
| serviceScope | id | 0x80 | 4.7.6 |
| programme | id | 0x80 | 4.7.1 |
| | shortId | 0x81 | 4.7.2 |
| | version | 0x82 | 4.8.3 |
| | recommendation | 0x83 | 4.6 |
| | broadcast | 0x84 | 4.6 |
| | Not used | 0x85 | N/A |
| | xml:lang | 0x86 | 4.8.1 |
| | Not used | 0x87 | N/A |
| programmeEvent | id | 0x80 | 4.7.1 |
| | shortId | 0x81 | 4.7.2 |
| | version | 0x82 | 4.8.3 |
| | recommendation | 0x83 | 4.6 |
| | broadcast | 0x84 | 4.6 |
| | Not used | 0x85 | N/A |
| | xml:lang | 0x86 | 4.8.1 |
| | Not used | 0x87 | N/A |
| time | time | 0x80 | 4.7.4 |
| | duration | 0x81 | 4.7.5 |
| | actualTime | 0x82 | 4.7.4 |
| | actualDuration | 0x83 | 4.7.5 |
| relativeTime | time | 0x80 | 4.7.5 |
| | duration | 0x81 | 4.7.5 |
| | actualTime | 0x82 | 4.7.5 |
| | actualDuration | 0x83 | 4.7.5 |
| memberOf | id | 0x80 | 4.7.1 |
| | shortId | 0x81 | 4.7.2 |
| | index | 0x82 | 4.8.2 |
| presentationTime | start | 0x80 | 4.7.4 |
| | end | 0x81 | 4.7.4 |
| | duration | 0x82 | 4.7.5 |
| acquistionTime | start | 0x80 | 4.7.4 |
| | end | 0x81 | 4.7.4 |

NOTE 1: This scheme does not encode any of the schema information (e.g. the "xmlns" attribute).
NOTE 2: Deprecated attributes are marked "Not used" for reasons of backwards compatibility.

# Annex F (normative): Enumerated types

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

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

**Table F.1: Enumerated type values**

| Element | Attribute | Value | Tag |
|---|---|---|---|
| programmeGroup | type | series | 0x02 |
| | | show | 0x03 |
| | | programConcept | 0x04 |
| | | magazine | 0x05 |
| | | programCompilation | 0x06 |
| | | otherCollection | 0x07 |
| | | otherChoice | 0x08 |
| | | topic | 0x09 |
| programme, programmeEvent | broadcast | on-air | 0x01 |
| | | off-air | 0x02 |
| programme, programmeEvent | recommendation | no | 0x01 |
| | | yes | 0x02 |
| multimedia | type | logo_unrestricted | 0x02 |
| | | Not used | 0x03 |
| | | logo_colour_square | 0x04 |
| | | Not used | 0x05 |
| | | logo_colour_rectangle | 0x06 |
| genre | type | main | 0x01 |
| | | secondary | 0x02 |
| | | other | 0x03 |
| NOTE: Deprecated enumeration values are marked "Not used" for reasons of backwards compatibility. | | | |

# History

| Document history | | |
|---|---|---|
| V1.1.1 | January 2005 | Publication |
| V1.2.1 | February 2006 | Publication |
| V1.3.1 | July 2008 | Publication |
| V3.1.1 | January 2015 | Publication |
| V3.2.1 | May 2016 | Publication |