INTERNATIONAL STANDARD

ISO 21219-5

First edition
2019-07

Intelligent transport systems — Traffic and travel information (TTI) via transport protocol experts group, generation 2 (TPEG2) —

Part 5: Service framework (TPEG2-SFW)

Systèmes intelligents de transport — Informations sur le trafic et le tourisme via le groupe expert du protocole de transport, génération 2 (TPEG2) —

Partie 5: Cadre de service (TPEG2-SFW)

Reference number
ISO 21219-5:2019(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 204, *Intelligent transport systems*.

This first edition cancels and replaces ISO/TS 21219-5:2015, which has been technically revised.

A list of all parts in the ISO 21219 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

History

TPEG technology was originally proposed by the European Broadcasting Union (EBU) Broadcast Management Committee, who established the B/TPEG project group in the autumn of 1997 with a brief to develop, as soon as possible, a new protocol for broadcasting traffic and travel-related information in the multimedia environment. TPEG technology, its applications and service features were designed to enable travel-related messages to be coded, decoded, filtered and understood by humans (visually and/or audibly in the user's language) and by agent systems. Originally, a byte-oriented data stream format, which may be carried on almost any digital bearer with an appropriate adaptation layer, was developed. Hierarchically structured TPEG messages from service providers to end-users were designed to transfer information from the service provider database to an end-user's equipment.

One year later, in December 1998, the B/TPEG group produced its first EBU specifications. Two documents were released. Part 2 (TPEG-SSF, which became ISO/TS 18234-2) described the syntax, semantics and framing structure, which was used for all TPEG applications. Meanwhile, Part 4 (TPEG-RTM, which became ISO/TS 18234-4) described the first application for road traffic messages.

Subsequently, in March 1999, CEN/TC 278, in conjunction with ISO/TC 204, established a group comprising members of the former EBU B/TPEG and this working group continued development work. Further parts were developed to make the initial set of four parts, enabling the implementation of a consistent service. Part 3 (TPEG-SNI, ISO/TS 18234-3) described the service and network information application used by all service implementations to ensure appropriate referencing from one service source to another.

Part 1 (TPEG-INV, ISO/TS 18234-1) completed the series by describing the other parts and their relationship; it also contained the application IDs used within the other parts. Additionally, Part 5, the public transport information application (TPEG-PTI, ISO/TS 18234-5), was developed. The so-called TPEG-LOC location referencing method, which enabled both map-based TPEG-decoders and non-map-based ones to deliver either map-based location referencing or human readable text information, was issued as ISO/TS 18234-6 to be used in association with the other applications of parts of the ISO/TS 18234 series to provide location referencing.

The ISO/TS 18234 series has become known as TPEG Generation 1.

TPEG Generation 2

When the Traveller Information Services Association (TISA), derived from former forums, was inaugurated in December 2007, TPEG development was taken over by TISA and continued in the TPEG applications working group.

It was about this time that the (then) new Unified Modelling Language (UML) was seen as having major advantages for the development of new TPEG applications in communities who would not necessarily have binary physical format skills required to extend the original TPEG TS work. It was also realized that the XML format for TPEG described within the ISO/TS 24530 series (now superseded) had a greater significance than previously foreseen, especially in the content-generation segment and that keeping two physical formats in synchronism, in different standards series, would be rather difficult.

As a result, TISA set about the development of a new TPEG structure that would be UML-based. This has subsequently become known as TPEG Generation 2.

TPEG2 is embodied in the ISO/TS 21219 series and it comprises many parts that cover introduction, rules, toolkit and application components. TPEG2 is built around UML modelling and has a core of rules that contain the modelling strategy covered in ISO 21219-2, ISO 21219-3 and ISO 21219-4 and the conversion to two current physical formats: binary and XML; others could be added in the future. TISA uses an automated tool to convert from the agreed UML model XMI file directly into an MS Word document file, to minimize drafting errors, that forms the annex for each physical format.

TPEG2 has a three-container conceptual structure: message management (ISO 21219-6), application (several parts) and location referencing (ISO/TS 21219-7). This structure has flexible capability and can accommodate many differing use cases that have been proposed within the TTI sector and wider for hierarchical message content.

TPEG2 also has many location referencing options as required by the service provider community, any of which may be delivered by vectoring data included in the location referencing container.

The following classification provides a helpful grouping of the different TPEG2 parts according to their intended purpose. Note that the list below may be incomplete, e.g. new TPEG2 parts may be introduced after publication of this document.

— Toolkit parts: TPEG2-INV (ISO/TS 21219-1), TPEG2-UML (ISO 21219-2), TPEG2-UBCR (ISO 21219-3), TPEG2-UXCR (ISO 21219-4), TPEG2-SFW (ISO 21219-5), TPEG2-MMC (ISO 21219-6), TPEG2-LRC (ISO/TS 21219-7).

— Special applications: TPEG2-SNI (ISO/TS 21219-9), TPEG2-CAI (ISO/TS 21219-10), TPEG2-LTE (ISO/TS 21219-24).

— Location referencing: TPEG2-OLR (ISO/TS 21219-22), TPEG2-GLR (ISO/TS 21219-21), TPEG2-TLR (ISO 17572-2), TPEG2-DLR (ISO 17572-3).

— Applications: TPEG2-PKI (ISO/TS 21219-14), TPEG2-TEC (ISO/TS 21219-15), TPEG2-FPI (ISO/TS 21219-16), TPEG2-TFP (ISO 21219-18), TPEG2-WEA (ISO/TS 21219-19), TPEG2-RMR (ISO/TS 21219-23), TPEG2-EMI (ISO/TS 21219-25), TPEG2-VLI (ISO/TS 21219-26).

TPEG2 has been developed to be broadly (but not totally) backward compatible with TPEG1 to assist in transitions from earlier implementations, while not hindering the TPEG2 innovative approach and being able to support many new features, such as dealing with applications having both long-term, unchanging content and highly dynamic content, such as parking information.

Intelligent transport systems — Traffic and travel information (TTI) via transport protocol experts group, generation 2 (TPEG2) —

Part 5: Service framework (TPEG2-SFW)

1 Scope

This document establishes a method of conveying data for a wide range of applications that require the efficient transmission of point to multi-point data over potentially unreliable broadcast channels. It is also suitable for point-to-point and multicast applications and may easily be encapsulated in Internet Protocol.

This document describes the basic capabilities of the generation 2 TPEG (TPEG2) for providing a multiplex of TPEG Services and applications. Together with the definitions of the general TPEG UML modelling rules and the particular physical TPEG representations for TPEG-binary streams (TISA: TPEG UML Conversion Rules) and tpegML files (TISA Specification: TPEG UML Conversion Rules), it replaces the former documents TPEG-INV and TPEG-SSF.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/TS 18234-3, *Intelligent transport systems — Traffic and travel information via transport protocol experts group, generation 1 (TPEG1) binary data format — Part 3: Service and network information (TPEG1-SNI)*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
TPEG Application
application layer protocol fulfilling the general TPEG requirements at the highest layer of the ISO OSI model and standardized by TISA/ISO

Note 1 to entry: A TPEG Application consists of a set of classes and rules for encoding information required for a traffic information service.

3.2
TPEG Client
device or entity on the receiving side of the TPEG transmission chain

Note 1 to entry: See 5.2.

3.3
TPEG Server
device or entity on the sending side of the TPEG transmission chain

Note 1 to entry: See 5.2.

3.4
TPEG Service
multiplex of TPEG Service Components with a dedicated Service ID

Note 1 to entry: See 6.1.

3.5
TPEG Service Component
virtual channel for messages of a dedicated TPEG Application

Note 1 to entry: See 6.1.

3.6
Service Frame
data-structure implementing the TPEG Service in the TPEG binary representation

3.7
Service Component Frame
data-structure implementing the TPEG Service Component stream in the TPEG binary representation

3.8
TPEG Service Multiplex
multiplex of TPEG Services within one data stream or file

3.9
TPEG Stream Directory
TPEG Structure used for signalling the TPEG Services within a Service Multiplex

3.10
TPEG Structure
data structure used by TPEG on the particular protocol layers of the service transmission

4 Abbreviated terms

AID	Application Identification
BPN	Broadcast, Production and Networks (an EBU document publishing number system)
CEN	Comité Européen de Normalisation
CRC	Cyclic Redundancy Check
DAB	Digital Audio Broadcasting
DVB	Digital Video Broadcasting
EBU	European Broadcasting Union
INV	Introduction, Numbering and Versions (see EBU BPN 027-1)
IPR	Intellectual Property Right(s)
ISO	International Organization for Standardization

ITU-T	International Telecommunication Union — Telecom
OSI	Open Systems Interconnection
PTI	Public Transport Information
RTM	Road Traffic Message Application
SFW	Service Framework (this Technical Specification)
SID	Service Identification
SNI	Service and Network Information Application (see EBU BPN 027-3)
SSF	Syntax, Semantics and Framing Structure
TPEG	Transport Protocol Experts Group
TTI	Traffic and Travel Information
UTC	Universal Coordinated Time
UML	Unified Modelling Language
XML	Extensible Markup Language
XSD	XML Schema Definition

5 General — TPEG

5.1 TPEG transmission

TPEG is intended to operate via almost any simple digital data channel, where it is primarily targeted at broadcast media using byte oriented transparent data channels. Other physical formats may pose different constraints on a transmission layer. Thus, TPEG assumes nothing of the channel other than the ability to convey a stream of bytes. To this end, the concept of transmission via a "piece of wire" is envisaged, in which the bearer has no additional service features.

In Figure 1, a variety of possible transmission channels are shown. The only requirement of the channel is that a sequence of bytes may be carried between the TPEG generator and the TPEG decoder. This requirement is described as "transparency". However, it is recognized that data channels may introduce errors. Bytes may be omitted from a sequence, bytes may become corrupted or additional and erroneous data could be received. Therefore, TPEG incorporates error detection features at appropriate points and levels. It is assumed that bearer systems will introduce an appropriate level of error correction.

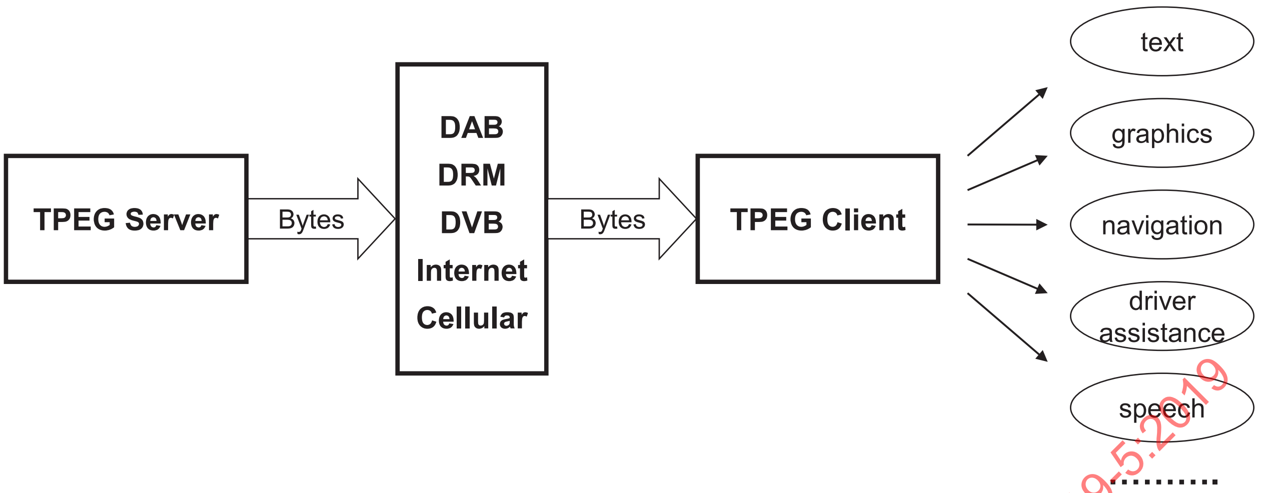

Figure 1 — TPEG data may be delivered simultaneously via different bearer channels

5.2 TPEG roles

The following roles are defined for TPEG devices:

— **TPEG Server** — is the device, group of devices or entity that provides the capabilities to encode TPEG objects, e.g. TPEG messages, TPEG Service Frames or TPEG Service Component Frames and which transmits it via a suitable digital bearer to the TPEG Client side.

— **TPEG Client** — is the device or entity that provides the capabilities to decode TPEG objects received from one or several TPEG Servers.

These terms are used in the rest of this document to designate these roles.

5.3 TPEG layer model

In Figure 2, the different layers of the TPEG protocol are identified in accordance with the ISO/OSI model.

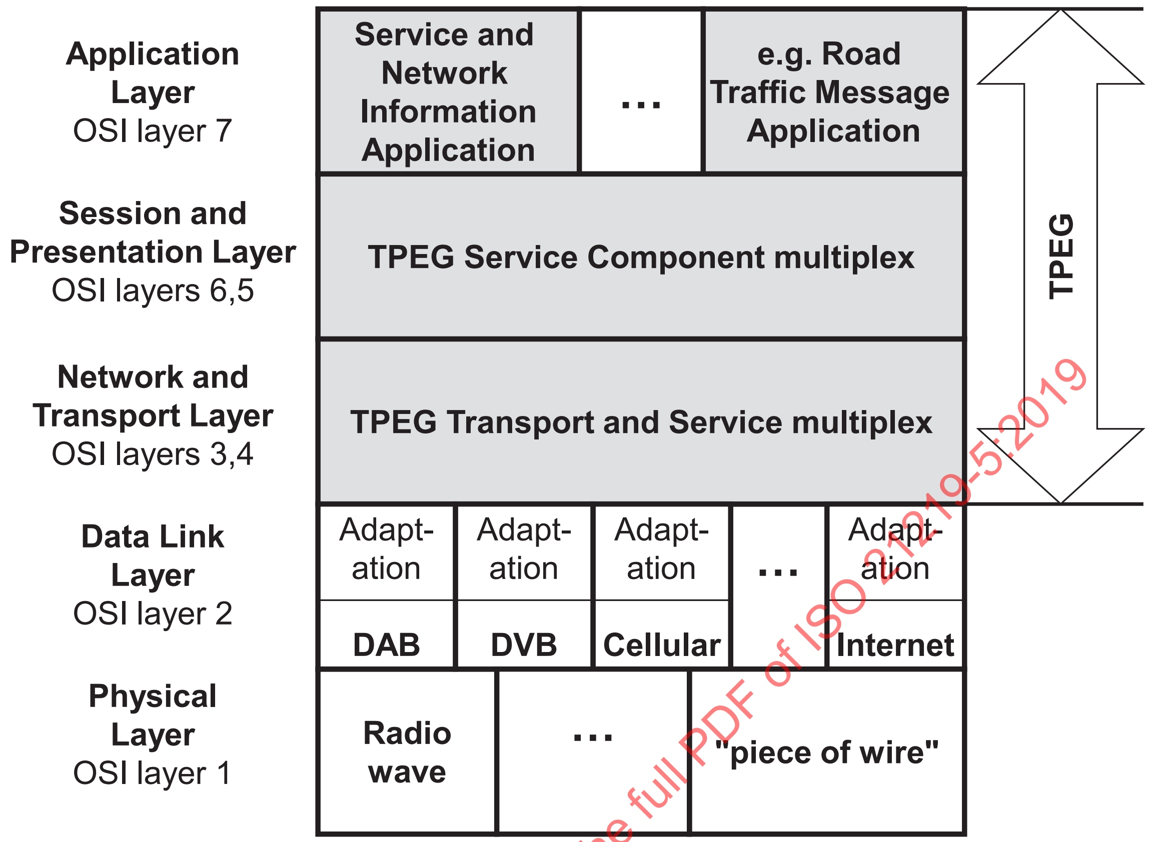

Figure 2 — TPEG in relation to the ISO/OSI Layer Model

Layer 7 is the top level and referred to in TPEG as the application layer. The following TPEG Applications are defined at the date of publication of this document:

— Service and Network Information (SNI) Application;

— Road Traffic Message (RTM) Application;

— Public Transport Information (PTI) Application;

— Location Referencing Container (LRC);

— Parking Information (PKI) Application;

— Traffic Event Compact (TEC) Application;

— Conditional Access Information (CAI) Application.

An up-to-date list of TPEG Applications can be found on the TISA webpage.

Layers 6 and 5 are the presentation and session layers. TPEG Service Components are merged into a single stream and encrypted and/or compressed.

Layers 3 and 4 are the transport and network layers. These layers define the means for synchronisation and routing. This is the lowest layer of the TPEG protocol.

Layer 2 is the datalink layer. This layer consists of a wide range of different bearers, which are suitable carriers for the TPEG protocol. An adaptation layer may be required in order to map the TPEG stream onto that bearer for that TPEG may also define requirements to the bearer.

Layer 1 is the physical layer. This defines the transmission medium (radio waves, wire, optical, etc.). One particular bearer can make use of different physical layers.

5.4 Design principles

The following general principles have been assumed in the development of the TPEG protocol, structure and semantics:

— TPEG is platform independent and bearer independent;

— TPEG is designed for but not restricted to unidirectional transmission;

— TPEG provides a protocol structure, which employs asynchronous framing;

— TPEG assumes that underlying systems may employ error correction;

— TPEG has a hierarchical data frame structure enabling a multiplex of TPEG Services and TPEG Applications within one data stream;

— TPEG provides worldwide unique identification of TPEG Services;

— TPEG permits the use of encryption mechanisms, if required by an application;

— TPEG Applications can be extended in a backwards compatible way;

— TPEG Applications are modelled using UML. Specific physical transmission formats are derived from these models.

6 Description of TPEG Multiplex and TPEG Structures

6.1 Overview

TPEG provides multiplexing functionality on several TPEG levels (see also Figure 3 below):

— **TPEG Service Components and TPEG Applications:** A TPEG Service Component is used to provide a virtual channel for streams of messages of one and only one TPEG Application type. Accordingly, the content of a TPEG Service Component is encoded following the definitions of the corresponding TPEG Application.

— **TPEG Service Component Multiplex and TPEG Services:** A TPEG Service consists of one or several TPEG Service Components, thus combining several application specific message streams. Each TPEG Service has a worldwide unique Service Identification (SID).

— **TPEG Service Multiplex:** One or several TPEG Service streams can be multiplexed to one bearer-related data-stream. Each of the TPEG Service data objects within this data stream can be unambiguously assigned to a TPEG Service by its unique SID. The Service multiplex enables the realization of several TPEG Services within one data stream.

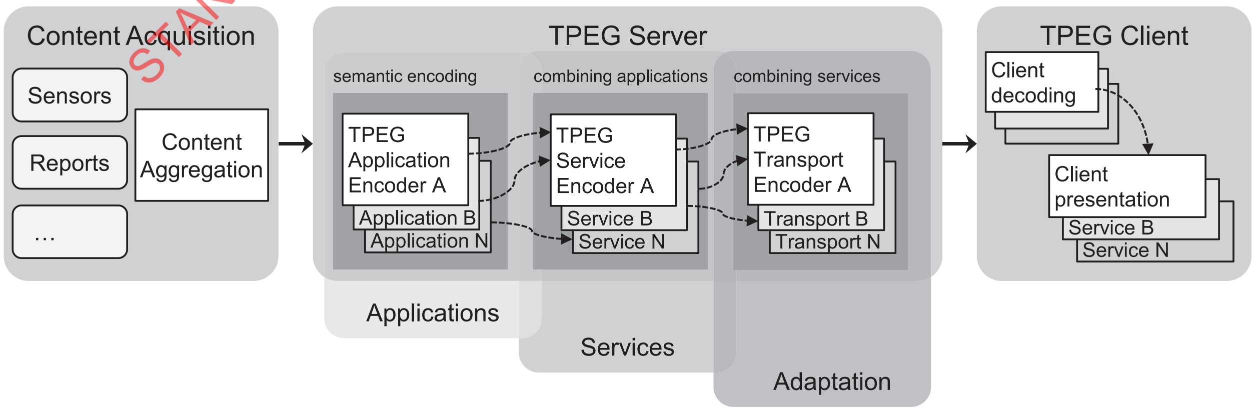

Figure 3 — TPEG multiplex hierarchy

The multiplex hierarchy described above requires corresponding data structures (TPEG Structures) for the particular protocol layers, i.e.:

— a TPEG Service Component structure for the messages multiplex on Service Component or TPEG Application layer;

— a TPEG Service structure for the Service Component Multiplex on the TPEG Service layer;

— TPEG Transport structures for the multiplex of several TPEG Services within a physical data stream, in particular for the bearer abstraction and the signalling of TPEG Services.

An overview of the structures is given in Figure 4. The hierarchy includes data structures for TPEG Service Components, TPEG Services and TPEG Transport, where the latter ensures the abstraction from the particular transmission bearer. A further structure is the TPEG Stream Directory, which may be used to signal the TPEG Services transmitted in a TPEG Service Multiplex.

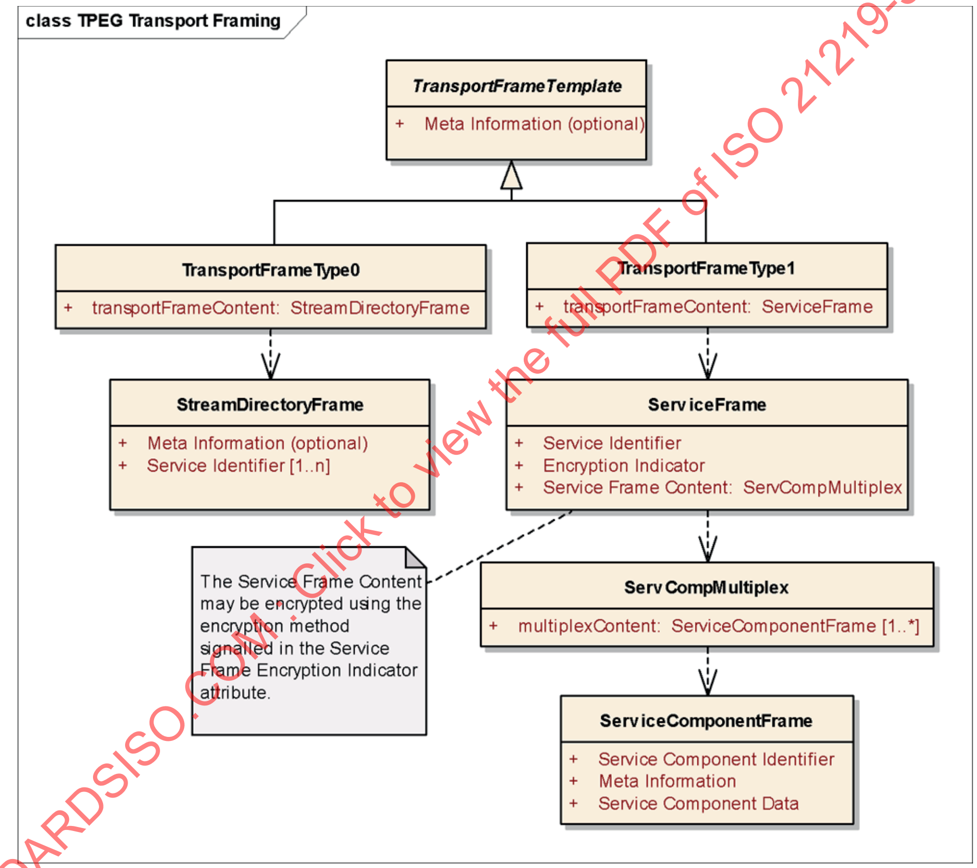
class TPEG Transport Framing
TransportFrameTemplate
+ Meta Information (optional)
TransportFrameType0
+ transportFrameContent: StreamDirectoryFrame
TransportFrameType1
+ transportFrameContent: ServiceFrame
StreamDirectoryFrame
+ Meta Information (optional)
+ Service Identifier [1..n]
ServiceFrame
+ Service Identifier
+ Encryption Indicator
+ Service Frame Content: ServCompMultiplex
The Service Frame Content may be encrypted using the encryption method signalled in the Service Frame Encryption Indicator attribute.
ServCompMultiplex
+ multiplexContent: ServiceComponentFrame [1..*]
ServiceComponentFrame
+ Service Component Identifier
+ Meta Information
+ Service Component Data

Figure 4 — Transport hierarchy of TPEG Stream Directory and Service Frame Structures

Further transport structures may be defined in the future, e.g. for TPEG multiplex information.

The TPEG Structures shown in the figures above are described on an abstract level in the following clauses. These UML definitions shall not be used for automatic generation of physical format descriptions as defined in TISA Specification: TPEG UML Modelling Rules, TISA: TPEG UML Conversion Rules and TISA Specification: TPEG UML Conversion Rules. In particular, the UML classes above shall not be considered as TPEG Service Components. For definitions of its physical data representations see Annex A for TPEG-binary and Annex B for tpegML.

6.2 TPEG Transport level

The TPEG Transport level is dependent on the physical representation used. The information provided on this level may include meta information for data structure identification, synchronization and error detection. For details see Annex A for TPEG-binary and Annex B for tpegML.

6.3 TPEG Service level

6.3.1 General

The following sub-clauses describe in an abstract way the data objects used by the TPEG Service Multiplex and Service Component Multiplex. The physical representations of these data objects in TPEG-binary and tpegML are defined in Annex A and Annex B. The attributes of these data objects have dedicated types, which are defined in this document or in TISA Specification: TPEG UML Modelling Rules.

6.3.2 TPEG Service structure

The TPEG Service Structure contains the Service Component Multiplex of a dedicated TPEG Service. Each instance of this structure includes the SID of the corresponding TPEG Service and a different range and number of Service Component Structures as required by the service provider.

The attributes of the TPEG Service structure are listed hereunder:

Table 1 — Attributes of the TPEG Service structure

Name	Type	Multiplicity	Description
SID	ServiceIdentifier	1	Service Identifier (SID A/B/C) of the TPEG Service corresponding to the including Service Structure instance. For details see 6.3.3.2.
ServEncID	IntUnTi	1	The Service Encryption Indicator signals the encryption method used for the Service Component Multiplex contained in the Service Structure object. For details see 6.3.3.1.
Service Component Multiplex	ServiceComponent	1..*	One or several instances of type Service Component Structure (see also 5.4). The resultant data object is transformed according to the encryption method required (if the Encryption Indicator is not 0) or is left unchanged (if the Encryption Indicator = 0).

This structure is solely used to transport the TPEG Stream Directory information, i.e. the SIDs of the TPEG Services transmitted in this data stream. The attributes of the TPEG Stream Directory Structure are listed hereunder:

Table 2 — Attributes of the TPEG Stream Directory structure

Name	Type	Multiplicity	Description
SID	ServiceIdentifier	1..*	A vector of Service Identifiers (SID A/B/C) of the TPEG Services transmitted in the corresponding data stream.

6.3.3 Service level attributes description

6.3.3.1 Service Encryption Indicator

The Service Encryption Indicator is an unsigned integer value with range 0-255. If the indicator has value 0 all data in the Service Component Multiplex are non-encrypted. Every other value of the Service Encryption Indicator indicates that one of several mechanisms for data encryption or compression has been utilized for all data in the following multiplex data. The encryption/compression technique and algorithms may be freely chosen by the service provider.

0	= no encryption/compression;
1 to 127	= reserved for standardised methods, for the current list of already allocated Encryption Indicators, see http://www.tisa.org;
128 to 255	= may be freely used by each service provider, may indicate the use of proprietary methods.

6.3.3.2 Service identification

The Service IDs are structured in a similar way to Internet IP addresses as follows:

SID-A . SID-B . SID-C

The range of each SID element is 0-255 (type IntUnTi). The combination of these three SID elements shall be uniquely allocated on a worldwide basis. The following address allocation system applies:

SID range for TPEG technical test SIDs = 000.000.000 – 000.127.255

Any SID may be used within this range for up to a maximum of 12 months, on a self allocation basis. Services with such an SID shall not be shown in any end-user client device

(allows for approximately 32,000 possibilities)

SID range for TPEG public test SIDs = 000.128.000 – 000.255.255

Any SID may be used within this range for up to a maximum of 12 months, on a self allocation basis. Services with such an SID may be shown in end-user client devices and shall be clearly marked as trials that may contain invalid data

(allows for approximately 32,000 possibilities)

SID range for TPEG regular public services SIDs = 001.000.000 – 100.255.255

SIDs within this range will be allocated by the TISA, for a small fee and may then be used on a long-term basis, subject to renewal

(allows for approximately 6 million possibilities)

SID range: reserved for future use SIDs = 101.000.000 – 255.255.255

SIDs within this range will be allocated by the TISA, at some time in the future

(allows for approximately 10 million possibilities)

A SID in the ‘TPEG technical test range’ may be used by a tester or service provider, self organized, making the assumption that NO client device will respond to such a SID. All SIDs in the ‘TPEG public test' and 'TPEG regular public services' ranges are assumed to cause a response in any client device. Specialist test client devices may be able to access any range of SIDs. For the current list of already allocated SID's, see http://www.tisa.org.

6.4 TPEG Service Component level

6.4.1 Service Component structure

The TPEG Service Component Multiplex is realized by a collection of one or more instances of TPEG Service Component Structures, the type and order of which are freely determined by the service provider.

The composition of the Service Component Multiplex of a TPEG Service is signalled by the tables in the related SNI as defined in ISO/TS 18234-3. Within a Service Component Multiplex stream one and only one SNI Service Component shall be present. This SNI Service Component shall have the Service Component ID (SCID) = 0. Note that the SNI Service Component may not be included in every TPEG Service Structure instance but shall be present in the Service Component stream. The service provider is free to determine the transmission frequency of the SNI though it should be sent in each Service Structure instance, if possible.

According to the Service Component Encryption Indicator (EncID) value in the SNI (ISO/TS 18234-3) the Service Component Data included in a Service Component may be encrypted (see also 6.3.3.1) while the other attributes of the Service Component Structure shall be unencrypted in this case.

The attributes of the TPEG Service Component Structure are listed hereunder.

Table 3 — Attributes of the TPEG Service Component structure

Name	Type	Multiplicity	Description
Meta Information	Dependent on physical representation	0..1	Meta information for data structure identification and error detection (CRC), as required by the physical representation applied
SCID	IntUnTi	1	Service Component Identifier (SCID) as defined in the related SNI component. For the SNI the value shall always be SCID = 0.
Service Component Data	ServiceComponentData	1..*	The Service Component Data contains the TPEG messages encoded according the specification of the related TPEG Application. The Service Component Data maybe encrypted if encryption on the Service Component level is applied.

Annex A
(normative)

TPEG-Binary Representation of Framework Structures

A.1 General definitions

For general definitions concerning TPEG data types refer to TISA Specification: TPEG UML Modelling Rules. The notation is defined in the UML to TPEG-Binary conversion rules specification. However, the notation is used in this document without having the UML equivalent.

A.2 TPEG data stream description

A.2.1 Diagrammatic hierarchy representation of frame structure

In the TPEG-binary representation the TPEG Structures on the Transport, Service and Service Component level are realized by data frames. The following diagram shows the hierarchical structure of the TPEG frames:

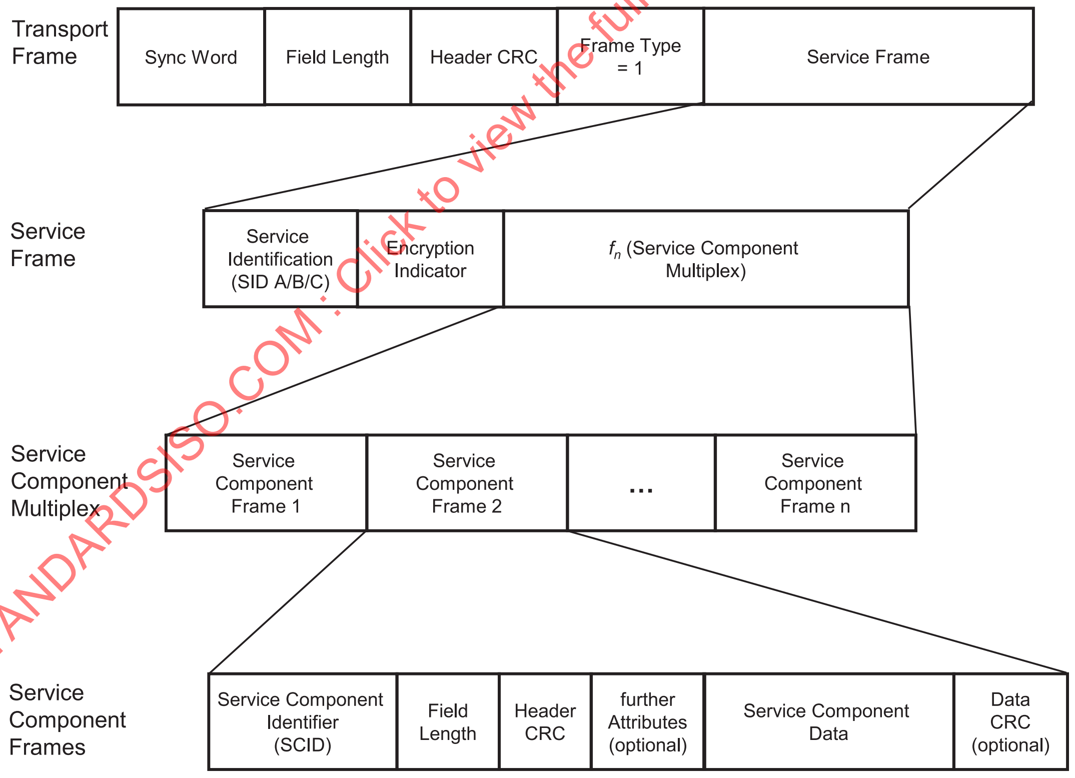
Transport Frame
Sync Word
Field Length
Header CRC
Frame Type = 1
Service Frame
Service Frame
Service Identification (SID A/B/C)
Encryption Indicator
f_n (Service Component Multiplex)
Service Component Multiplex
Service Component Frame 1
Service Component Frame 2
...
Service Component Frame n
Service Component Frames
Service Component Identifier (SCID)
Field Length
Header CRC
further Attributes (optional)
Service Component Data
Data CRC (optional)

Figure A.1 — TPEG Frame Structure, Frame Type = 1 (i.e. conventional data)

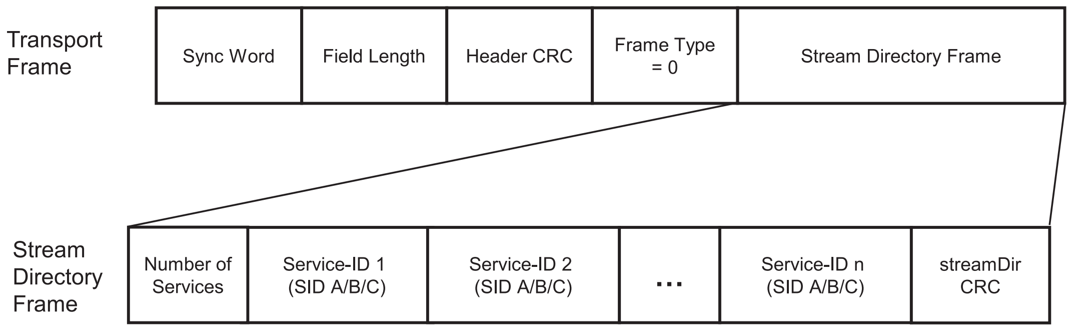

Figure A.2 — TPEG Frame Structure, Frame Type = 0 (i.e. Stream Directory)

A.2.2 Syntactical representation of the TPEG stream

A.2.2.1 TPEG Transport Frame

A.2.2.1.1 Structure

The following boxes are the syntactical representation of the TPEG frame structure shown in A.2.1. The byte stream consists of consecutive TPEG Transport Frames. The Transport Frames in a TPEG data stream may belong to one service provider only or may build a multiplex of TPEG Services of different service providers.

Each Transport Frame shall include exactly one TPEG Service Frame whose length is restricted to max. 65535 Bytes.

The byte stream shall be built according to the above-mentioned repetitive structure of Transport Frames. If possible, one Transport Frame should follow another directly, however if any spacing bytes are required these should be set to 0 hex (padding bytes).

```
< TpegStream > : =                    : The data stream
    infinite {                        : Control element, (loop continues infinitely)
        n *  < IntUnTi > (0),         : Any number of padding bytes (0 hex)
     < TransportFrame >               : Transport Frame
    };
```

```
< TransportFrame(0) > : =
     < IntUnLi > (FF0F hex),          : Synchronization word (FF0F hex), see A.2.2.1.2
     < IntUnLi > (m),                 : Field length, number of bytes in Service Frame (max. 65535 Bytes),
                                        see A.2.2.1.2
     < CRC > (tFrameHeaderCRC),       : Transport Frame Header CRC, see A.2.2.1.2
     < IntUnTi > (x),                 : Frame type of Service Frame, see A.2.2.1.2
< ServiceFrame(x) > ;                 : Any Service Frame follows
```

A.2.2.1.2 TPEG Transport Frame attributes

Syncword

The Syncword shall be 2 bytes long and shall have the value of FF0F hex.

The nibbles F hex and 0 hex have been chosen for simplicity of processing in decoders. The patterns 0000 hex and FFFF hex were deprecated to avoid the probability of false triggering in the cases of some commonly used transmission channels. The byte sequence FF0F may also occur as regular TPEG data in the transport frame payload. As a consequence, additional verification of synchronization status is required. This may for instance be performed by checking the transport frame header CRC.

Field Length

The value of the Field Length attribute shall be equal to the number of bytes in the Service Frame, i.e. all bytes following the Frame Type attribute. This derives from the need of variable length frames.

Frame Type

The Value of the Frame Type attribute indicates the content of the Service Frame The following table gives the meaning of the Frame Type:

Frame Type value (dec):	Kind of information in Service Frame:
0	Stream Directory information included in the Transport Frame (see A.2.2.3)
1	Service Component Multiplex data included in the Transport Frame (see A.2.2.4)

Transport Frame Header CRC

The Transport Frame Header CRC enables an error detection of the header of Service Component Frames:

The CRC shall be two bytes long, and shall be based on the ITU-T polynomial $x^{16} + x^{12} + x^5 + 1$.

The CRC shall be calculated on 16 bytes including the Syncword, the Field Length, the Frame Type and the first 11 bytes of the Service Frame. In the case that a Service Frame is shorter than 11 bytes, the Syncword, the Field Length, the Frame Type and the *whole* Service Frame shall be taken into account. In this case the Header CRC calculation does not run into the next Transport Frame.

The calculation of the CRC is described in Annex D.

A.2.2.1.3 Synchronization method

On the TPEG Service Level a three-step synchronization algorithm can be implemented to synchronize the TPEG Client:

a) Search for an FF0F hex value (Syncword).

b) Calculate and check the header CRC, which follows.

c) Check the two bytes which follow the end of the Service Frame as defined by the field length.

The two bytes following the end of the Service Frame should either be a sync word or 00 hex, when spaces are inserted for padding.

A.2.2.2 TPEG Service Frame template structure

A Service Frame comprises:

< ServiceFrame(x) > : =	: Template for Service Frame
n * **< byte > ;**	: Content of service frame

A.2.2.3 Service Frame of frame type = 0 (Stream Directory Frame)

A.2.2.3.1 Structure

This Service Frame is solely used to transport the Stream Directory information.

< StreamDirectory < ServiceFrame(0) > > : =	: Stream Directory frame
< IntUnTi > (n),	: Number of services
n * **< ServiceIdentifier > ,**	: Any number of Service IDs
< CRC > (streamDirCRC);	: Stream Directory CRC for the number (n) of Service IDs and the SID-list, see A.2.2.3.2

A.2.2.3.2 Attributes

Stream Directory CRC

The Stream Directory CRC enables an error detection of the Stream Directory Frame data.

For the Stream Directory Frame (Frame Type = 0) an extra CRC calculation is done over the whole Stream Directory data, i.e. starting with the attribute n encoding the number of SIDs in the Stream Directory and ending with SID-C of the last SID.

The calculation of the CRC is described in Annex D.

A.2.2.4 Service Frame of frame type = 1 (Service Data Frame)

A TPEG Service Frame may contain a different range and number of Service Component Frames as required by the service provider. Furthermore, each Service Data Frame shall include service information that comprises the service identification elements and the encryption indicator.

< ServiceData < ServiceFrame(1) > > : =	: Service data frame
< ServiceIdentifier > (SID),	: Service identification (SID-A, SID-B, SID-C)
< IntUnTi > (ServEncID),	: Service encryption indicator, 0 = no encryption
f_n**(<ServCompMultiplex >);**	: Function f_n (...) is utilized according to the chosen encryption algorithm

A.2.2.5 TPEG Service Component Multiplex

The Service Component Multiplex is a collection of one or more Service Component Frames, whose type and order are freely determined by the service provider. The resultant multiplex is transformed according to the encryption method required (if the encryption indicator of the Service Frame is not 0) or is left unchanged (if the encryption indicator = 0). The length of the resultant data shall be less than or equal to 65531 bytes.

< ServCompMultiplex > : =	
n * **< ServCompFrame >** (data);	: Any number of any Service Component Frames

A.2.2.6 Interface to application specific Service Component Frames

The TPEG Service Component Frame introduces the application specific code. This means further details of the data stream are specified by the application specification. In the history for different needs slightly different Service Component Frames have been defined in the existing application specifications. To harmonize this kind of frame, especially for new developments of specifications, this section specifies not only a basic Service Component Frame which is required for any application but also a selection of possible other Service Component Frames, whereof an application can just choose one without the need to specify its own Service Component Frame.

An application specification, however, can specify its own Service Component Frame, which shall at minimum include the following base Service Component Frame as first sub type.

A.2.2.6.1 TPEG Base Service Component Frame structure

In a TPEG data stream it is possible to transmit a variety of TPEG message types. For that, the Service and Network Information (SNI) Application provides a variable directory-like information of the TPEG Service structure. This includes also the definition which the TPEG Application can be expected in a specific Service Component Frame.

Thus, the Service Component Frame starts not with a typical interface template, but with a header, defining the three first frame attributes being in common for all Service Component Frames. Therefore, any Service Component Frame is built as shown below:

< **ServCompFrame** > : =	: Service Component Frame
< **ServCompFrameHeader** > (header),	: Common Service Component header
< **ApplicationData** > (data);	: Component data

Where the Service Component header is specified as:

< **ServCompFrameHeader** > : =	: Common Service Component Frame header
< **IntUnTi** > (SCID),	: Service Component identifier (SCID is defined by SNI Service Component, which is amongst others designating the application in this Service Component Frame)
< **IntUnLi** > (fieldLength),	: Length of the component data in bytes, starting with the first byte after the scHeaderCRC attribute
< **CRC** > (scHeaderCRC);	: Service Component Header CRC (see A.2.3.3)

At the Service Component level data are carried in Service Component Frames which have a limited length. If applications require greater capacity then the application shall be designed to distribute data between Service Component Frames and to recombine this information in the decoder.

The inclusion of the field length enables the decoder to skip a component.

The maximum field length of the Service Component data (assuming that there is no transformation, and only one component is included in the Service Frame) = 65526.

A.2.2.6.2 TPEG specialized Service Component data schemata

For reasons of consistency, Service Component Frames should be defined as similar as possible for the different applications. Specifically, three further header attributes of general nature may be used in the application specific Service Component Frames for the following purposes:

— The application data of a Service Component Frame with **dataCRC** is error-free.

  — Data CRC on this level makes it possible, that in case of errors only the Service Component Frame (e.g. one relatively small package of data) would be lost. Other parts of the service multiplex may still be valid.

— Count of messages the Service Component Frame contains named **messageCount**.

  — Sometimes it is useful not only to know the opaque count of bytes, but also how many messages are included in the Service Component Frame data.

— Prioritization can be made by assigning a **groupPriority**.

In some cases the different Service Components received shall not only be handled by a FIFO buffer but also with some qualification of severity of messages. In this case a high priority message may take precedence over other messages in the decoder. These may be presented to the user even before low priority messages are decoded.

A.2.2.6.2.1 Service Component Data with dataCRC

The basic recommended Service Component frame extends the < ServCompFrame > with a data CRC as defined in A.2.3.4. This ensures that bit errors can be detected on Service Component Frame level.

< **ServCompFrameProtected** > : =	: CRC protected Service Component Frame
< **ServCompFrameHeader** > (header),	: Service Component Frame header as defined in A.2.2.6.1
external < **ApplicationContent >** (content),	: Content specified by the individual application
< **CRC** > (dataCRC);	: CRC starting with first byte after the Service Component Header CRC

A.2.2.6.2.2 Service Component Data with dataCRC and messageCount

Applications requiring direct access to the number of messages that are available in a Service Component can use the < ServCompFrameCountedProtected > Service Component Frame type. This Service Component Frame type also includes the recommended data CRC for bit error detection at Service Component Frame level.

< **ServCompFrameCountedProtected** > : =	: CRC protected Service Component Frame with message count
< **ServCompFrameHeader** > (header),	: Service Component Frame header as defined in A.2.2.6.1
< **IntUnTi** > (messageCount),	: count of messages in this ApplicationContent
external < **ApplicationContent** > (content),	: actual payload of the application
< **CRC** > (dataCRC);	: CRC starting with first byte after the Service Component Header CRC

A.2.2.6.2.3 Service Component Data with dataCRC and groupPriority

Applications requiring priority handling at Service Component Frame level may use the < ServCompFramePrioritisedProtected > Service Component Frame type. If not all messages within the frame have the same priority, the groupPriority attribute shall have the value 'typ007:000 - undefined'. This Service Component Frame type also includes the recommended data CRC for bit error detection at Service Component Frame level.

< **ServCompFramePrioritisedProtected** > : =	: CRC protected Service Component Frame with message count
< **ServCompFrameHeader** > (header),	: Service Component Frame header as defined in A.2.2.6.1
< **typ007:Priority** > (groupPriority),	: group priority applicable to all messages in this ApplicationContent
external < **ApplicationContent** > (content),	: actual payload of the application
< **CRC** > (dataCRC);	: CRC starting with first byte after the Service Component Header CRC

A.2.2.6.2.4 Service Component Frame with dataCRC, groupPriority, and messageCount

Applications requiring both priority handling at Service Component Frame level and direct access to the number of messages that are available in a Service Component Frame can use the < ServCompFramePrioritisedCountedProtected > Service Component Frame type. This Service Component Frame type also includes the recommended data CRC for bit error detection at Service Component Frame level.

< **ServCompFramePrioritisedCountedProtected** > : =	: CRC protected Service Component Frame with group priority and message count
< **ServCompFrameHeader** > (header),	: Service Component Frame header as defined in A.2.3.3
< **typ007:Priority** > (groupPriority),	: group priority applicable to all messages in the ApplicationContent
< **IntUnTi** > (messageCount),	: count of messages in this ApplicationContent
external < **ApplicationContent** > (content),	: actual payload of the application
< **CRC** > (dataCRC);	: CRC starting with first byte after the Service Component Header CRC

A.2.3 Service Component level attributes

A.2.3.1 Service Component Identifier

The Service Component Identifier (SCID) with the value 0 is reserved for the SNI Application (ISO/TS 18234-3).

A.2.3.2 Field Length

The value of the Field Length attribute shall be equal to the number of bytes of the application data in the Service Component Frame, i.e. all bytes following the Service Component Header CRC attribute until the end of the last byte of Service Component frame.

A.2.3.3 Service Component Header CRC

The Service Component Header CRC provides error detection of the header of Service Component Frames. The CRC is two bytes long, and based on the ITU-T polynomial $x^{16} + x^{12} + x^5 + 1$.

The Service Component Header CRC is calculated from the Service Component Identifier, the Field Length and the first 13 bytes of the application data, i.e. the first 13 bytes following the header CRC. In the case of Component Data shorter than 13 bytes, the Component Identifier, the Field Length and all Component Data shall be taken into account.

The calculation of the CRC is described in Annex D.

A.2.3.4 Service Component Data CRC

The DataCRC is two bytes long, and is based on the ITU polynomial $x^{16} + x^{12} + x^5 + 1$. This CRC is calculated from all the bytes of the Service Component Frame Data, following the Service Component Header CRC.

The calculation of the CRC is described in Annex D.

Annex B
(normative)

tpegML Representation of Framework Structures

B.1 Introduction

The XML framing described in this annex allows TPEG data to be stored in XML documents. The structure is similar to the binary framing, but offers more use cases. While the TPEG binary framing is designed for the actual delivery of the data via telecommunication links to the end user, tpegML can, in addition, be used as an interface format to playout systems, or to store decoded TPEG binary data in receivers. The tpegML framing allows tpegML application level messages to be embedded in full featured XML documents.

The framing is supposed to suite two technical purposes:

— hold a one to one XML equivalent of binary tpeg data frames;

— allow to aggregate tpeg XML application messages within an XML document.

The following requirements have been collected and are the basis for the specified format:

a) The framing needs to represent the same structure as the binary framing including at least the StreamDirectory, SID, EncID, SCID and AID. This is important to allow full matching to the binary framing and to provide the full TPEG service infrastructure as specified in the main part of this document and in the TPEG SNI (ISO/TS 18234-3).

b) The format does not need to follow UML 2 XML modelling rules for the layers transport framing to service component framing, but for the application message level it is mandatory.

The document-based format shall:

a) be usable as an interface format to playout, as well as a delivery format for web service based transmission. This includes the ability to have binary and XML representations of the same data within one document at several framing levels.

b) allow transmission of all available data for a service in one document. This can then be used to provide a full stock of messages, e.g. after system failure or for periodic full updates.

c) allow bundling of messages in components, which may be streamed/saved in several documents. Despite requirement 4, it is necessary, to allow incremental updates of messages, without providing all messages. This requires message management to be available on both sides.

d) clearly indicate which of the two variants (requirements 4 or 5) is used. This is important to allow the systems to apply proper message management. For example, if a full message set is received, all messages in the system can be removed and the new messages can be imported. If only some of the messages are received, the receiver's message stock has to be updated by applying the message management rules.

e) have a binary (base64) representation of the data included at several framing levels, at least at application level/service component content. This allows encoding/decoding systems to use the format as interface format at various encoding/decoding levels.

f) include a document generation time stamp to indicate when the file has been created. This allows to always identify which document is the latest.

g) include versioning information about the used file format. This information can be used to identify the specification based on which a data file has been created.

h) allow to be extended to include meta information, e.g. for playout. Some information does not need to be transported to client devices over the air, but is relevant in playout systems, for example transmission statistics. The XML format may be extended to allow inclusion of such meta data.

i) allow to simultaneously include the same data as base64 and tpegML representation in one file. This allows easy handling and verification of binary encoding/decoding processes and storage of the results.

j) allow to include base64 encoded messages in the XML file. This allows for full independence of application level message generation and framing/multiplexing logic system components.

k) allow each level to include several versions of the data/specifications used on the included levels. This shall be indicated. For example, there may be two service component streams, one for version 1.0 of an application and one for 2.0 within one service to allow provision of the service to new and old devices at the same time.

Based on the mentioned requirements, tpegML framing has the following core features:

— All relevant parameters for unique message identification are represented in the XML framing, e.g. application id, service id and content id.

— It is possible to store a complete message set in one XML document, but also only parts of a message set may be stored in several files.

— It is possible to store the binary representation on several framing levels next to the XML representation. This allows the usage of the format to feed encoding systems, playout systems, as well as the usage as a meta format on receiver side.

— The framing is not derived from a UML model via the UML2XML rules, but where applicable TPEG data types have been used and the generic structures described in the main section of this document have been included.

— The tpegML XML framing format is specified via the full XML schema given in B.4. In the following sections the elements are described in detail.

— There is no need to modify this document to include new TPEG Applications. The new applications just need to interface to this document's ApplicationRootMessageML data type.

B.2 TPEG XML document structure

B.2.1 Diagrammatic hierarchy representation of the document structure

The format reflects the generic structure of the main document and the XML tags appear in the structure indicated in Figure B.1. Yellow highlighted tags (or all boxes containing text ending in "Bin") carry base64 encoded binary representation of TPEG data. The colours blue and white are only used for better readability of the diagrams. The numbers in brackets give an indication of the multiplicity of the tags. The used XML attributes are not included in these diagrams.

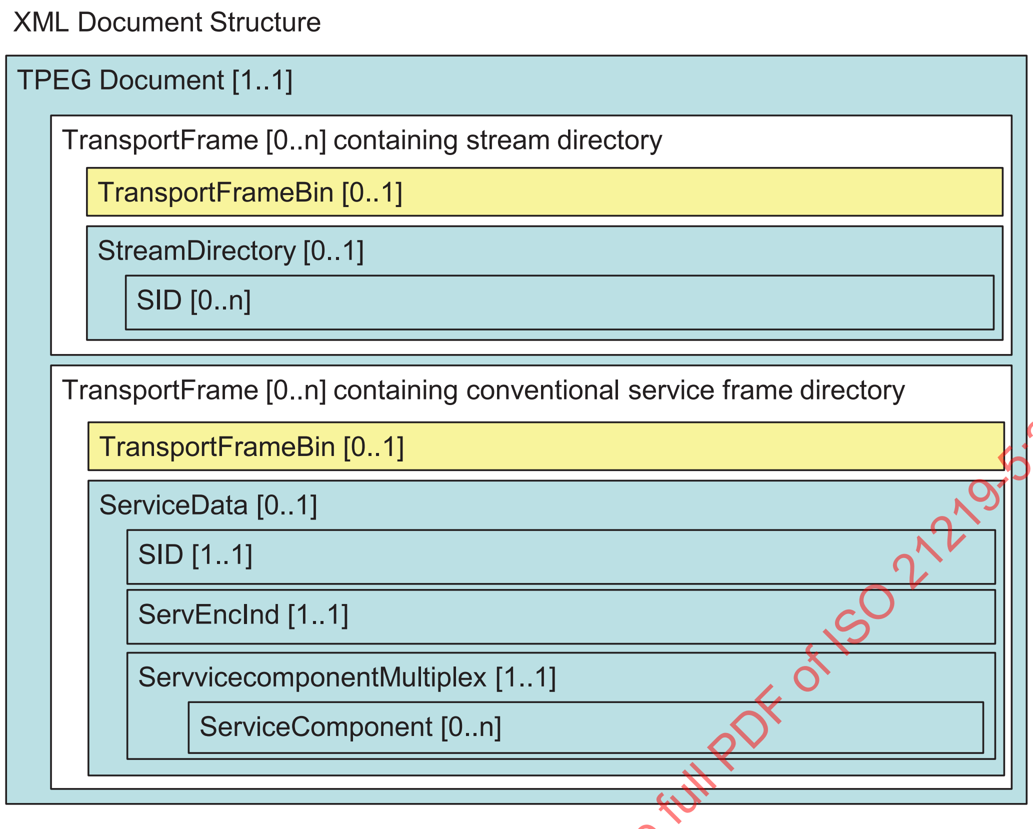

Figure B.1 — XML document structure

All boxes with text ending in "Bin" are highlighted in yellow colour in the diagram for better visibility.

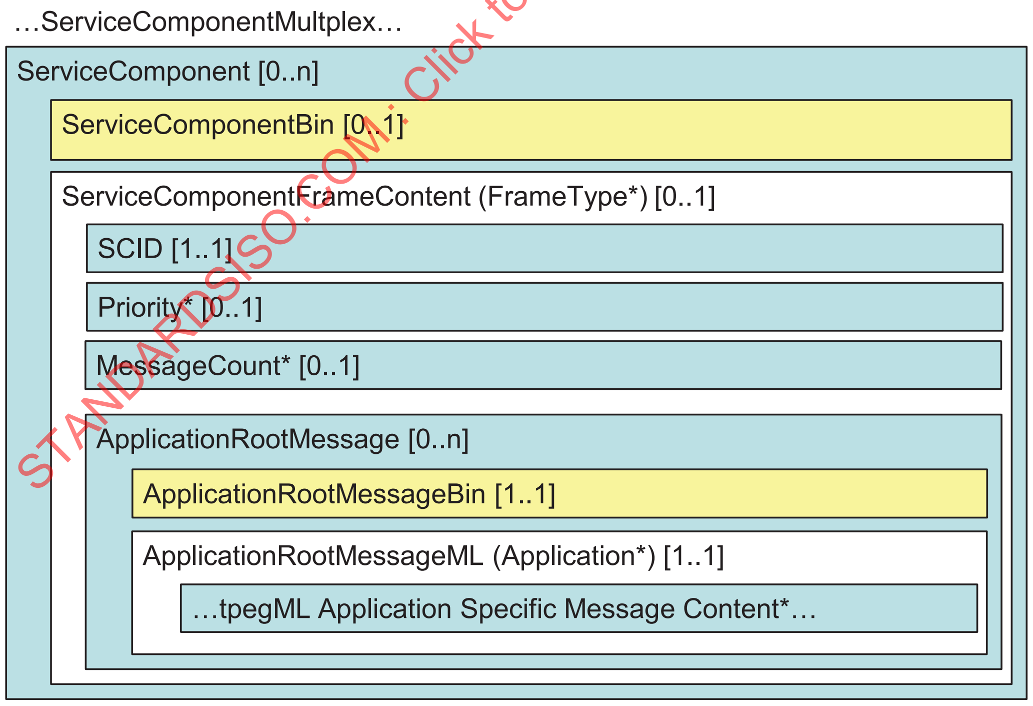
…ServiceComponentMultplex…
ServiceComponent [0..n]
ServiceComponentBin [0..1]
ServiceComponentFrameContent (FrameType*) [0..1]
SCID [1..1]
Priority* [0..1]
MessageCount* [0..1]
ApplicationRootMessage [0..n]
ApplicationRootMessageBin [1..1]
ApplicationRootMessageML (Application*) [1..1]
…tpegML Application Specific Message Content*…

Figure B.2 — Detailed view of the Service Component Multiplex

All boxes with text ending in "Bin" are highlighted in yellow colour in the diagram for better visibility.

While Figure B.1 describes the structure up to the Service Component Multiplex, the details for the Service Components are shown in Figure B.2.

(*) The TPEG Application specific data are included within the ApplicationRootMessageML tag according to the related application specification and is indicated in Figure B.2 with an asterisk.

At several framing levels it is possible to include either a binary representation, or an XML representation of the same content, or both. Once a binary representation child element is present, the respective XML content tag shall only include the number and content of messages that are actually encoded in the binary representation and vice versa. However, if problems appear in the encoding or decoding process, subsets of the data may be present in one of the representations.

B.2.2 Syntactical schema representation of the tpegML document

B.2.2.1 Document headers

The headers include the encoding instructions, as well as namespace definitions. The namespace definitions and schema locations shall be modified according to the versions of the related documents. For example, the namespace "http://www.tisa.org/TPEG/TPEGDataTypes_2_0" matches the version of the applied TPEG2 UML to XML Conversion Rules (TPEG2 Part 4), which is used for the XSD and XML examples hereunder; however, other versions are also possible.

XSD:

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns="http://www.tisa.org/TPEG/SFW_1_1" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tdt="http://www.tisa.org/TPEG/TPEGDataTypes_2_0"
targetNamespace="http://www.tisa.org/TPEG/SFW_1_1"
elementFormDefault="qualified" attributeFormDefault="qualified">

<xs:import namespace="http://www.tisa.org/TPEG/TPEGDataTypes_2_0" schemaLocation="TDT_2_0.xsd" />

< ?xml version = "1.0" encoding = "UTF-8"? >
```

XML:

```
< ?xml version = "1.0" encoding = "UTF-8"? >
```

B.2.2.2 Special data types and tags

B.2.2.2.1 XML Attributes

In tpegML all application data are encoded as XML elements despite their UML Class/Attribute property. However, in this XML framing, XML attributes are used to provide meta data that is not directly included in the TPEG framing structure, but is relevant for encoding or decoding of the data contained in the elements.

B.2.2.2.2 TPEG Binary Data Type

XML type name: TPEGbin

On three levels it is foreseen to include TPEG binary data as base64 encoded content, that is encoded according to Annex A or TPEG2 Part 3: UML to Binary Conversion Rules.

The TPEGbin type provides attributes to give meta information about the content and its processing stage.

Attribute	Description
byteSize [1..1]	Number of bytes used by the included TPEG binary data before base64 encoding took place. Knowing this size allows pre-allocation of memory/space for the base64 decoded data in outer framings, before actually base64 decoding the data.
byteCRChex [0..1]	4 hex digit CRC value according to the Service Framework method calculated over the TPEG binary data before base64 encoding. This value is useful to ensure, that the binary data are still valid, even after base64 en/decoding and XML processing. This attribute is optional.
statusLevel [0..1]	This value indicates whether the binary data has had any issues when it was created. When encoding or decoding data, problems may appear, e.g. unknown tags or invalid values, version problems, etc. This attribute is optional. Values: — Success: Binary data are valid, no issues during binary creation — Error: Data are probably not usable. — Warning: Data are usable, but may be not complete.
statusDescription [0..1]	A descriptive text that may give additional information about the binary encoded data related to the statusLevel value, e.g. version of used encoder/decoder software. This attribute is optional.
serviceComponentIsEncrypted [0..1]	The XML framing allows to carry data at various processing levels, although the encryption indicators represent the desired over the air situation. For example one document may carry unencrypted service components, which are then fed to a scrambler and the resulting document may carry the scrambled data so an indicator is needed to allow systems to identify whether to apply scrambling/descrambling or not. If this attribute is present and set to 'True', the binary service content included within this element is encrypted. If set to 'False' the content is not encrypted, even if the encryption indicator in the SNI indicates otherwise. This is possible if the format is used for feeding of playout systems, where the data are not encrypted at some processing stages or in decoders, where decryption may have happened before decoding the actual content. This attribute is optional. If the attribute is not present, the data are supposed to be encoded as indicated by the related encryption indicator. Related CRCs within the data have to match the actual state of the content. This flag shall not occur if the framing is used for over the air or end user delivery, where the signalled values always have to match the actual data.

XSD:

```
  < xs:complexType name = ”TPEGbin” >
  < xs:simpleContent >
< xs:extension base = ”xs:base64Binary” >
  < xs:attribute name = ”byteSize” type = ”tdt:IntUnLi” use = ”required”/ >
      <!--size of the base64decoded data -->
  < xs:attribute name = ”byteCRChex” type = ”xs:string”/ >
      <!-- crc of the base64decoded data -->
  < xs:attribute name = ”statusLevel” type = ”statusLevel”/ >
  < xs:attribute name = ”statusDescription” type = ”xs:string”/ >
      <!-- OK, Warning, Error, WarningLostContent, ErrorInvalidFrame, ErrorMissingContent,
       ErrorTooMuchContent -->
  < xs:attribute name = ”isEncrypted” type = ”xs:boolean”/ >
< /xs:extension >
  < /xs:simpleContent >
  < /xs:complexType >
  < xs:simpleType name = ”statusLevel” >
    < xs:restriction base = ”xs:string” >
      < xs:enumeration value = ”Success”/ >
      < !–statusDescription is informal only– >
      < xs:enumeration value = ”Error”/ >
      < !–the data are probably not usable– >
      < xs:enumeration value = ”Warning”/ >
      < !– the frame is syntactically correct, but content, may have been lost– >
    < /xs:restriction >
  < /xs:simpleType >
```

XML:

```
< ApplicationRootMessageBin byteSize = ”17”
statusLevel = ”Success” > TWFydGluIERyZWhlciA7LSk = < /ApplicationRootMessageBin >
```

B.2.2.3 TPEG Document

XML element tag name: TPEGDocument

Child element names: TransportFrame

This is the root tag of the document and includes the namespace definitions. It includes a timestamp as well as an indicator of whether the document contains all messages of the stock (DocumentType = fullRepository), or only parts of it (DocumentType = partialRepository). Additionally, the version of this specification XSD is given.

The TPEGDocument may only contain TransportFrame elements, which represent the transport framing level, one transport frame each.

The following attributes represent meta information for the TPEG document.

Attribute	Description
timeStamp [0..1]	The timestamp represents the time, when the XML document was created.
version [1..1]	For this version of the tpegML framing, the value of „3“ shall be used.

Attribute	Description
docType [1..1]	fullRepository: All messages of the stock are contained in the document and fully replace the stock of the receiving system. partialRepository: Only some (updated) messages are in the document. The receiving system shall only update its stock with the messages contained in this document by applying the message management rules.

XSD:

```
< xs:element name = ”TPEGDocument” type = ”TPEGDocument”/ >
  < xs:complexType name = ”TPEGDocument” >
    < xs:sequence >
      < xs:element name = ”TransportFrame” type = ”TransportFrame” maxOccurs = ”unbounded”/ >
    < /xs:sequence >
    < xs:attribute name = ”timeStamp” type = ”tdt:DateTime”/ >
    < xs:attribute name = ”version” type = ”tdt:IntUnTi” use = ”required”/ >
    < xs:attribute name = ”docType” type = ”DocumentType” use = ”required”/ >
  < /xs:complexType >
  < xs:simpleType name = ”DocumentType” >
    < xs:restriction base = ”xs:string” >
      < xs:enumeration value = ”fullRepository”/ >
      < xs:enumeration value = ”partialRepository”/ >
    < /xs:restriction >
  < /xs:simpleType >
```

XML:

```
<TPEGDocument xmlns="http://www.tisa.org/TPEG/SFW_1_1" xmlns:tdt="http://www.tisa.org/TPEG/TPEGDataTypes
_2_0" xmlns:sfw="http://www.tisa.org/TPEG/SFW_1_1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=http://www.tisa.org/TPEG/SFW_1_1
SFW_1_1.xsd http://www.tisa.org/TPEG/TPEGDataTypes_2_0 TDT_2_0.xsd
sfw:timeStamp="2009-11-25T08:16:30Z" sfw:version="2" sfw:docType="fullRepository">
```

B.2.2.4 Transport Frame Level

XML type/tag name: TransportFrame

Child element names: TransportFrameBin, StreamDirectory, ServiceData

This element represents the content of the transport frame level. If no TransportFrameBin child is present, it may as well contain more content than actually fits into a TPEG binary transport frame. A TPEG binary encoder will have to split the content to several TransportFrames later on.

It can contain next to an optional TransportFrameBin element, either one StreamDirectory (Transport Frame Type 0), or one ServiceData (Transport Frame Type 1). Further frame types may be added in the future, but always only one type shall be included next to the TransportFrameBin element.

XSD:

```
< xs:complexType name = ”TransportFrame” >
 < xs:sequence >
  < xs:element name = ”TransportFrameBin” type = ”TPEGbin” minOccurs = ”0”/ >
  < xs:choice minOccurs = ”0” >
   < xs:element name = ”StreamDirectory” type = ”StreamDirectoryFrame”/ >
   < xs:element name = ”ServiceData” type = ”ServiceFrame”/ >
  < /xs:choice >
 < /xs:sequence >
< /xs:complexType >
```

B.2.2.5 Transport Frame Binary Representation

XML element tag name: TransportFrameBin

XML type name: TPEGBin

This element contains the fully encoded TPEG Transport Frame, according to the TPEG binary representation as stated in Annex A that is equivalent to the other child component of the TransportFrame element. This element shall not be used, if other elements on this level (e.g. ServiceData) contain more tpegML messages than are present in the binary frame.

B.2.2.6 Stream Directory

XML type name: StreamDirectoryFrame

XML element tag name: StreamDirectory

The stream directory contains a list of TPEG Service ids that occur in the ISO/TS 21219 series.

XSD:

```
< xs:complexType name = ”StreamDirectoryFrame” >
 < xs:sequence >
  < xs:element name = ”SID” type = ”tdt:ServiceIdentifier” maxOccurs = ”unbounded”/ >
 < /xs:sequence >
< /xs:complexType >
```

XML:

```
< StreamDirectory >
< SID >
  < tdt:SID_A > 0 < /tdt:SID_A >  < tdt:SID_B > 0 < /tdt:SID_B >  < tdt:SID_C > 1 < /tdt:SID_C >
< /SID >
< SID >
  < tdt:SID_A > 0 < /tdt:SID_A >  < tdt:SID_B > 0 < /tdt:SID_B >  < tdt:SID_C > 2 < /tdt:SID_C >
< /SID >
< /StreamDirectory >
```

B.2.2.7 Service Frame Level

XML type name: ServiceFrame

XML element tag name: ServiceData

The ServiceFrame element includes the mandatory SID, ServEncID and ServiceComponentMultiplex elements. There is no TPEGBin element at this level, as the difference between the transport level and the service level are not significant for XML framing.

XSD:

```
< xs:complexType name = ”ServiceFrame” >
   < xs:sequence >
    < xs:element name = ”SID” type = ”tdt:ServiceIdentifier” minOccurs = ”0”/ >
    < xs:element name = ”ServEncID” type = ”EncryptionIndicator” minOccurs = ”0”/ >
    < xs:element name = ”ServiceComponentMultiplex” type = ”ServCompMultiplex”/ >
   < /xs:sequence >
< /xs:complexType >
```

XML:

```
  < ServiceFrame >
   < SID >
    < tdt:SID_A > 0 < /tdt:SID_A >  < tdt:SID_B > 0 < /tdt:SID_B >  < tdt:SID_C > 1 < /tdt:SID_C >
   < /SID >
   < ServEncID > 0 < /ServEncID >
   < ServiceComponentMultiplex >
…
   < /ServiceComponentMultiplex >
   < /ServiceFrame >
```

B.2.2.7.1 Service Identification

XML type name: ServiceIdentifier

XML element tag name: SID

This element contains the TPEG Service id to which the content belongs. See B.2.2.7 for the definition.

B.2.2.7.2 Service Encryption Indicator

XML type name: EncryptionIndicator

XML element tag name: ServEncID

The value of the encryption indicator that is valid for the binary representation at Service Frame level (see Annex A). This element does not specify an encryption for the XML representation of the data. The value shall be set to the appropriate value of the binary data, even if the data within the XML document has been decrypted or not yet encrypted. Playout systems may use this value to trigger proper encoding. On the decoder side, the value is listed to allow a decoder to descramble encoded TPEG binary content with an „isEncrypted“-Flag set to true.

XSD:

```
< xs:simpleType name = ”EncryptionIndicator” >
  < xs:restriction base = ”tdt:IntUnTi” >
    < xs:minInclusive value = ”0”/ >
    < xs:maxInclusive value = ”255”/ >
  < /xs:restriction >
< /xs:simpleType >
```

B.2.2.7.3 Service Component Multiplex

XML type name: ServCompMultiplex

XML element tag name: ServiceComponentMultiplex

This element hosts any number of ServiceComponent elements.

B.2.2.8 Service Component Level

XML type name: ServiceComponent

XML element tag name: ServiceComponent

A ServiceComponent can include either a binary representation, or an XML representation of its content, or both. Once a ServiceComponentBin child element is present, the ServiceComponentFrameContent shall only include the number of messages that are actually encoded in the binary representation.

To allow decoding of the content without having to parse the SNI application, it is possible to give the relevant values as attributes to the ServiceComponent element. During TPEG reception these values have to be extracted by a binary decoder from the SNI GST1 Fast Tuning Table.

XML-Attribute	Description
applicationId [0..1]	Application ID as defined in TPEG INV.
contentId [0..1]	See SNI GST1 Fast Tuning Table.
majorApplicationVersion [0..1]	Version of the application specification. See SNI specification.
minorApplicationVersion [0..1]	Version of the application specification. See SNI specification.
encId [0..1]	If this attribute is present and not equal „0“ it indicates the method used for binary encryption of the „on air“ data. However this does not indicate any encryption of or within the XML content. Whether the actual binary content is encrypted is indicated in the TPEGBin element „isEncrypted“ flag.
safetyFlag [0..1]	Is defined in the SNI Fast Tuning Table and identifies whether the content is safety relevant. If set to „True“, the messages are safety relevant.
originatorSID_A [0..1] originatorSID_B [0..1] originatorSID_C [0..1]	OriginatorSID as decimal integer A.B.C (see SNI specification). If one of the originator SID appears, the two others are considered as mandatory.

XSD:

```
 < xs:complexType name = ”ServiceComponent” >
   < xs:sequence >
     < xs:element name = ”ServiceComponentBin” type = ”TPEGbin” minOccurs = ”0”/ >
     < xs:element name = ”ServiceComponentFrameContent” type = ”ServiceComponentFrameContent”
minOccurs = ”0”/ >
   < /xs:sequence >
   < xs:attribute name = ”applicationId” type = ”tdt:IntUnLi”/ >
   < xs:attribute name = ”contentId” type = ”tdt:IntUnTi”/ >
   < xs:attribute name = ”majorApplicationVersion” type = ”tdt:IntUnTi”/ >
   < xs:attribute name = ”minorApplicationVersion” type = ”tdt:IntUnTi”/ >
   < xs:attribute name = ”encId” type = ”tdt:IntUnTi”/ >
   < xs:attribute name = ”safetyFlag” type = ”xs:boolean”/ >
   <xs:attribute name="originatorSID_A" type="tdt:IntUnTi"/>
   <xs:attribute name="originatorSID_B" type="tdt:IntUnTi"/>
   <xs:attribute name="originatorSID_C" type="tdt:IntUnTi"/>
 < /xs:complexType >
```

B.2.2.9 Service Component Binary Representation

XML type name: TPEGbin

XML element tag name: ServiceComponentBin

This element contains the fully encoded TPEG Service Component frame, according to the TPEG binary representation as stated in Annex A that is equivalent to the other child component of the ServiceComponent element.

B.2.2.10 Service Component Frame Content

XML type name: ServiceComponentFrameContent

XML element tag name: ServiceComponentFrameContent

Child element names: SCID, MessageCount, Priority, ApplicationRootMessage

This is the XML representation of the Service Component Frame level, including the SCID as used in the binary format to link to the SNI service component. Additionally, the most important values from SNI GST1 are included as attributes in the XML representation, to allow interpretation without having to decode the actual SNI data.

Depending on the application type, the appropriate framing, the child elements are chosen. Header/data CRCs are not taken into account for the XML representation.

The ServiceComponetFrameContent element can be of one of several framing types similar to the binary representation. This is done by instantiating one of the frame types derived from the abstract ServiceComponentFrameContent so that the only element that is present in all Service Component Frames is the ServiceComponentIdentifier element. Further elements (Priority, MessageCount, ApplicationRootMessage) are than added according to the actual framing.

The following frame types are defined:

— ServCompFrameProtected

— ServCompFrameCountedProtected

— ServCompFramePrioritisedProtected

— ServCompFramePrioritisedCountedProtected

In the XML framing, the protected Service Component Frames do not contain the data CRC, because the data CRC is not considered for the XML format. This implies that an unprotected frame would be listed with the same elements as a protected one.

XSD:

```
 < xs:complexType name = ”ServiceComponentFrameContent” abstract = ”true” >
  < xs:sequence >
   < xs:element name = ”SCID” type = ”ServiceComponentIdentifier”/ >
  < /xs:sequence >
 < /xs:complexType >

 < xs:complexType name = ”ServCompFrameProtected” >
  < xs:complexContent >
   < xs:extension base = ”ServiceComponentFrameContent” >
    < xs:sequence >
     < xs:element name = ”ApplicationRootMessage” type = ”ApplicationRootMessage” minOccurs = ”0”
maxOccurs = ”unbounded”/ >
     < !– binCRC not suitable in xml– >
    < /xs:sequence >
   < /xs:extension >
  < /xs:complexContent >
 < /xs:complexType >
 < xs:complexType name = ”ServCompFrameCountedProtected” >
  < xs:complexContent >
   < xs:extension base = ”ServiceComponentFrameContent” >
    < xs:sequence >
     < xs:element name = ”MessageCount” type = ”tdt:IntUnTi” minOccurs = ”0”/ >
     < xs:element name = ”ApplicationRootMessage” type = ”ApplicationRootMessage” minOccurs = ”0”
maxOccurs = ”unbounded”/ >
     < !– binCRC not suitable in xml– >
    < /xs:sequence >
   < /xs:extension >
  < /xs:complexContent >
 < /xs:complexType >
 < xs:complexType name = ”ServCompFramePrioritisedProtected” >
  < xs:complexContent >
```

```
    < xs:extension base = "ServiceComponentFrameContent" >
     < xs:sequence >
      < xs:element name = "Priority" type = "tdt:typ007_Priority"/ >
      < xs:element name = "ApplicationRootMessage" type = "ApplicationRootMessage" minOccurs = "0"
maxOccurs = "unbounded"/ >
      < !– binCRC not suitable in xml– >
     < /xs:sequence >
    < /xs:extension >
   < /xs:complexContent >
  < /xs:complexType >
  < xs:complexType name = "ServCompFramePrioritisedCountedProtected" >
   < xs:complexContent >
    < xs:extension base = "ServiceComponentFrameContent" >
     < xs:sequence >
      < xs:element name = "Priority" type = "tdt:typ007_Priority"/ >
      < xs:element name = "MessageCount" type = "tdt:IntUnTi" minOccurs = "0"/ >
      < xs:element name = "ApplicationRootMessage" type = "ApplicationRootMessage" minOccurs = "0"
maxOccurs = "unbounded"/ >
      < !– binCRC not suitable in xml– >
     < /xs:sequence >
    < /xs:extension >
   < /xs:complexContent >
  < /xs:complexType >
```

B.2.2.10.1 Service Component Identifier

XML type name = ServiceComponentIdentifier

XML element tag name: SCID

The ServiceComponentIdentifier (SCID) is used to identify a specific service component type and to determine the relevant information from the service and network Information. The ServiceComponent with the value „0“ is reserved for the SNI application.

XSD:

```
  < xs:simpleType name = "ServiceComponentIdentifier" >
   < xs:restriction base = "tdt:IntUnTi" >
    < xs:minInclusive value = "0"/ >
    < xs:maxInclusive value = "255"/ >
   < /xs:restriction >
  < /xs:simpleType >
```

B.2.2.10.2 Priority

XML element tag name: Priority

If the Priority element is present, it contains the priority for decoding as defined in the relevant tdt table. If messages of several priorities are aggregated within one ServiceComponentFrameContent element, a value of „0“ shall be chosen.

B.2.2.10.3 Message Count

XML element tag name: MessageCount

This value can be considered as informative/for binary compatibility only within the XML framing, the available number of ApplicationRootMessageTags should be used. However, it shall match the number of messages in the binary format, if available.

Although the messageCount is limited to 255 within the binary framing, storing more than 255 tpegML messages within one XML ServiceComponentFrame is allowed. Encoders have to split the allowed number of messages into several binary frames. If the MessageCount tag is present, it shall contain the number of ApplicationRootMessage tags in the Service Component Frame.

B.2.2.10.4 Application Root Message Level

XML type name: ApplicationRootMessage

XML element tag name: ApplicationRootMessage

Child element names: ApplicationRootMessageBin, ApplicationRootMessageML

This is the element that represents the actual application specific data unit. To match the term MessageCount, it is called „Message“, but may contain any application specific data unit.

For each of these elements two children can be included to allow binary and XML representation of the same content at this level.

XSD:

```
< xs:complexType name = ”ApplicationRootMessage” >
 < xs:sequence >
  < xs:element name = ”ApplicationRootMessageBin” type = ”TPEGbin” minOccurs = ”0”/ >
  < xs:element name = ”ApplicationRootMessageML” type = ”ApplicationRootMessageML” minOccurs = ”0”/ >
 < /xs:sequence >
< /xs:complexType >
```

B.2.2.10.5 Application Root Message Binary Representation

XML type name: TPEGbin

XML element tag name: ApplicationRootMessageBin

This element contains the fully encoded TPEG binary message, according to the TPEG binary representation as stated in Annex A that is equivalent to the other child component of the ApplicationRootMessage element.

B.2.2.10.6 Application Root Message XML Representation

XML element tag/type name: ApplicationRootMessageML

This is the abstract XML application root element that is instantiated by the element tagged with „ApplicationRoot“ in the UML. All child elements are specified according to the UML2XML Conversion Rules specification (TPEG2 Part 4: UML to XML conversion rules).

XSD:

```
< xs:complexType name = ”ApplicationRootMessageML” abstract = ”true”/ >
```

All application specific namespaces can be defined for the deriving element.

Application XSD example:

```
< xs:complexType name = ”TFPMessage” >
  < xs:complexContent >
   < xs:extension base = ”tsf:ApplicationRootMessageML” >
    < xs:sequence >
     < xs:element name = ”mmt” type = ”mmc:MMCTemplate” / >
      < xs:element name = “method” type = ”TFPMethod” minOccurs = ”0” maxOccurs = ”unbounded” / >
      < xs:element name = ”loc” type = ”lrc:LocationReferencingContainer” minOccurs = ”0” / >
     < /xs:sequence >
    < /xs:extension >
   < /xs:complexContent >
  < /xs:complexType >
```

Application XML example:

```
<ApplicationRootMessageML xsi:type="tfp:TFPMessage"
xmlns:tfp = http://www.tisa.org/TPEG/TFP_1_1
xmlns:mmc = http://www.tisa.org/TPEG/MMC_1_1
xmlns:lrc = http://www.tisa.org/TPEG/LRC_2_1
xmlns:tlr = "http://www.tisa.org/TPEG/TLR_2_0"
xsi:schemaLocation="http://www.tisa.org/TPEG/TFP_1_1 TFP_1_1.xsd" >
```

B.3 XML example document

```
<?xml version="1.0" encoding="UTF-8"?>

<TPEGDocument xmlns=http://www.tisa.org/TPEG/SFW_1_1
xmlns:tdt=http://www.tisa.org/TPEG/TPEGDataTypes_2_0
xmlns:sfw=http://www.tisa.org/TPEG/SFW_1_1
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=http://www.tisa.org/TPEG/SFW_1_1SFW_1_1.xsd
http://www.tisa.org/TPEG/TPEGDataTypes_2_0TDT_2_0.xsd
sfw:timeStamp="2017-11-25T08:16:30Z" sfw:version="2" sfw:docType="fullRepository">

    <TransportFrame>

        <ServiceData>

            <SID>

                <tdt:SID_A>0</tdt:SID_A>

                <tdt:SID_B>0</tdt:SID_B>

                <tdt:SID_C>0</tdt:SID_C>

            </SID>

            <ServiceComponentMultiplex>

                <ServiceComponent sfw:applicationId="5" sfw:contentId="2"
                sfw:majorApplicationVersion="3" sfw:minorApplicationVersion="0"
                sfw:encId="0" sfw:safetyFlag="true" originatorSID_A = "0" originatorSID_B = "0"

                originatorSID_C = "0" >

                    <ServiceComponentFrameContent
                    xsi:type="ServCompFramePrioritisedCountedProtected">
```

```
                        <SCID>2</SCID>

                        <Priority tdt:table="typ007_Priority" tdt:code="0" />

                        <MessageCount>199</MessageCount>

                        <ApplicationRootMessage>
                        <!-- <ApplicationRootMessageML xsi:type="tec:TECMessage"
                            xmlns:tec=http://www.tisa.org/TPEG/TEC_3_2
                            xmlns:mmc="http://www.tisa.org/TPEG/MMC_1_1"
                            xmlns:lrc="http://www.tisa.org/TPEG/LRC_2_1"
                            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                            xsi:schemaLocation="http://www.tisa.org/TPEG/TEC_3_2
                            http://www.tisa.org/TPEG/TEC_3_2.xsd"> <tec:mmt
                            xsi:type="mmc:MessageManagementContainer">
                            </ApplicationRootMessageML> -->

                        </ApplicationRootMessage>

                    </ServiceComponentFrameContent>

                </ServiceComponent>

            </ServiceComponentMultiplex>

        </ServiceData>

    </TransportFrame>

</TPEGDocument>
```

B.4 Full tpegML XSD Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns=http://www.tisa.org/TPEG/SFW_1_1
    xmlns:xs=http://www.w3.org/2001/XMLSchema
    xmlns:tdt=http://www.tisa.org/TPEG/TPEGDataTypes_2_0
    targetNamespace="http://www.tisa.org/TPEG/SFW_1_1"
    elementFormDefault="qualified" attributeFormDefault="qualified">

    <xs:import namespace="http://www.tisa.org/TPEG/TPEGDataTypes_2_0"
        schemaLocation="TDT_2_0.xsd" />

    <xs:element name="TPEGDocument" type="TPEGDocument" />

    <xs:complexType name="TPEGDocument">

        <xs:sequence>

            <xs:element name="TransportFrame" type="TransportFrame"
                maxOccurs="unbounded" />

        </xs:sequence>

        <xs:attribute name="timeStamp" type="tdt:DateTime" />

        <xs:attribute name="version" type="tdt:IntUnTi" use="required" />

        <xs:attribute name="docType" type="DocumentType" use="required" />

    </xs:complexType>
```

```
<xs:complexType name="TransportFrame">
    <xs:sequence>
        <xs:element name="TransportFrameBin" type="TPEGbin" minOccurs="0" />
        <xs:choice minOccurs="0">
            <xs:element name="StreamDirectory" type="StreamDirectoryFrame" />
            <xs:element name="ServiceData" type="ServiceFrame" />
        </xs:choice>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="StreamDirectoryFrame">
    <xs:sequence>
        <xs:element name="SID" type="tdt:ServiceIdentifier" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceFrame">
    <xs:sequence>
        <xs:element name="SID" type="tdt:ServiceIdentifier" minOccurs="0" />
        <xs:element name="ServEncID" type="EncryptionIndicator" minOccurs="0" />
        <xs:element name="ServiceComponentMultiplex" type="ServCompMultiplex" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ServCompMultiplex">
    <xs:sequence>
        <xs:element name="ServiceComponent" type="ServiceComponent"
            minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="ServiceComponent">
    <xs:sequence>
        <xs:element name="ServiceComponentBin" type="TPEGbin" minOccurs="0" />
        <xs:element name="ServiceComponentFrameContent"
            type="ServiceComponentFrameContent" minOccurs="0" />
    </xs:sequence>
    <xs:attribute name="applicationId" type="tdt:IntUnLi" />
```