ETSI TS 102 371 V3.3.1 (2023-12)

TECHNICAL SPECIFICATION

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

Reference

RTS/JTC-DAB-109

Keywords

audio, broadcasting, DAB, digital, DRM, EPG

Contents

Intellectual Property Rights

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. The SPI permits a wide range of features to be supported, but not all such features are applicable to broadcast receivers. Using only the basic SPI profile allows:

- navigation and selection of services, including with logo images;
- the display of schedules for programmes from a range of services.

Addition of the advanced SPI profile allows:

- 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 (V3.5.1 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]) is profiled, compressed 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, voice control provisions have been added and the text has been restructured to reflect the development of SPI service delivery and devices.

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 (V3.5.1 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 103 177: "Digital Audio Broadcasting (DAB); Filecasting; User application specification".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms 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.2]

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 Symbols

Void.

3.3 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 Introduction

4.1 The SPI service

The provision of an SPI service can enhance the user experience. The ability to provide visual identification of services though the use of logos can aid brand recognition and service selection, particularly in vehicles where rapid recognition helps to reduce driver distraction. The details of individual programmes can also be provided. The SPI is a rich resource (see ETSI TS 102 818 [1]) designed for a wide variety of uses for both broadcast and IP delivery and for both consumer devices and business applications.

Broadcast receivers tend towards more functional user interfaces with limited interactivity compared to connected devices. For this reason, the elements of the SPI are divided into two profiles - "basic" and "advanced". It is recommended that only the core functions of the SPI service, those in the basic profile, are provided over broadcast bearers: this ensures that the SPI service is reliable and practical. The separation of appropriate metadata from the complete hybrid SPI in XML format can be achieved by using transformation scripts, such as XSLTs, and these can also be used to combine together SPI data from different service providers so that all services in a DAB ensemble can be delivered in a single SPI service.

A broadcast SPI service consists of a set of objects, delivered using an MOT carousel. Different kinds of SPI service are possible, but all will include Service Information (SI) delivered by a basic profile SI object. A typical SPI service will include the logos of services to provide brand visualization, and these logo objects are included in the same MOT carousel. Some broadcasters may also want to provide programme schedules, so that the details of each programme can be displayed. This Programme Information (PI) is delivered by basic profile PI objects, one for each day of each service. Group Information (GI), which provides connections between similar programmes, may also be broadcast, as can advanced profile SI and PI objects, but this type of information is not especially suited to display on broadcast receivers. The decisions on what to include in the broadcast SPI service lie with the service providers, but minimizing the overall size of the SPI service increases the speed and reliability of delivery.

It is strongly recommended that every DAB ensemble and every DRM channel should include an SPI service that provides the logos for all the services in the ensemble or channel. This can be achieved using very little of the overall capacity, for example, with DAB, an 8 kbps packet data channel at EEP-3A uses only 6 of the total 864 Capacity Units. This simple approach enhances the attractiveness of the broadcast delivery.

However, device manufacturers should be aware that regulatory or commercial issues may preclude the provision of a single, DAB ensemble-wide SPI service, and so they need to make provision for SPI services delivered in various ways. It is possible for there to be more than one packet mode SPI service in a DAB ensemble, each covering a subset of the services. It is also possible, subject to maximum size constraints, to provide SPI information for services in other DAB ensembles: in this case there will be multiple basic profile SI objects in the MOT carousel. The SPI can also be delivered as PAD if an individual service provider wishes to provide an SPI service but cannot access a packet data service.

4.2 Compression

The SPI service for broadcast is derived from the SPI XML files and the logo image files. In order to deliver the SPI efficiently, the XML is compressed using a binary encoding process that replaces the textual names of XML elements and attributes with single byte "tags". Additionally, frequently used strings of text can be replaced by tokens, and default values do not need to be encoded. All of these measures are designed to reduce the size of the objects that are transmitted and so decrease the amount of time taken to send all the objects. Because users may tune in at any time, the SPI service is continually repeated.

The typical broadcast SPI consists of only basic profile objects and logo objects. The basic profile objects are each limited in size to 16 kbytes in order to allow devices with limited resources to process them. There is no size limit for advanced profile objects, and, because they are likely to contain a high proportion of text, they may have additional compression applied through the use of the GZIP (deflate) compression scheme. However, it should be recognized that not all types of device will have the resources to process advanced profile objects, and their inclusion will increase the overall size of the MOT carousel and thus extend the cycle time at a given datarate. Logo objects are not size limited but should also be optimized to reduce the file size by removing details not required for display.

5 Binary encoding

5.1 General

An SPI service consists of a number of data objects which are delivered by use of the MOT protocol [2]. To minimize the total amount of data to be delivered, the SPI data objects are prepared for broadcast by applying a binary compression.

Broadcast SPI services may be delivered over DAB [3] and DRM [6]. There are some differences in the encoding of some information, especially relating to service identification. For this reason, the present document uses the term "delivery system" to distinguish the required encoding according to whether the SPI service is being delivered over DAB or DRM. The delivery system is not identified, however, because decoders can easily determine which system delivered the binary data.

The binary encoding uses a tag-length-value encoding. All data objects are encoded using a tag field, a length field (indicating the length of the value field within the data object) and the value field which contains the payload. This enables decoders to easily skip data objects that are not wanted or unrecognised.

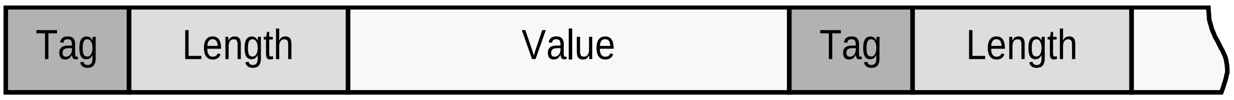


Figure 1: Tag-length-value encoding scheme

The first data object in an SPI binary object is the top-level object holding child objects in the value field. These child objects may each contain child objects, so that an SPI object forms a tree of data objects. Data objects are either an element or an attribute. Elements may hold child elements and attributes. Attributes do not hold any child objects and thus can be thought forming leaves on the object tree.

Examples of binary encoding are given in annex C.

5.2 Syntax of the binary object tree

5.2.1 Syntax specification

The specifications of syntax in this clause 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

5.2.2 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() {`
`    `**`top_level_element()`**
`}`		

top_level_element: A top level element as defined in clause 5.3.2.

5.2.3 Elements syntax

All elements are encoded as defined in table 3.

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 are reserved for future use; undefined tags and their associated content shall be ignored 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 and child elements.

5.2.4 Attributes syntax

All attributes use the same encoding, as defined here.

Table 4: 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 `0x01` (see clause 5.3.1) and those defined in annex E. Attributes with tags that are not defined are reserved for future use and shall be ignored 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 (see clause 5.4.2), an enumerated data value (see clause 5.4.3), an integer (see clause 5.4.4) or a specific encoding (see clause 5.4.5).

5.3 Elements encoding

5.3.1 General

The data value field of elements may carry attributes and child elements. These shall be encoded in the following order:

1) Attributes.

2) String token table (top-level elements only).

3) Default language (top-level elements only).

4) Child elements.

5) CDATA content.

All CDATA and text strings inside elements shall be encoded as an attribute as defined in clause 5.2.4. The **attribute tag** shall be `0x01`. Entity references in text strings shall be replaced (e.g. &amp; shall be replaced with the & character).

All CDATA and 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.

5.3.2 Top-level elements

5.3.2.1 General

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 5.2) as its first element spanning the entire binary object. The possible values of the *element_tag* for top-level elements are defined in table 5. Top-level elements with tags that are not defined here are reserved for future use; these tags and their associated content shall be ignored by devices.

Table 5: Top-level element tags

Element	Tag
epg	0x02
serviceInformation	0x03

The top-level elements may optionally contain a string token table (see clause 5.5) and a default language tag (see clause 5.6). If present, these shall appear after the top-level element's attributes and before the child elements and CDATA content.

5.3.2.2 serviceInformation element

The structure of the XML 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 5.3.2.3.
- 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**.

5.3.2.3 DAB ensemble element encoding

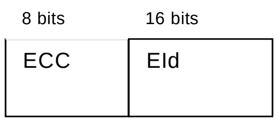
The **ensemble** element is no longer defined in the XML specification but for compatibility reasons, it is still required in the binary encoding when the delivery system is DAB. Two methods of providing the minimum necessary information are possible:

- 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 2: 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.

5.3.3 serviceScope element

When the domain of the **id** attribute matches the delivery system, it shall be encoded according to clause 5.4.5.1.

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

5.3.4 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 5.4.5.1.

As a child element of **onDemand**, when the domain of the **id** attribute is http:, it shall be encoded according to clause 5.4.5.1, 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.

5.3.5 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.

5.3.6 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.

5.3.7 point and polygon elements

The **point** element carries a coordinate pair; the **polygon** element carries a sequence of coordinate pairs. The decoder can derive the number of coordinate pairs from the length field of the element. 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).

5.3.8 genre element

The **href** attribute shall be encoded as described in clause 5.4.5.4.

The **type** attribute shall be encoded as an enumerated attribute as described in clause 5.4.3.

5.3.9 Unencoded elements

The following hybrid XML elements are not encoded for broadcast delivery: **credits, credit, organization, person, services**, **serviceProvider**, **serviceGroups**, **serviceGroup** (except as the "ensemble" element, see clause 5.3.2.3), **serviceGroupMember**.

5.4 Attributes encoding

5.4.1 Introduction

Many attributes are encoded according to their xml data type, but special encodings exist to provide greater data compression.

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.

5.4.2 String attributes

The **attribute_tag** is set according to the definition in annex E. The **attribute_data_byte**s contain the string.

All 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. Entity references in text strings shall be replaced (e.g. &amp; shall be replaced with the & character).

5.4.3 Enumerated data values

The **attribute_tag** is set according to the definition in annex E. The **attribute_data_byte**s contain a single byte; the value of this byte is specific to that particular attribute and is defined in annex F.

5.4.4 Integer data values

The following attributes carry a single integer value in the attribute data bytes:

- The attributes **height**, **index**, **numOfItems**, **version**, **width** are coded as a 16-bit unsigned integer;
- The attribute **shortId** is coded as a 24-bit unsigned integer.

The **attribute_tag** is set according to the definition in annex E. The **attribute_data_byte**s contain the integer.

5.4.5 Specific coding of attributes

5.4.5.1 bearerURI type

5.4.5.1.1 General

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

5.4.5.1.2 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 3: 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).

5.4.5.1.3 drm: domain

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

24 bits
SId

Figure 4: bearerURI encoding for DRM

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

5.4.5.1.4 http: domain

Attributes defined as bearerURI type in the http: domain are encoded using the "url" tag as a string attribute (see clause 5.4.2).

5.4.5.2 timepoint type

The 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 so that the format is the same as the time delivered in the FIC. A conversion shall be performed to convert the time specified correctly.

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.

Attributes defined as timePoint type shall be encoded as follows:

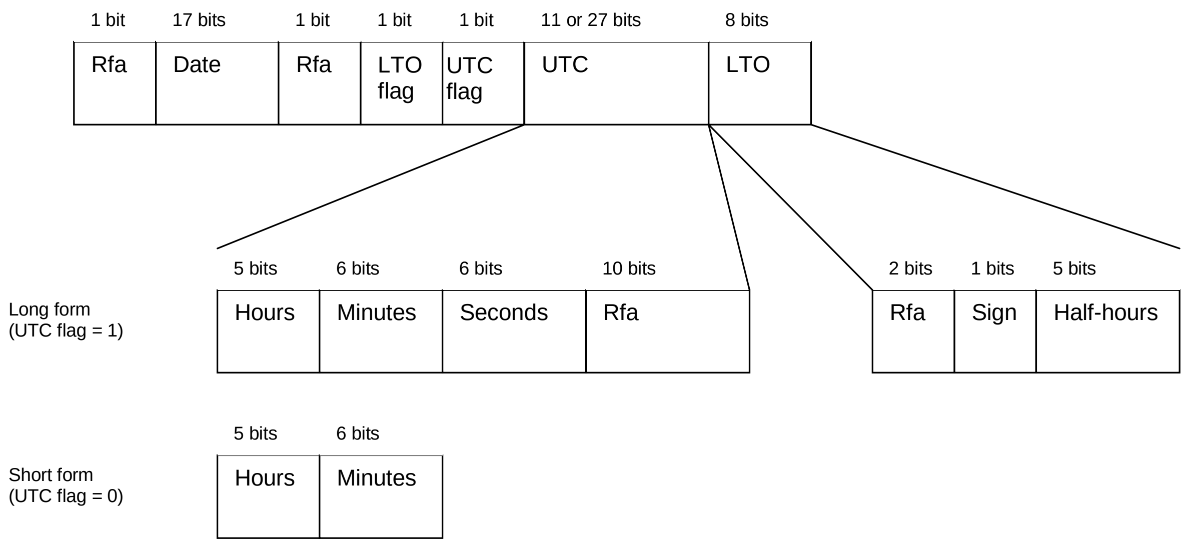


Figure 5: Date and time encoding (LTO flag == 1)

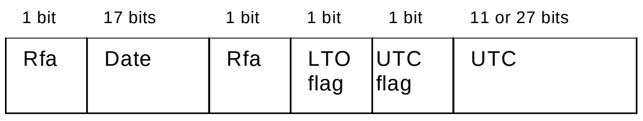


Figure 6: Date and time encoding (LTO flag == 0)

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 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. 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.

5.4.5.3 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).

5.4.5.4 href attribute

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 7: 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.

5.5 String token table

5.5.1 General

The string token table is a way of compressing textual information by assigning a token to a string definition. It can only occur within the two top-level elements (**epg** and **serviceInformation**) and, if present, it shall occur after the attributes and before any elements and CDATA content.

Tokens shall not be used to compress the **url** attributes of **multimedia** elements as this prevents simple string comparison of the contentNames of multimedia (logo) objects.

A maximum of 16 tokens are allowed per table. The 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.

The token table applies to all character data within the current binary object. It shall be encoded as defined in clause 5.2.4, with the following provisos.

attribute_tag: This shall always be `0x04`.

attribute_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.

5.5.2 Tokens

Entries in the string 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 6: 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 bytes that make up the token string.

5.6 Default language

Providing the default language allows every instance where an **xml:lang** attribute with the same value as the **default language** occurs to be omitted from the encoding.

The **default language** shall 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 child elements and CDATA content. The default language applies to the entire binary object.

The **default language** string is identical to the value of the **xml:lang** that it represents.

It shall be encoded as defined in clause 5.2.4 with the following provisos:

attribute_tag: This shall always be `0x06`.

attribute_data_byte: The bytes that make up the default language string.

Whenever a decoder finds a missing **xml:lang** attribute for an element, 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.

6 Transportation

6.1 Basics

MOT in Directory Mode, see ETSI EN 301 234 [2], shall be used as the method for transporting the SPI data. The SPI may be transported in packet mode or in PAD.

The SPI data is divided and carried as MOT objects. Multimedia files (typically service logos) that are referenced by the SPI shall be carried in the same MOT carousel.

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 service is limited to a maximum of 128 kbps.

6.2 Maximum object size

The MOT directory object has a maximum size of 8 kbytes (8 192 bytes).

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]).

Basic profile MOT objects shall have a maximum size of 16 kbytes (16 384 bytes).

The size of advanced profile objects is not restricted.

The size of multimedia objects is not restricted, but should be kept as small as possible. Optimization should be used to reduce the file size.

The SPI service shall be designed in such a way that the maximum object sizes are respected. The size of the MOT directory object is related to the number of services in the SPI data and to the length of contentNames. Consideration should be given to these factors so as not to exceed the maximum size of the MOT directory object. Similarly, the size of the basic profile SI object is related to the number of services, and when multimedia files (e.g. service logos) are included, to the length of the multimedia file names, and hence the length of the contentNames of the multimedia objects. For basic profile PI files, the level of detail will determine the object size. Consideration should be given to these factors to ensure the basic profile objects are at or below the maximum size.

6.3 Transformation into MOT objects

6.3.1 Basics

ETSI TS 102 818 [1] clause 9.2 describes the conventions for the XML documents used for the broadcast SPI service. These documents represent the *master* XML descriptions of the SPI service, divided into SI, PI and GI documents.

The SPI service shall always include Service Information (SI), and may include Programme Information (PI) and Group Information (GI).

The possible master XML documents for a broadcast SPI service are as follows:

- one SI document per DAB ensemble or DRM channel, named YYYYMMDD_<name>_SI.xml;
- one PI document per service per day, named: YYYYMMDD_<service>_PI.xml;
- one GI document per ensemble/channel, named YYYYMMDD_<name>_GI.xml.

Where YYYY represents the year, MM represents the month, DD the day and <name> is a string identifying the ensemble/channel and <service> is a string identifying the service.

Each SI document contains a single <serviceInformation> element.

Each PI document contains a single <schedule> element.

Each GI document contains a single <programmeGroups> element.

A broadcast SPI service shall have at a minimum one master SI document.

For efficient broadcast, these master XML documents are partitioned into **basic** and **advanced** profile documents before binary encoding to create MOT objects.

NOTE: The XML filenames are only used for processing, they are not delivered to devices.

6.3.2 Scheme to create the profile data

The XML elements which are deemed essential to a broadcast SPI service make up the basic profile. Within the basic profile, some XML elements are required, whilst others are optional. Clause A.1 identifies the permitted content of the basic profile for SI, PI and GI content. All devices supporting the SPI service shall decode basic profile objects. The remaining XML elements make up the advanced profile.

The basic and advanced profile data is derived from each *master* XML document that conforms to the SPI schemas defined in ETSI TS 102 818 [1]. These derived files are well-formed XML documents that are subsequently encoded separately into MOT objects using the binary encoding format defined in clause 5.

Each basic profile XML document has the same tree as its master XML document, but includes only the basic profile elements and attributes. Each element shall have the same nesting and order as in the master document.

Each advanced profile XML document has the same tree as its master XML document, but has a hollowed out structure because content in the basic profile is not duplicated. Tables 7, 8 and 9 define the required core elements and attributes of the advanced profile XML documents that allow devices that support the advanced profile to merge the binary coded basic and advanced profile objects appropriately. Each advanced profile XML document is built from its master XML document, retaining the required core elements and attributes defined in tables 7, 8 and 9 but removing the following data:

- elements and attributes 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.

Table 7: Service information core attributes

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

Table 8: Programme information core attributes

Element	Attribute
epg.schedule	version
programme	shortId

Table 9: Group information core attributes

Element	Attribute
epg.programmeGroups	version
epg.programmeGroup	shortId

A device supporting the advanced profile 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.

NOTE: An SPI master XML document that contains no advanced profile data will produce an empty advanced profile XML document and the basic profile XML document will be the same as the master XML document.

For examples of the transformation of the data, see annex B.

6.3.3 Encoding

Each derived XML document is converted into an MOT object using the binary encoding scheme defined in clause 5.

When advanced profile objects exist, they shall be carried in the same MOT data carousel as the corresponding basic profile objects.

6.4 MOT parameters for SPI objects

6.4.1 General

In order to provide useful tracking information for devices to efficiently download and cache data, MOT parameters are 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 carry SI, PI or GI information are given in table 10, and are specified in detail by the following clauses.

Any parameters that are encountered that are not understood by a given device profile shall be ignored.

The MOT parameters detailed in table 10 are used to identify the content of the individual SPI objects.

Table 10: 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, see clause 6.4.3)	Yes	Single
CAInfo	`0x23`	MOT	No (but the parameter shall be used for all objects encrypted on MOT level, see clause 6.4.4)	Yes (non CA capable devices shall discard encrypted objects)	Single
ContentName	`0x0C`	MOT	Yes (see clause 6.4.5)	Yes	Single
CompressionType	`0x11`	MOT	No (but the parameter shall be used for all objects compressed on MOT transport level, see clause 6.4.6)	Yes for "Advanced" profile devices ("basic" profile objects cannot be compressed using GZIP)	Single
ScopeStart	`0x25`	SPI	No (but the parameter shall be used for all PI objects, see clause 6.4.7)	No	Single
ScopeEnd	`0x26`	SPI	No (but the parameter shall be used for all PI objects, see clause 6.4.8)	No	Single
ScopeID	`0x27`	SPI	Yes (see clause 6.4.9)	No	Single

6.4.2 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 11. The `ContentType` of all SPI-specific data is the "application" value (7).

NOTE: `ContentSubType` values are only unique within the (SPI) application. Other applications could use the same values for other content types.

Table 11: 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

6.4.3 ProfileSubset

When the carousel contains only basic profile objects, this parameter shall not be used.

When the carousel contains both basic profile objects and advanced profile objects, then the advanced profile objects only shall use the `ProfileSubset` parameter set to indicate the Advanced profile using the profile IDs defined in table 12.

NOTE: Devices that decode only the basic profile can easily identify advanced profile objects using the `ProfileSubset` parameter and can ignore them.

6.4.4 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.

6.4.5 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]).

The `ContentName` shall use ISO/IEC 10646 [4] with UTF-8 encoding.

NOTE 1: The `ContentName` itself should be kept to a minimum, as it contains no useful information other than a unique identifier.

NOTE 2: Some legacy encoders set the `ContentName` in a particular way, including the use of certain extensions (e.g. ".EHB", ".EIB", etc.). Devices cannot derive any significance as to the content of an object from its `ContentName`: the MOT parameters detailed in table 10 are used to identify the content of the individual objects.

6.4.6 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 GZIP 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.

6.4.7 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 5.4.5.2, 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.

6.4.8 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 5.4.5.2, 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.

6.4.9 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 5.4.5 (i.e. without the domain prefix).

6.5 MOT parameters for other objects

The MOT carousel may also contain multimedia content. Typically, these objects will be the service logos. 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.

The `ContentName` of all other objects carried in an SPI MOT carousel shall use ISO/IEC 10646 [4] with UTF-8 encoding.

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

6.6 Carousel cycle time and object repetition

In order to assist devices, it is strongly recommended to ensure that the MOT directory object is transmitted at regular intervals within the cycle time of the overall carousel: a repetition rate of approximately once per minute is suggested. Especially in the case of an SPI service carrying service logos, it is also recommended that the basic profile SI object is transmitted after every transmission of the MOT directory object. In this way, devices are able to determine the `contentNames` of the objects they require in a timely fashion.

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 FIG 0/13 with a UserApplicationType value of "EPG", see ETSI TS 101 756 [5].

The user application data field for the SPI user application is a sequence of 1-byte values, each being a `ProfileID`, indicating the profile(s) of the SPI service. 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 12. Any remaining values for the `ProfileID` are reserved for future use.

Table 12: Profile IDs

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

Currently there are no additional user application specific parameters. However, should these be defined in future, they will come after the `ProfileID`s and be separated from them by a delimiting 0x00 byte.

NOTE: The list of `ProfileID`s 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.

7.1.2 FIG 0/9 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.1.1 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.1.2 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.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
	creationTime	O
serviceInformation.ensemble.service.radiodns		O
	fqdn	R1
	serviceIdentifier	R1
serviceInformation.ensemble.service.alias		O
	xml:lang	R2
	prefer	R2
serviceInformation.ensemble.service.phoneme		O
	xml:lang	R2
	prefer	R2
	alphabet	R2

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
	creationTime	O
serviceInformation.service.radiodns		O
	fqdn	R1
	serviceIdentifier	R1
serviceInformation.ensemble.service.alias		O
	xml:lang	R2
	prefer	R2
serviceInformation.ensemble.service.phoneme		O
	xml:lang	R2
	prefer	R2
	alphabet	R2

A.1.3 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

Element	Attribute	Required?
epg.schedule.programme.genre		O
	href	R1
	type	R2
epg.schedule.programme.memberof		O
	shortId	R1
	index	O
epg.schedule.programme.alias		O
	xml:lang	R2
	prefer	R2
epg.schedule.programme.phoneme		O
	xml:lang	R2
	prefer	R2
	alphabet	R2

A.1.4 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.

Annex B (informative): Transformation examples

B.1 Transformation example 1

A DAB ensemble contains 11 services. The SPI service includes logos and a 7-day programme guide for all services. All the master XML documents are filtered to generate only basic profile objects for the broadcast SPI service.

The number of SPI objects generated is:

- 1 × Service Information object, basic profile;
- 77 × Programme Information objects, basic profile (7 days × 11 services).

In addition, the MOT carousel contains the Directory object and 44 logo objects (4 sizes × 11 services), a total of 123 objects.

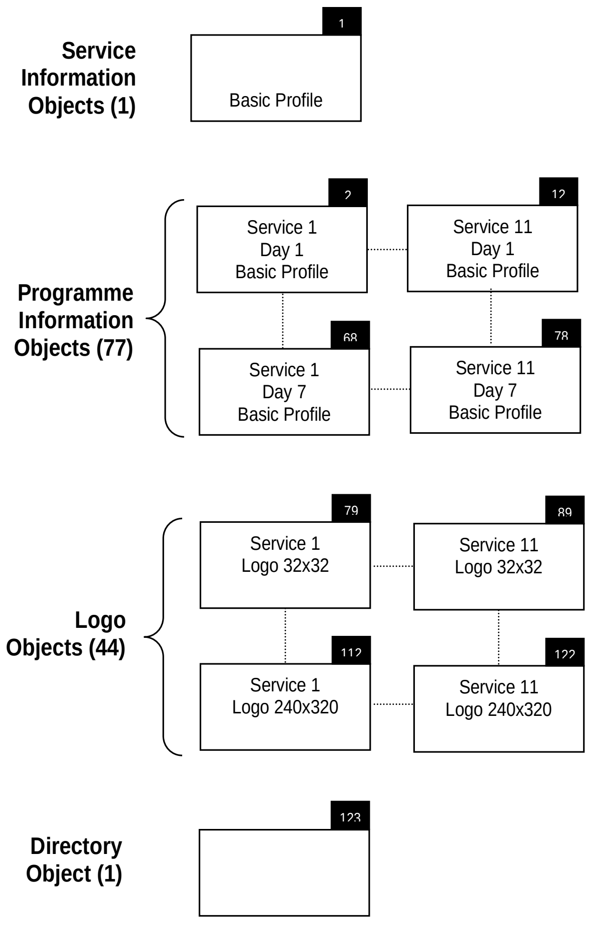

Service Information Objects (1)
1
Basic Profile
Programme Information Objects (77)
2
Service 1 Day 1 Basic Profile
12
Service 11 Day 1 Basic Profile
68
Service 1 Day 7 Basic Profile
78
Service 11 Day 7 Basic Profile
Logo Objects (44)
79
Service 1 Logo 32x32
89
Service 11 Logo 32x32
112
Service 1 Logo 240x320
122
Service 11 Logo 240x320
Directory Object (1)
123

Figure B.1: Overview of objects in the MOT carousel

B.2 Transformation example 2

There are three DAB ensembles, "A", "B" and "C", each with eight services but only ensemble "A" provides an SPI service. However, by cooperation, the logos for all services in all three ensembles are provided.

The number of SPI objects generated is:

- 3 × Service Information objects (basic profile) – one for each ensemble;

In addition, the MOT carousel contains the Directory object and 96 logo objects (4 sizes × 8 services × 3 ensembles), a total of 100 objects.

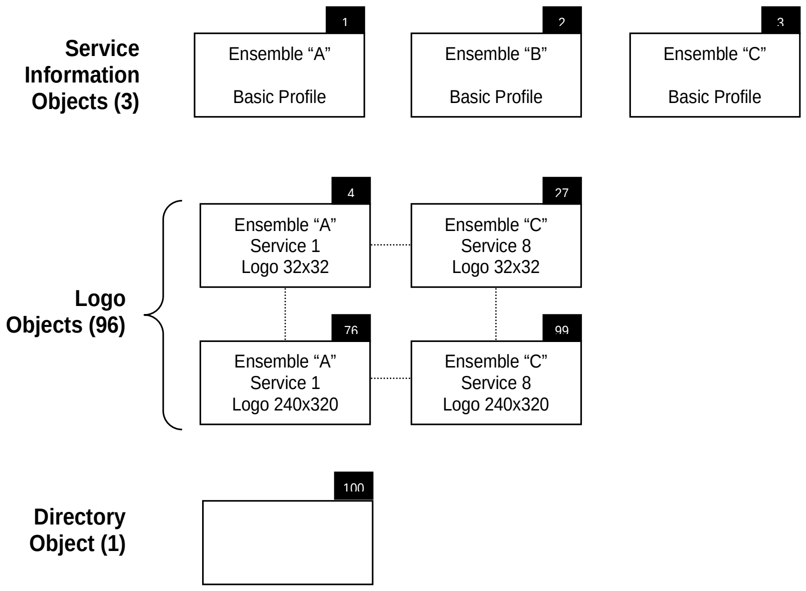

Service Information Objects (3)
1
Ensemble "A" Basic Profile
2
Ensemble "B" Basic Profile
3
Ensemble "C" Basic Profile
Logo Objects (96)
4
Ensemble "A" Service 1 Logo 32x32
27
Ensemble "C" Service 8 Logo 32x32
76
Ensemble "A" Service 1 Logo 240x320
99
Ensemble "C" Service 8 Logo 240x320
Directory Object (1)
100

Figure B.2: Overview of objects in the MOT carousel

Annex C (informative): Binary encoding examples

C.1 Service information

This example is taken from ETSI TS 102 818 [1], annex D.

NOTE: Typically, all the services of the ensemble would be present in the SI document.

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceInformation xmlns="http://www.worlddab.org/schemas/spi"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.worlddab.org/schemas/spi
http://www.worlddab.org/schemas/spi/spi_35.xsd"
   creationTime="2023-04-25T00:05:31+01:00" originator="Global Radio"
   xml:lang="en">
   <services>
      <service>
         <shortName>Capital</shortName>
         <mediumName>Capital FM</mediumName>
         <mediaDescription>
            <multimedia url="http://owdo.thisisglobal.com/2.0/id/25/logo/32x32.png"
                        type="logo_colour_square" />
         </mediaDescription>
         <mediaDescription>
            <multimedia url="http://owdo.thisisglobal.com/2.0/id/25/logo/112x32.png"
                        type="logo_colour_rectangle" />
         </mediaDescription>
         <mediaDescription>
            <multimedia url="http://owdo.thisisglobal.com/2.0/id/25/logo/128x128.png"
                        type="logo_unrestricted" mimeValue="image/png" height="128" width="128" />
         </mediaDescription>
         <mediaDescription>
            <multimedia url="http://owdo.thisisglobal.com/2.0/id/25/logo/320x240.png"
                        type="logo_unrestricted" mimeValue="image/png" height="240" width="320" />
         </mediaDescription>
         <mediaDescription>
            <multimedia url="http://owdo.thisisglobal.com/2.0/id/25/logo/600x600.jpg"
                        type="logo_unrestricted" mimeValue="image/jpeg" height="600" width="600" />
         </mediaDescription>
         <genre href="urn:tva:metadata:cs:ContentCS:2004:3.6.10" />
         <bearer id="dab:ce1.c185.c479.0" mimeValue="audio/mpeg" offset="2000" cost="20" />
      </service>
   </services>
</serviceInformation>
```

These are the bytes for the encoded binary basic profile object.

Table C.1: Binary encoded SI example

Bytes	Description
03	**<serviceInformation>** (see note 1)
9E	length = 158 bytes
26	**<ensemble>** (see note 2)
9C	length = 156
80	**id** attribute
03	length = 3
E1 C1 85	e1.c185
10	**<shortName>**
0A	length = 10
01	CDATA
08	length = 8
4C 6F 6E 64 6F 6E 20 31	London 1
11	**<mediumName>**
0A	length = 10
01	CDATA
08	length = 8
4C 6F 6E 64 6F 6E 20 31	London 1

Bytes	Description
28	**<service>**
7D	length = 125
10	**<shortName>**
09	length = 9
01	CDATA
07	length = 7
43 61 70 69 74 61 6C	Capital
11	**<mediumName>**
0C	length = 12
01	CDATA
0A	length = 10
43 61 70 69 74 61 6C 20 46 4D	Capital FM
13	**<mediaDescription>**
0B	length = 11
2B	**<multimedia>**
09	length = 9
82	**url** attribute
04	length = 4
34 37 39 53	479S (see note 3)
83	**type** attribute
01	length = 1
06	logo_colour_square
13	**<mediaDescription>**
0B	length = 11
2B	**<multimedia>**
09	length = 9
82	**url** attribute
04	length = 4
34 37 39 52	479R (see note 3)
83	**type** attribute
01	length = 1
04	logo_colour_rectangle
13	**<mediaDescription>**
1E	length = 30
2B	**<multimedia>**
1C	length = 28
82	**url** attribute
04	length = 4
34 37 39 53	479A (see note 3)
83	**type** attribute
01	length = 1
02	logo_unrestricted
80	**mimeValue** attribute
09	length = 9
69 6D 6A 67 65 2F 70 6E 67	image/png
85	**height** attribute
02	length = 2
00 80	128
84	**width** attribute
02	length = 2
00 80	128
13	**<mediaDescription>**
1E	length = 30
2B	**<multimedia>**
1C	length = 28
82	**url** attribute
04	length = 4
34 37 39 4C	479L (see note 3)
83	**type** attribute
01	length = 1
02	logo_unrestricted
80	**mimeValue** attribute
09	length = 9
69 6D 6A 67 65 2F 70 6E 67	image/png
85	**height** attribute
02	length = 2
00 F0	240

Bytes	Description
84	**width** attribute
02	length = 2
01 40	320
	<mediaDescription> (see note 4)
	<genre> (see note 5)
29	**<bearer>** (see note 6)
08	length = 8
80	**id** attribute
06	length = 6
40 E1 C1 85 C4 79	dab:ce1.c185.c479.0
NOTE 1: The **serviceInformation** attributes are not binary encoded. NOTE 2: The **services** element is not binary encoded but the **ensemble** tag is inserted using values stored in the encoder configuration. NOTE 3: The long **url** (more than 50 characters) is replaced by a short unique string that is also used for the logo contentName in the MOT carousel. By this change, more than 200 bytes are saved in the binary encoded SI object and more than 200 bytes are saved in the directory object. NOTE 4: The 600x600 logo is not coded as it is not one of the broadcast sizes. NOTE 5: The **genre** element is not coded as it is not part of the basic profile. NOTE 6: Only the **id** attribute is binary encoded.	

C.2 Programme information

```
<?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 basic profile object.

Table C.2: Binary encoded PI example

Bytes	Description
02	**<epg>**
35	length = 53 bytes
21	**<schedule>**
33	length = 51 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	dab:ce1.ce15.c224.0
1C	**<programme>**
19	length = 25
81	**shortId** attribute
03	length = 3
FA E4 51	16442449
	id attribute is not encoded in the basic profile object
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	PT1H = 3 600 seconds

Annex D (normative): Element tags

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	See also clause
epg	Top-level	0x02	5.3.2
serviceInformation	Top-level	0x03	5.3.2
tokenTable	epg, serviceInformation	0x04	5.5
-	Not used	0x05	
defaultLanguage	epg, serviceInformation	0x06	5.6
shortName	programmeGroup, ensemble, service, programme, programmeEvent	0x10	
mediumName	programmeGroup, ensemble, service, programme, programmeEvent	0x11	
longName	programmeGroup, ensemble, service, programme, programmeEvent	0x12	
mediaDescription	programmeGroup, ensemble, service, programme, programmeEvent	0x13	
genre	programmeGroup, service, programme, programmeEvent	0x14	5.3.8
-	Not used	0x15	
keywords	programmeGroup, ensemble, service, programme, programmeEvent	0x16	
memberOf	programmeGroup, programme, programmeEvent	0x17	
link	programmeGroup, ensemble, service, programme, programmeEvent	0x18	
location	programme, programmeEvent	0x19	5.3.5
shortDescription	mediaDescription	0x1A	
longDescription	mediaDescription	0x1B	
programme	schedule	0x1C	
programmeGroups	epg	0x20	
schedule	epg	0x21	
-	Not used	0x22	
programmeGroup	programmeGroups	0x23	
scope	schedule	0x24	
serviceScope	scope	0x25	5.3.3
ensemble	serviceInformation	0x26	5.3.2.3
-	Not used	0x27	
service	ensemble, serviceInformation	0x28	
bearer (see note 2)	service	0x29	5.3.4
presentationLanguage	service, schedule, programme, programmeEvent	0x2A	
multimedia	mediaDescription	0x2B	
time	location	0x2C	
bearer	location, onDemand	0x2D	5.3.4
programmeEvent	programme	0x2E	
relativeTime	location	0x2F	
-	Not used	0x30	
radiodns	service	0x31	
geolocation	service, bearer	0x32	
country	geolocation	0x33	
point	geolocation	0x34	5.3.7
polygon	geolocation	0x35	5.3.7
onDemand	programme, programmeEvent	0x36	5.3.6
presentationTime	onDemand	0x37	
acquisitionTime	onDemand	0x38	
alias	service, programme, programmeEvent	0x39	
phoneme	service, programme, programmeEvent	0x3A	
NOTE 1: Deprecated elements are marked "Not used" for reasons of backwards compatibility. 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 5.4. Additional encoding rules clauses are referenced below.

Table E.1: Common attribute tags

Element	Attribute	Tag	Type	See also clause
genre	href	0x80	Specific	5.4.5.4
	type	0x81	Enumerated	5.4.3
keywords	xml:lang	0x80	String	5.4.2
link	uri	0x80	String	5.4.2
	mimeValue	0x81	String	5.4.2
	language	0x82	String	5.4.2
	description	0x83	String	5.4.2
	expiryTime	0x84	Specific	5.4.5.2
	xml:lang	0x85	String	5.4.2
shortName	xml:lang	0x80	String	5.4.2
mediumName	xml:lang	0x80	String	5.4.2
longName	xml:lang	0x80	String	5.4.2
shortDescription	xml:lang	0x80	String	5.4.2
longDescription	xml:lang	0x80	String	5.4.2
multimedia	mimeValue	0x80	String	5.4.2
	language	0x81	String	5.4.2
	url	0x82	String	5.4.2
	type	0x83	Enumerated	5.4.3
	width	0x84	Integer	5.4.4
	height	0x85	Integer	5.4.4
	creationTime	0x86	Specific	5.4.5.2
bearer	id	0x80	Specific	5.4.5.1
	Not used	0x81		N/A
	url (see note 2)	0x82	String	5.4.5.1
geolocation	xml:id	0x80	String	5.4.2
	ref	0x81	String	5.4.2
alias	xml:lang	0x80	String	5.4.2
	prefer	0x81	Enumerated	5.4.3
phoneme	xml:lang	0x80	String	5.4.2
	prefer	0x81	Enumerated	5.4.3
	alphabet	0x82	String	5.4.5.1
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 5.4.5.1.				

Table E.2: SI attribute tags

Element	Attribute	Tag	Type	See also clause
serviceInformation (see note 1)	version	0x80	Integer	5.4.4
	creationTime	0x81	Specific	5.4.5.2
	originator	0x82	String	5.4.2
	serviceProvider	0x83	String	5.4.2
	Not used	0x84		N/A
	alphabet	0x85	String	5.4.2
ensemble	id	0x80	Specific	5.3.2.3
	Not used	0x81		N/A
service	version	0x80	Integer	5.4.4
	Not used	0x81		N/A
	Not used	0x82		N/A
	Not used	0x83		N/A
	Not used	0x84		N/A
radiodns	fqdn	0x80	String	5.4.2
	serviceIdentifier	0x81	String	5.4.2
NOTE 1: This scheme does not encode any of the schema information (e.g. the "xmlns" attribute) If xml:lang is provided it is coded using the defaultLanguage element. NOTE 2: Deprecated attributes are marked "Not used" for reasons of backwards compatibility.				

Table E.3: PI and GI attribute tags

Element	Attribute	Tag	Type	See also clause
epg (see note 1)	Not used	0x80		N/A
programmeGroups	version	0x80	Integer	5.4.4
	creationTime	0x81	Specific	5.4.5.2
	originator	0x82	String	5.4.2
programmeGroup	id	0x80	String	5.4.2
	shortId	0x81	Integer	5.4.4
	version	0x82	Integer	5.4.4
	type	0x83	Enumerated	5.4.3
	numOfItems	0x84	Integer	5.4.4
schedule	version	0x80	Integer	5.4.4
	creationTime	0x81	Specific	5.4.5.2
	originator	0x82	String	5.4.2
	aphabet	0x83	String	5.4.2
scope	startTime	0x80	Specific	5.4.5.2
	stopTime	0x81	Specific	5.4.5.2
serviceScope	id	0x80	Specific	5.4.5.1
programme	id	0x80	String	5.4.2
	shortId	0x81	Integer	5.4.4
	version	0x82	Integer	5.4.4
	recommendation	0x83	Enumerated	5.4.3
	broadcast	0x84	Enumerated	5.4.3
	Not used	0x85		N/A
	xml:lang	0x86	String	5.4.2
	Not used	0x87		N/A
programmeEvent	id	0x80	String	5.4.2
	shortId	0x81	Integer	5.4.4
	version	0x82	Integer	5.4.4
	recommendation	0x83	Enumerated	5.4.3
	broadcast	0x84	Enumerated	5.4.3
	Not used	0x85		N/A
	xml:lang	0x86	String	5.4.2
	Not used	0x87		N/A
time	time	0x80	Specific	5.4.5.2
	duration	0x81	Specific	5.4.5.3
	actualTime	0x82	Specific	5.4.5.2
	actualDuration	0x83	Specific	5.4.5.3
relativeTime	time	0x80	Specific	5.4.5.3
	duration	0x81	Specific	5.4.5.3
	actualTime	0x82	Specific	5.4.5.3
	actualDuration	0x83	Specific	5.4.5.3
memberOf	id	0x80	String	5.4.2
	shortId	0x81	Integer	5.4.4
	index	0x82	Integer	5.4.4
presentationTime	start	0x80	Specific	5.4.5.2
	end	0x81	Specific	5.4.5.2
	duration	0x82	Specific	5.4.5.3
acquistionTime	start	0x80	Specific	5.4.5.2
	end	0x81	Specific	5.4.5.2
NOTE 1: This scheme does not encode any of the schema information (e.g. the "xmlns" attribute). If xml:lang is provided it is coded using the defaultLanguage element. 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 show programConcept magazine programCompilation otherCollection otherChoice topic	0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09
programme, programmeEvent	broadcast	*on-air* off-air	0x01 0x02
programme, programmeEvent	recommendation	*no* yes	0x01 0x02
multimedia	type	logo_unrestricted Not used logo_colour_square Not used logo_colour_rectangle	0x02 0x03 0x04 0x05 0x06
genre	type	*main* secondary other	0x01 0x02 0x03
alias, phoneme	prefer	*false* true	0x01 0x02
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
V3.3.1	December 2023	Publication