IP Datacast over DVB-H: Electronic Service Guide (ESG)

DVB Document A099

November 2005

Contents

Introduction

IP Datacast over DVB-H is an end-to-end broadcast system for delivery of any types of digital content and services using IP-based mechanisms optimized for devices with limitations on computational resources and battery. An inherent part of the IPDC system is that it comprises of a unidirectional DVB broadcast path that may be combined with a bi-directional mobile/cellular interactivity path. IPDC is thus a platform that can be used for enabling the convergence of services from broadcast/media and telecommunications domains (e.g., mobile / cellular).

1 Scope

Electronic Service Guide (ESG) contains information about the services available. Through the information in the ESG, the user can select the services and items he/she is interested in and find stored items on the terminal.

The present document defines the datamodel, the representation format, the encapsulation and the transport of the Electronic Service Guide of DVB-H [1].

2 References

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

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

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

[1] ETSI EN 302 304: “Transmission System for Handheld Terminals”.

[2] IETF RFC 2046: “Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types”, N. Freed, N. Borenstein, November 1996.

[3] ISO/IEC 15938-1, Information technology -- Multimedia content description interface -- Part 1: Systems.

[4] ISO/IEC 15938-5, Information technology -- Multimedia content description interface -- Part 5: Multimedia Description Schemes.

[5] DVB BlueBook A101: IP Datacast over DVB-H: Content Delivery Protocols (CDP).

[6] ETSI TS 102 005: Digital Video Broadcast, Specification for the use of video and audio coding in DVB services delivered directly over IP.

[7] IETF RFC 1952: “GZIP File Format Specification Version 4.3”, P. Deutsch, May 1996.

[8] ETSI TS 102 323: Digital Video Broadcasting (DVB);Carriage and signaling of TV-Anytime information in DVB transport streams

[9] IETF RFC 3450: “Asynchronous Layered Coding (ALC) Protocol Instantiation”, M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft, December 2002.

[10] IETF RFC 3926: “FLUTE - File Delivery over Unidirectional Transport”, T. Paila et al., October 2004, http://www.ietf.org/rfc/rfc3926.txt.

[11] IETF RFC 3451: “Layered Coding Transport (LCT) Building Block”, M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, J. Crowcroft, December 2002.

[12] ETSI TS 102 822-3-1: Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1");Part 3: Metadata; Sub-part 1: Metadata schemas

[13] IANA, “Internet Mutlicast Addresses”, http://www.iana.org/assignments/multicast-addresses, 9th September 2005.

[14] IETF RFC 3629: “UTF-8, a transformation format of ISO 10646”, F. Yergeau, November 2003.

[15] 3GPP TS 26.346 v6.1.0: “Multimedia Broadcast/Multicast Service; Protocols and Codecs (Release 6)”

[16] ANSI/IEEE Std 754-1985: “Standard for Binary Floating-Point Arithmetic”

[17] XML Schema, W3C Recommendation, 2nd May 2001.

[18] DVB CAS http://www.dvb.org/index.php?id=16

[19] IETF RFC 2616: “Hypertext Transfer Protocol -- HTTP/1.1”

[20] IANA, “Hypertext Transfer Protocol Parameters”, http://www.iana.org/assignments/http-parameters, 1st May 2001.

[21] ETSI TS 102 822-3-2 Broadcast and On-line Services: Search, select, and rightful use of content. on personal storage systems ("TV-Anytime Phase 1"; Part 3: Metadata; Sub-part 2: System aspects in a uni-directional environment.

[22] IANA, “Port Numbers”, http://www.iana.org/assignments/port-numbers, 16th September 2005.

[23] IANA, “Internet Protocol Version 6 Mutlicast Addresses”, http://www.iana.org/assignments/ipv6-multicast-addresses, 9th September 2005.

[24] DVB BlueBook A098: IP Datacast over DVB-H: Architecture

[25] ETSI TR 101 162 "Digital broadcasting systems for television, sound and data services; Allocation of Service Information (SI) codes for Digital Video Broadcasting (DVB) systems"

3 Definitions and abbreviations

3.1 Definitions

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

IPDCKMSId: an id, which is assigned by DVB to every Key Management System (see CA systems at http://www.dvb.org/products_registration/dvb_identifiers/).

Context Path Code: A code assigned to a ContextPath in Table 6.2 or in the DecoderInit to signal ESG XML Fragment Types.

Datatype: Describes in this document an XML Schema datatype [17].

Encapsulated ESG XML Fragment: A representation of an ESG XML Fragment, which also contains header information for instance the Context Path Code of the ESG XML Fragment.

Encapsulated ESG Fragment: A representation of an ESG Fragment, which also contains header information of the ESG Fragment.

ESG Auxiliary Data: ESG data, which is referenced from an instance of the XML based Data Model e.g. an SDP file, an HTML page or a PNG file.

Context Path: In the XML terminology the Context Path identifies the context of an element and its datatype in an XML Instance by describing the path from the root element to that element. The context of an element is defined in this document by all parent elements of that element and their datatypes.

Current ESG XML Document: An instantiation of the ESGMain Element together with all ESG XML Fragments transported in the ESG Fragment Stream carousel at a particular point in time. Note that the ESGMain Element can be explicitly or implicitly be signaled in an ESG Fragment Stream.

Current ESG: A consistent set of ESG Fragments transported in the ESG Fragment Stream carousel at a particular point in time. Note that the ESGMain Element can be explicitly or implicitly be signaled in an ESG Fragment Stream.

ESG Fragment: A fragment of ESG data delivered in the ESG stream and referred to by a fragment reference in the encapsulation structure. Namely an ESG Fragment can be according to this specification an ESG XML Fragment, ESG Auxiliary Data or Private Auxiliary data.

ESG Container: A structure to group ESG data into one transport object for delivery purposes.

ESG Fragment Stream: A stream of ESG Fragments which contributes to the same ESG on receiver end. With respect to the transport layer this ESG Fragment stream can be compiled from several transport Streams e.g. IP Streams.

ESG Fragment Type: The ESG Fragment Type is a category of ESG Fragment e.g. ESG XML Fragment, ESG Auxiliary Data or Private Auxiliary Data.

ESG Init Container: An ESG Container carrying data structures for initialization e.g. the ESG Init Message and the ESG Main Fragment.

ESG Init Message: Initialisation information to decode ESG Fragments.

ESG XML Fragment: An ESG Fragment of an XML instance which is an instantiation of a datatype. A limited set of ESG XML Fragment Types have been defined in this specification document.

ESG XML Fragment Type: According to the ESG data model defined in this specification the ESG XML Fragment Type is a category of ESG XML Fragments. The ESG XML Fragment Types are defined based on the ContextPath which identifies an element and its datatype in an XML Instance. These ContextPaths and the related datatypes are declared in Table 6.2 or in the DecoderInit (section 6.2).

Fragment Reference: A reference within an instance of the ESG Data Model to an ESG XML Fragment. Note: Contrary to this a reference to an Encapsulated ESG Fragment refers from fragment management information to the storage location of the Encapsulated ESG Fragment.

Private Auxiliary Data: Data of which the format is not specified in this specification.

Service: In the context of this document a service is an offer from a service provider and has media content related to it.

3.2 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bitstream.

Name	*Definition*
bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in this specification. Bit strings are generally written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance. For convenience large strings are occasionally written in hexadecimal, in this case conversion to a binary in the conventional manner will yield the value of the bit string. Thus the left most hexadecimal digit is first and in each hexadecimal digit the most significant of the four bits is first.
uimsbf	Unsigned integer, most significant bit first.
vluimsbf8	Variable length code unsigned integer, most significant bit first. The size of vluimsbf8 is a multiple of one byte. The first bit (Ext) of each byte specifies if set to 1 that another byte is present for this vluimsbf8 code word. The unsigned integer is encoded by the concatenation of the seven least significant bits of each byte belonging to this vluimsbf8 code word An example for this type is shown in Figure 3.1.
vluimsbf5	Variable length code unsigned integer, most significant bit first. The first n bits (Ext) which are 1 except of the n-th bit which is 0, indicate that the integer is encoded by n times 4 bits. An example for this type is shown in Figure 3.2.

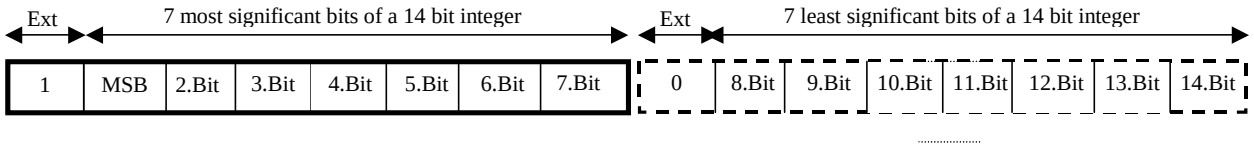


Figure 3.1 - Informative example for the vluimsbf8 data type

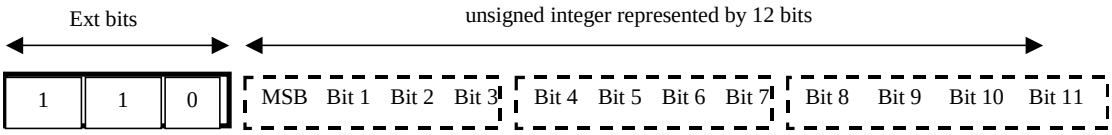


Figure 3.2 - Informative example for the vluimsbf5 data type

3.3 Functions

nextByteBoundary()

The function “nextByteBoundary()” reads and consumes bits from the binary stream until but not including the next byte-aligned position in the binary description stream.

ReservedBits

ReservedBits: a binary syntax element whose length is indicated in the syntax table. The value of each bit of this element shall be “1”. These bits may be used in the future for DVB defined extensions.

3.4 Abbreviations

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

3GPP	3G Partnership Project
ALC	Asynchronous Layered Coding
BiM	Binary format for Metadata
DVB-H	Digital Video Broadcast – Handheld
ESG	Electronic Service Guide
FDT	File Delivery Table (in FLUTE)
FEC	Forward Error Correction
FLUTE	File Delivery over Unidirectional Transport
HTTP	Hypertext Transfer Protocol
KMS	Key Management System
LCT	Layered Coding Transport
MBMS	Multimedia Broadcast/Multicast Service
MIME	Multipurpose Internet Mail Extensions
OMA	Open Mobile Alliance
RFC	Request for Comments
SDD	Session Delivery Descriptor
SDP	Session Description Protocol
TOI	Transport Object Identifier (in LCT)
TVA	TV-Anytime
URI	Uniform Resource Indentifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
XML	eXtensible Markup Language

4 Overview

4.1 ESG Processing Flow

Electronic Service Guide (ESG) contains information about the services available. Through the information in the ESG, the user can select the services and items he/she is interested in and find stored items on the terminal.

ESG operations takes place after the DVB-H receiver has been started and the terminal is synchronized to a particular transport stream carrying IPDC services.

Based on the ESG information rendered to a user through an ESG application, a specific service can be selected. The ESG also provides information which enables the terminal to connect to the related IP stream in the DVB-H transport stream.

The ESG operations be broken down in three main operations:

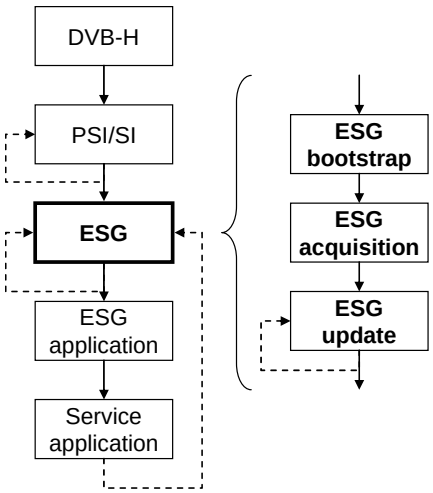


Figure 4.1

- ESG bootstrap: the operation through which the terminal knows which ESGs are available and how to acquire them.
- ESG acquisition: the operations through which the terminal gathers and processes the ESG information for the first time or after a long time without connecting.
- ESG update: the operation through which the terminal refreshes the ESG information stored in the terminal with the latest versions.

Note: even though in Figure 4.1 the steps are shown sequentially this does not mean that steps cannot be processed in parallel.

4.2 Service Discovery

Once the terminal has connected to a valid DVB-H transport stream carrying IPDC services on a particular IP Platform, it receives from the PSI/SI tables the location (PID) where the well-known IP address for the ESG bootstrap information of that IP Platform is located (see [24] for details). From the ESG bootstrap information, the terminal can figure out how many ESGs are available on that IP Platform, what is the relevant ESG to consume and the required information to configure the selected ESG session. Note that for starting on the selected ESG, the terminal needs to know the location of the related IP stream, through the PSI/SI tables.

Once the terminal located the IP stream of the selected ESG, it can initialize the file delivery session on the terminal and the ESG processing. Then the terminal can start to receive the ESG information.

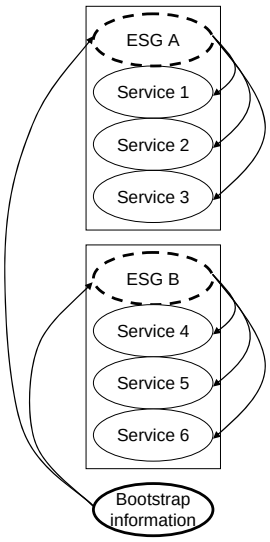


Figure 4.2

4.3 ESG layers

The ESG specification covers the description of the data model, the representation, the encapsulation and the transport:

- The Role of the ESG data model is to define a set of data structures which can be instantiated to describe available services. The ESG data model is defined based on XML Schema [17] and it is aimed at being consistent across all implementations of a system to ensure interoperability.
- The ESG Instance of the ESG Data Model is a consistent set of ESG data describing the available IP Datacast services.
- The ESG Representation supports fragmentation of the ESG Instance into ESG XML Fragments and allows an efficient representation of the ESG XML Fragments which minimizes the size of the metadata delivered to users. The partitioning of the ESG Instance into fragments for transportation is supported to enable separately updating parts of ESG data and for performance optimization.
- Encapsulation of ESG Fragments into containers aims at supporting the processing and transmission of ESG information in units of considerable size. The processing of ESG Fragments is supported by providing fragment management information which identifies already received fragments, updated fragments and new fragments.
- Transport is achieved by the use of FLUTE sessions to enable the optimal delivery of containers as files.

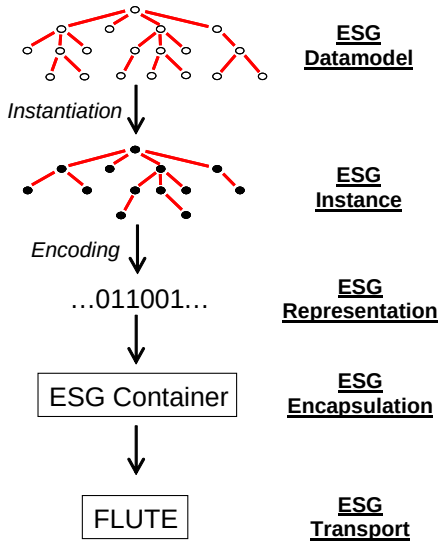


Figure 4.3

5 ESG Datamodel

5.1 Overview

In this section the datamodel of the ESG is specified. The datamodel is described by an XML Schema definition according to the XML Schema recommendation [17]. The ESG is subdivided into ESG Fragments, which can be instantiated as parts of the ESG. Figure 5.1 depicts the ESG Fragments specified in this section and the relations between them. The indicated cardinalities of the references correspond to the cardinalities specified in the XML Schema definition of the ESG datamodel. Beside the specification of each ESG Fragment the ESG Wrapper specifies how the ESG is compiled based on the ESG Fragments.

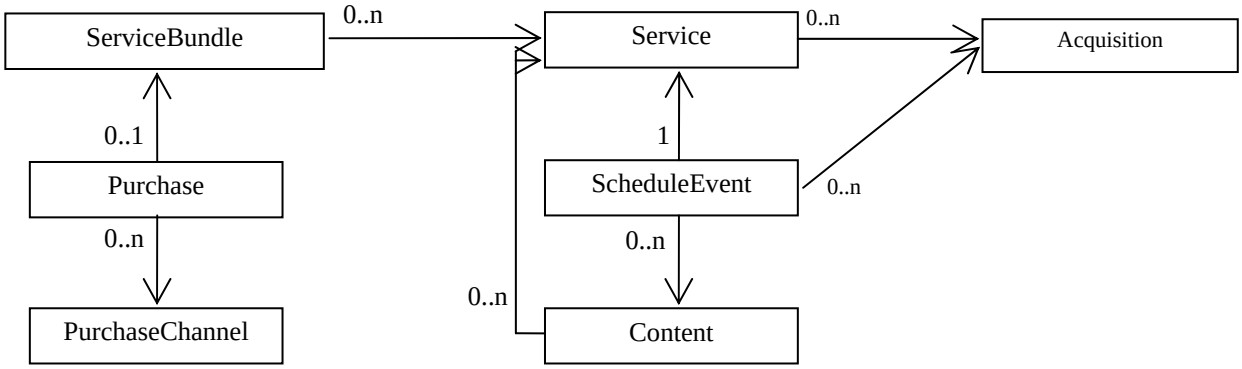


Figure 5.1: Block diagram of the specified ESG XML Fragments and the relations between them.

5.2 ESG Wrapper

In this section the ESG Main element is declared. Besides the type definitions of the Elements declared in the ESG Main element also the default ESG Main element is specified which should be assumed by the receiver if no ESG Main element is signaled.

5.2.1 ESG Namespace Declaration

In this section the target namespace urn:dvb:ipdc:esg:2005 is declared. This namespace contains the data types and the global elements defined respectively declared in this document. Also the imported namespaces and their namespace aliases used throughout this document are declared in this section.

```
<schema targetNamespace="urn:dvb:ipdc:esg:2005" xmlns:esg="urn:dvb:ipdc:esg:2005"
                xmlns:tva="urn:tva:metadata:2005" xmlns="http://www.w3.org/2001/XMLSchema"
                xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" elementFormDefault="qualified"
                attributeFormDefault="unqualified">
    <import namespace="http://www.w3.org/XML/1998/namespace" />
    <import namespace="urn:mpeg:mpeg7:schema:2001" />
    <import namespace="urn:tva:metadata:2005" />
```

5.2.2 ESG Main Element

In this section the ESG Main Element is specified. The ESG Main Element is the root element of the ESG. For the sender it is optional to signal the ESG Main Element. If the ESG Main Element is not signaled a default ESG Main Element should be assumed by the decoder as specified in section 5.2.4.

5.2.2.1 ESG Main Element Syntax

```
<element name="ESGMain" type="esg:ESGMainType"/>

<complexType name="ESGMainType">
    <sequence>
        <element name="CopyrightNotice" type="string" minOccurs="0"/>
        <element name="ClassificationSchemeTable" type="tva:ClassificationSchemeTableType"
                minOccurs="0"/>
        <element name="ESG" type="esg:ESGType" minOccurs="0"/>
    </sequence>
    <attribute ref="xml:lang" default="en" use="optional"/>
    <attribute name="publisher" type="string" use="optional"/>
    <attribute name="publicationTime" type="dateTime" use="optional"/>
    <attribute name="rightsOwner" type="string" use="optional"/>
</complexType>
```

5.2.2.2 ESG Main Element Semantics

Field	Semantics
ESGMain	The root element for a valid instance document of the ESG schema that provides a description of available services.
ESGMainType	Specifies the root element for an ESG instance document that provides a complete description of the ESG and that is valid with respect to the ESG Schema urn:dvb:ipdc:esg:2005.
CopyrightNotice	Specifies the copyright information for the ESG data.
ClassificationSchemeTable	Contains the classification schemes used by the various descriptions in the ESG document and their aliases (optional)
ESG	Contains the description of the ESG
xml:lang	Specifies the language of the description. The default value of this field is 'en' specifying that the description is in english.
publisher	Specifies the name of the publisher of the description
publicationTime	Specifies the time the metadata description was published.
rightsOwner	Specifies the entity that holds the rights of the description

5.2.3 ESG

5.2.3.1 ESG Syntax

```
<complexType name="ESGType">
    <sequence>
        <element name="ContentTable" type="esg:ContentTableType" minOccurs="0"/>
        <element name="ScheduleEventTable" type="esg:ScheduleEventTableType" minOccurs="0"/>
        <element name="ServiceTable"  type="esg:ServiceTableType" minOccurs="0"/>
        <element name="ServiceBundleTable"  type="esg:ServiceBundleTableType" minOccurs="0"/>
        <element name="PurchaseTable" type="esg:PurchaseTableType" minOccurs="0"/>
        <element name="PurchaseChannelTable" type="esg:PurchaseChannelTableType" minOccurs="0"/>
        <element name="AcquisitionTable" type="esg:AcquisitionTableType" minOccurs="0"/>
    </sequence>
</complexType>
```

```
<complexType name="ContentTableType">
    <sequence>
        <element name="Content" type="esg:ContentType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="ScheduleEventTableType">
    <sequence>
        <element name="ScheduleEvent" type="esg:ScheduleEventType" minOccurs="0"
                maxOccurs="unbounded"/>
```

```
    </sequence>
</complexType>

<complexType name="ServiceTableType">
    <sequence>
        <element name="Service" type="esg:ServiceType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="ServiceBundleTableType">
    <sequence>
        <element name="ServiceBundle" type="esg:ServiceBundleType" minOccurs="0"
                maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="PurchaseTableType">
    <sequence>
        <element name="Purchase" type="esg:PurchaseType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="PurchaseChannelTableType">
    <sequence>
        <element name="PurchaseChannel" type="esg:PurchaseChannelType" minOccurs="0"
                maxOccurs="unbounded"/>
    </sequence>
</complexType>

<complexType name="AcquisitionTableType">
    <sequence>
        <element name="Acquisition" type="esg:AcquisitionType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

5.2.3.2 ESG Semantics

Field	Semantics
ESGType	A complex type that aggregates the tables that contain ESG description metadata
ContentTable	Specifies the content table
ScheduleEventTable	Specifies the table of schedule events
ServiceTable	Specifies the service table
ServiceBundleTable	Specifies the service bundle table
PurchaseTable	Specifies the purchase table
PurchaseChannelTable	Specifies the purchase channel table
AcquisitionTable	Specifies the acquisition table

Field	Semantics
ContentTableType	Specifies a complex type that contains a table of content fragments
Content	Specifies a content fragment
ScheduleEventTableType	Specifies a complex type that contains a table of schedule event records
ScheduleEvent	Specifies schedule event record.
ServiceTableType	Specifies a complex type that contains a table of service fragments
Service	Specifies a service fragment
ServiceBundleTableType	Specifies a complex type that contains a table of service bundle records
ServiceBundle	Specifies a service bundle fragment
PurchaseTableType	Specifies a complex type that contains a table of purchase fragments
Purchase	Specifies a purchase fragment
PurchaseChannelTableType	Specifies a complex type that contains a table of purchase channel fragments
PurchaseChannel	Specfies a purchase channel fragment
AcquisitionTableType	Specifies a complex type that contains a table of acquisition fragments
Acquisition	Specifies an acquisition fragment

5.2.4 Default ESG Main Element Instantiation

The transmission of the ESGMain Fragment is not mandatory (see section 6.2.1). If the ESGMain fragment is not delivered to the decoder, the decoder is initialized with the default ESG Main fragment. The default ESGMain fragment is defined as follows:

```
<ESGMain xmlns="urn:dvb:ipdc:esg:2005">
    <ESG>
        <ContentTable/>
        <ScheduleEventTable/>
        <ServiceTable/>
        <ServiceBundleTable/>
        <PurchaseTable/>
        <PurchaseChannelTable/>
        <AcquisitionTable/>
    </ESG>
</ESGMain>
```

If the ESGMain fragment is delivered to the decoder, it shall be carried in theESG Init Container with the restrictions specified in section 8.1.1.

5.3 Basic ESG Datatypes

In this section Basic ESG Datatypes are defined which are used in declarations of different ESG XML Fragments.

5.3.1 ESG XML Fragment Reference

The ESGIDRefType data type is defined to specify references between ESG XML fragments.

5.3.1.1 ESG XML Fragment Reference Syntax

```
<complexType name="ESGIDRefType">
    <attribute name="IDRef" type="anyURI"/>
</complexType>

<complexType name="AcquisitionRefType">
    <complexContent>
        <extension base="esg:ESGIDRefType">
            <sequence>
                <element name="Label" type="mpeg7:TextualType" minOccurs="0"
            maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="ServiceRefType">
    <complexContent>
        <extension base="esg:ESGIDRefType">
            <attribute name="serviceNumber" type="unsignedShort"/>
        </extension>
    </complexContent>
</complexType>
```

5.3.1.2 ESG XML Fragment Reference Semantics

Field	Semantics
IDRef	Specifies the reference to an ESG XML Fragment.
Label	Specifies the characteristic of the referenced Acquisition Fragment in the scope of the referenced Acquisition Fragments of the service. For instance the Label can specify that the referenced Acquisition Fragment describes the “high resolution video” compared to other Acquisition Fragments.
serviceNumber	The logical number of the service in the context of the described service bundle. This number can be displayed to the user alongside the service name when a certain service bundle is selected. The user may enter the number on a numeric keypad to select the service. The terminal may also use this number to order the services of a service bundle displayed to the user on-screen.

5.3.2 Related Material

The RelatedMaterialType is used in a number of fragment types to enable the signalling of media assets that are related to the ESG XML Fragment (i.e. Content, Service or ServiceBundle Fragment).

5.3.2.1 Related Material Syntax

```
<complexType name="RelatedMaterialType">
```

```
    <sequence>
      <element name="HowRelated" type="tva:ControlledTermType" minOccurs="0"/>
      <element name="MediaLocator" type="mpeg7:MediaLocatorType"/>
      <element name="PromotionalText" type="mpeg7:TextualType" minOccurs="0"
        maxOccurs="unbounded"/>
      <element name="PromotionalMedia" type="mpeg7:TitleMediaType" minOccurs="0"
        maxOccurs="unbounded"/>
    </sequence>
  </complexType>
```

5.3.2.2 Related Material Semantics

Field	Semantics
HowRelated	Specifies the nature of the relationship between the described fragment (Content, Service or ServiceBundle Fragment) and the related media assets.
MediaLocator	Specifies the location of the media asset. Defined as an MPEG-7 datatype, MediaLocatorType (see clause 6.5.2 of ISO/IEC 15938-5 [4] for a detailed description).
PromotionalText	Specifies promotional information about the link, which can be used as an additional attractor (e.g. record "Pride and Prejudice" series).
PromotionalMedia	Specifies non-text promotional information such as a logo.

5.3.3 ProviderType

The ProviderType specifies information about a provider (e.g. Service Provider) and specifies a unique identification of the provider.

5.3.3.1 ProviderType Syntax

```
  <complexType name="ProviderType">
    <sequence>
      <element name="ProviderURI" type="anyURI" minOccurs="0"/>
      <element name="ProviderName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
      <element name="ProviderLogo" type="mpeg7:TitleMediaType" minOccurs="0"/>
      <element name="ProviderInformationURL" type="anyURI" minOccurs="0"/>
    </sequence>
  </complexType>
```

5.3.3.2 ProviderType Semantics

Field	Semantics
ProviderURI	A URI that uniquely identifies the provider.
ProviderName	The textual name of the provider, possibly in different languages.
ProviderLogo	Specifies a graphical representation of the provider promotional logo.
ProviderInformationURL	Specifies a URL of more detailed information about the provider.

5.3.4 Private Data

The datatype defined in this section provides an extension point to specify private data. Neither the syntax nor the semantics of this private data are defined in this document.

5.3.4.1 Private Data Syntax

```
<complexType name="PrivateDataType" abstract="true"/>
```

5.4 Service Fragment

The service fragment describes an IPDC service, for instance a traditional TV channel or a service supplying ring tones.

5.4.1 Service Fragment Syntax

```
<complexType name="ServiceType">
    <sequence>
        <element name="ServiceName" type="tva:ServiceInformationNameType"
            maxOccurs="unbounded"/>
        <element name="ServiceNumber" type="unsignedShort" minOccurs="0"/>
        <element name="ServiceLogo" type="mpeg7:TitleMediaType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="ServiceDescription" type="tva:SynopsisType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="ServiceGenre" type="tva:GenreType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="ServiceType" type="tva:ControlledTermType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="ParentalGuidance" type="mpeg7:ParentalGuidanceType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="ServiceLanguage" type="language" minOccurs="0" />
        <element name="ServiceProvider" type="esg:ProviderType" minOccurs="0" />
        <element name="AcquisitionRef" type="esg:AcquisitionRefType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="RelatedMaterial" type="esg:RelatedMaterialType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="PrivateData" type="esg:PrivateDataType" minOccurs="0"
            maxOccurs="unbounded" />
    </sequence>
    <attribute name="serviceID" type="anyURI" use="required"/>
    <attribute name="freeToAir" type="boolean" use="optional"/>
    <attribute name="clearToAir" type="boolean" use="optional"/>
</complexType>
```

5.4.2 Service Fragment Semantics

Field	Semantics
ServiceName	Specifies the name given to the described service. The ServiceName may be specified in different languages.
ServiceNumber	Specifies a number assigned to the service by the ESG provider, which is unique within the Current ESG XML Document. This number can be displayed to the user alongside the service name, and the user may enter the number on a numeric keypad to select the service. The terminal may also use this number to order the services displayed to the user on-screen.
ServiceLogo	Specifies the reference to media representing the title of the described Service. The media can be of type image, audio or video. Note: This can take the form of a reference for instance to an image or a sound available externally to the metadata, or the media data can be provided inline.
ServiceDescription	Specifies the textual description of the described service in a specified language.
ServiceGenre	Specifies a genre that characterises the main genre of the media content available from the described service.
ServiceType	Specifies the characteristic of the Service e.g. if it is a download service, a streaming service or a combination of both. This field might be overwritten by the ContentType field in associated Content Fragments.
ParentalGuidance	Specifies the parental rating of the described services.
ServiceLanguage	Specifies the primary spoken language used within the content available from this service. Note: this attribute eases the processing of the language information of one service by duplicating information which can be compiled from language information in ESG XML Fragments related to the Service Fragment.
ServiceProvider	Specifies the service provider offering the described service (see section 5.3.3). Examples: for a TV channel, usually the service provider is the TV channel itself. For a daily newspaper downloading service, the service provider may be the Newspaper editor.
AcquisitionRef	Specifies the acquisitionID of the Acquisition Fragment which specifies generic information for the acquisition of this service. Note: This generic Acquisition Fragment referenced in the AcquisitionRef element of the Service Fragment might be overwritten by more specific Acquisition Fragments referenced in a Schedule Event Fragment (see section 5.7).
RelatedMaterial	Specifies reference to material related to the described service. For instance, for a sport TV channel, a RelatedMaterial may be an URL of a web site which provides sports information.
PrivateData	Generic element instantiated to add private data. This is used e.g. for fields of specific services such as a rights object service.
serviceID	Unique identifier for the described Service. This identifier may just have local scope i.e. within the network from which the Service Fragment was acquired. Alternatively provided there is agreement between a number of network operators or the service provider, the identifier may have global scope, where a common identifier is used for identifying a service no matter which network the service description is acquired from. For example broadcaster “Foo” may mandate that the identifier be, "foo.com/foo1" for a service description, describing Channel 1 of broadcaster Foo.
freeToAir	Set to ‘true’ the attribute specifies that the content associated to the service is available for

	free. Set to ‘false’ the attribute specifies that the content associated to the service is not available for free. If the attribute is not specified this information has to be derived based on the related Purchase Fragments.
clearToAir	Set to ‘true’ the attribute specifies that the content associated to the service is not scrambled. Set to ‘false’ the attribute specifies that the content associated to the service is scrambled.

5.5 Service Bundle Fragment

The Service Bundle Fragment specifies a bundle of services. A bundle is understood to be a grouping of items offered to the user in the form of services. This grouping can be used to bind certain purchase information to the group or to add information to the items in context of the group (e.g. service number).

The bundle notion is primarily commercial and gives the ability to group services together forming packages to customers i.e. sport service bundle, cinema service bundle.

5.5.1 Service Bundle Fragment Syntax

```
<complexType name="ServiceBundleType">
    <sequence>
        <element name="ServiceBundleName" type="mpeg7:TextualType" maxOccurs="unbounded"/>
        <element name="ServiceBundleProvider" type="esg:ProviderType" minOccurs="0"/>
        <element name="ServiceBundleMediaTitle" type="mpeg7:TitleMediaType" minOccurs="0"/>
        <element name="ServiceBundleDescription" type="mpeg7:TextualType" minOccurs="0"
                maxOccurs="unbounded"/>
        <element name="ServiceBundleGenre" type="tva:GenreType" minOccurs="0"
                maxOccurs="unbounded"/>
        <element name="ServiceRef" type="esg:ServiceRefType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="ParentalGuidance" type="mpeg7:ParentalGuidanceType" minOccurs="0"
                maxOccurs="unbounded"/>
        <element name="RelatedMaterial" type="esg:RelatedMaterialType" minOccurs="0"
                maxOccurs="unbounded"/>
    </sequence>
    <attribute name="serviceBundleID" type="anyURI" use="required"/>
</complexType>
```

5.5.2 Service Bundle Fragment Semantics

Field	Semantics
ServiceBundleName	The name of the service bundle in text form. The name may be specified in different languages.
ServiceBundleProvider	Specifies the provider of the service bundle.
ServiceBundleMediaTitle	Specifies the reference to media representing the title of the described service bundle. The media can be of type image, audio or video. Note: This can take the form of a reference for instance to an image or a sound available externally to the metadata, or the media data can be provided inline.
ServiceBundleDescription	Specifies the textual description of the service bundle in a specified language.
ServiceBundleGenre	Specifies a genre that characterises the genre of the medias available from the described service bundle.
ServiceRef	Specifies the services contained in the service bundle. Note: To enable future extensions of the data model the reference is specified as an optional field. However in this version of the data model the field is mandatory.
ParentalGuidance	Specifies the parental rating of the described services.
RelatedMaterial	Specifies reference to material related to the described service bundle. For instance, for a TV channel bouquet, a RelatedMaterial may be a web site URL where additional information about the bouquet can be found
serviceBundleID	Specifies a unique identifier of the instantiated Service Bundle Fragment. For the scope of uniqueness see the semantics of serviceID in section 5.4.2

5.6 Content Fragment

A Content Fragment contains the metadata that describes the content independently of any particular delivery instantiation of that content. All types of contents (e.g. A/V, text, images,) are described using the same data type.

5.6.1 Content Fragment Syntax

```
<complexType name="ContentType">
    <sequence>
        <element name="Title" type="mpeg7:TitleType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="MediaTitle" type="mpeg7:TitleMediaType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="ServiceRef" type="esg:ESGIDRefType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="Synopsis" type="tva:SynopsisType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="Keyword" type="tva:KeywordType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="Genre" type="tva:GenreType" minOccurs="0" maxOccurs="unbounded"/>
        <element name="ContentType" type="tva:ControlledTermType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="ParentalGuidance" type="mpeg7:ParentalGuidanceType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="Language" type="mpeg7:ExtendedLanguageType" minOccurs="0"
            maxOccurs="unbounded"/>
```

```
        <element name="CaptionLanguage" type="tva:CaptionLanguageType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="SignLanguage" type="tva:SignLanguageType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="CreditsList" type="tva:CreditsListType" minOccurs="0"/>
        <element name="RelatedMaterial" type="esg:RelatedMaterialType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="Duration" type="duration" minOccurs="0"/>
        <element name="PrivateData" type="esg:PrivateDataType" minOccurs="0"
            maxOccurs="unbounded" />
    </sequence>
    <attribute name="contentID" type="anyURI" use="required"/>
</complexType>
```

5.6.2 Content Fragment Semantics

Field	Semantics
Title	Specifies the title assigned to the described content. The title may be associated to a language.
MediaTitle	Specifies the reference to media representing the title of the described content. Note: This can take the form of a reference to an image or a sound available externally to the metadata, or the media data can be provided inline.
ServiceRef	Specifies the ServiceId of the Service Fragment to which the described content is associated to.
Synopsis	Specifies a short, textual summary of the described content. The language of the description is specified by the lang attribute
Keyword	Specifies Keywords characterizing the described content.
Genre	Specifies a genre that characterizes the genre of the described content.
ContentType	Specifies the characteristic of the content e.g. if the content is download content, streaming content or a combination of both. If this element is present it overwrites the information specified in the field ServiceType in the ServiceFragment. If it is not present, the ContentType is inherited from the field ServiceType in the Service Fragment this Content belongs to.
ParentalGuidance	Specifies the parental rating of the described content. This element can be associated to a region.
Language	Specifies a language of the described content. Note: The language specification in the Content Fragment is seen to be user attracting data which can be displayed even if the related Acquisition Fragment is not available. However it is understood that the language specification in the Acquisition Fragment is more precise with respect to the components of the content.
CaptionLanguage	Specifies a caption language of the described content. Note: The caption language specification in the Content Fragment is seen to be user attracting data which can be displayed even if the related Acquisition Fragment is not available. However it is understood that the language specification in the Acquisition Fragment is more precise with respect to the components of the content.
SignLanguage	Specifies a sign language of the described content. Note: The sign language specification in the Content Fragment is seen to be user attracting data which can be displayed even if the related Acquisition Fragment is not available. However it is understood that the language specification in the Acquisition Fragment is more precise with respect to the components of the content.
CreditsList	The list of credits (e.g. actors, directors, etc.) for the content.
RelatedMaterial	Specifies reference to material related to the described content. For instance, a related material associated to an AV program may be a web site URL that provides more information about the AV program.
Duration	Specifies the duration of the described content. Note: The duration specifies in the Content Fragment differs from the duration which can be derived from the Schedule Event Fragment in that the first describes the duration of the content and the latter the time period in which it is available..
PrivateData	Generic element instantiated to add private data. This is used e.g. for fields of specific content such as a rights objects.
contentID	Specifies a unique Identifier of the instantiated Content Fragment. For the scope of

	uniqueness see the semantics of serviceID in section 5.4.2

5.7 Schedule Event Fragment

The Schedule Event Fragment specifies the broadcast time of a scheduled item which is a content item of a service. The specified broadcast time is the time to be displayed to the end user.

5.7.1 Schedule Event Fragment Syntax

```
<complexType name="ScheduleEventType">
    <sequence>
        <element name="PublishedStartTime" type="dateTime" minOccurs="0"/>
        <element name="PublishedEndTime" type="dateTime" minOccurs="0"/>
        <element name="ServiceRef" type="esg:ESGIDRefType"/>
        <sequence maxOccurs="unbounded">
            <element name="ContentFragmentRef" type="esg:ESGIDRefType" minOccurs="0"/>
            <sequence maxOccurs="unbounded">
                <element name="AcquisitionRef" type="esg:AcquisitionRefType" minOccurs="0"/>
                <element name="ContentLocation" type="anyURI" minOccurs="0"/>
            </sequence>
        </sequence>
    </sequence>
    <attribute name="live" type="boolean" use="optional"/>
    <attribute name="repeat" type="boolean" use="optional"/>
    <attribute name="freeToAir" type="boolean" use="optional"/>
    <attribute name="clearToAir" type="boolean" use="optional"/>
    <attribute name="scheduleId" type="anyURI" use="optional"/>
</complexType>
```

5.7.2 Schedule Event Fragment Semantics

Field	Semantics
PublishedStartTime	Specifies the start time of the scheduled item.
PublishedEndTime	Specifies the end time of the scheduled item.
ServiceRef	Specifies the ServiceId of the Service Fragment this Schedule Event is assigned to.
ContentFragmentRef	This field is used to reference a Content fragment, which describes the Content available during this Schedule event. Following this element a number of AcquisitionRef, and optional ContentLocation elements may be instantiated. These elements are used to declare the format and delivery parameters for the Content described by the referenced Content fragment.
AcquisitionRef	This field signals the Acquisition fragment that describes the particular format and acquisition parameters for the content item referenced by the previous ContentFragmentRef element. The following ContentLocation element announces the Content location for this particular instance of a Content Item.
ContentLocation	This field signals the URI of the Content location, that is used to identify the content file within the FLUTE session described in the Acquisition Fragment. Note: This element is only applicable for FLUTE sessions which carry more than one file during the lifetime of the event.
live	This flag is set to "true" to indicate that the schedule event is a live broadcast. This flag is set to “false” to indicate that the schedule event is a broadcast from recorded material.
repeat	This flag is set to "true" to indicate that the schedule event is a repeat of a previous broadcast.
freeToAir	Set to ‘true’ the attribute specifies that the content associated to the scheduled service is available for free. Set to ‘false’ the attribute specifies that the content associated to the scheduled service is not available for free. The freeToAir value in the Schedule Event fragment completely overrides its counterpart in the associated Service Fragment. When the freeToAir attribute is not present in the Schedule Event fragment, the freeToAir attribute from the associated Service Fragment must be taken into account.
clearToAir	Set to ‘true’ the attribute specifies that the content associated to scheduled service is not scrambled. Set to ‘false’ the attribute specifies that the content associated to schedule service is scrambled. The clearToAir value in the Schedule Event fragment completely overrides its counterpart in the associated Service Fragment. When the clearToAir attribute is not present in the Schedule Event Fragment, this attribute from the associated Service Fragment must be taken into account.
scheduleID	Specifies a unique Identifier of the instantiated Schedule Fragment. For the scope of uniqueness see the semantics of serviceID in section 5.4.2

5.8 Purchase Fragment

The Purchase Fragment specifies the purchase information of a service which can be displayed to the end user for information purposes. The Purchase Fragment also contains information required to make the actual purchase.

5.8.1 Purchase Fragment Syntax

```
<complexType name="PurchaseType">
    <sequence>
        <element name="ServiceBundleRef" type="esg:ESGIDRefType" minOccurs="0"/>
```

```
        <element name="Price" maxOccurs="unbounded">
            <complexType>
            <simpleContent>
                <extension base="float">
                    <attribute name="currency" type="esg:currencyCodeType" use="required"/>
                </extension>
            </simpleContent>
            </complexType>
        </element>
        <element name="UsageConstraints" type="esg:UsageConstraintsType" minOccurs="0"
            maxOccurs="unbounded">
        <element name="Description" type="mpeg7:TextualType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="PurchaseRequest" type="esg:PurchaseRequestType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="MediaTitle" type="mpeg7:TitleMediaType" minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
    <attribute name="start" type="dateTime" use="optional"/>
    <attribute name="end" type="dateTime" use="optional"/>
    <attribute name="purchaseId" type="anyURI" use="optional"/>
</complexType>

<simpleType name="currencyCodeType">
    <restriction base="string">
        <pattern value="[a-zA-Z]{3}"/>
    </restriction>
</simpleType>

<complexType name="UsageConstraintsType">
    <sequence>
        <element name="PurchaseType" type="tva:ControlledTermType" minOccurs="0"/>
        <element name="QuantityUnit" type="tva:ControlledTermType" minOccurs="0"/>
        <element name="QuantityRange" minOccurs="0">
            <complexType>
            <attribute name="min" type="unsignedInt" use="optional"/>
            <attribute name="max" type="unsignedInt" use="optional"/>
            </complexType>
        </element>
    </sequence>
</complexType>

<complexType name="PurchaseRequestType">
    <sequence>
        <element name="DRMSystem" type="anyURI" />
        <element name="PurchaseData" type="esg:PurchaseDataBaseType" minOccurs="0"/>
        <element name="PurchaseChannelIDRef" type="esg:ESGIDRefType" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="PurchaseDataBaseType" abstract="true"/>

<complexType name="PurchaseDataType">
    <complexContent>
        <extension base="esg:PurchaseDataBaseType">
            <sequence>
                <element name="Data" type="string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

5.8.2 Purchase Fragment Semantics

Field	Semantics
ServiceBundleRef	Refers to an instantiated ServiceBundle Fragment which can be purchased according to the information provided within the Purchase Fragment. Note: To enable future extensions of the data model the reference is specified as an optional field. However in this version of the data model the field is assumed to be mandatory.
Price	Specifies the price of an offer described by the instantiated Purchase Fragment. If several Price elements are instantiated these shall describe the price in different currencies. .Note: The price information is non contractual information.
UsageConstraints	Describes the usage that is permitted when purchasing the rights associated with this Purchase Fragment. Multiple occurrences of the UsageConstraints element shall be treated as having an AND relationship. This allows for example business rules such as: maximum of 2 plays within the next 3 days.
PurchaseType	Specifies the type of purchase e.g. if the purchase legitimates to consume a service for a restricted period of time, a dedicated number of times or other types of purchase.
QuantityUnit	Specifies a quantisation of the purchased item e.g. hour, day, number of plays.
QuantityRange	Specifies the range of the purchased item which is part of purchase according to the quantisation specified in QuantityUnit. For example the maximum number of plays can be specified or the number of days one is legitimated to consume a service. If the offer varies depending on the content also the quantity range of the legitimation can be specified for example the legitimation to consume content from a min of four to a maximum of seven days.
Description	Specifies text which contains promotional information and which can not be described by Price and UsageConstraints elements.
PurchaseRequest	Specifies the Request to initiate the purchase.
DRMSystem	Specifies a URI which identifies the DRM system for which the PurchaseRequests associated PurchaseData is valid for. This field may be used to select an appropriate purchase mechanism based on the DRM systems supported within the terminal. The CA_System_ID registered in DVB shall be signalled by prefixing the CA_System_ID with “urn:dvb:casystemid:”. An example of the field value is “urn:dvb:casystemid:709”.
PurchaseData	Specifies a string in the Data element which is used to hold information necessary to perform the purchase. The interpretation of the data contained within this field is dependent on the DRM System. This field may for example contain a PurchaseId which indicates to a Purchase Server which option the user has chosen to purchase.
PurchaseChannelIDRef	Specifies a reference to a PurchaseChannel fragment, which provides additional information about the operator who offers the purchase option. Additionally parameters of the Purchase Channel can be signaled which are specific to the purchase channel.
MediaTitle	Specifies the reference to media representing the title of the service bundle to which purchase information is attached. The specified MediaTitle replaces the MediaTitle of the ServiceBundle if the Purchase Fragment is applicable. The media can be of type image, audio or video. Note: This can take the form of a reference for instance to an image or a sound available externally to the metadata, or the media data can be provided inline.
start	Specifies the time from which the purchase information is valid.
end	Specifies the time until which the purchase information is valid.

purchaseID	Specifies a unique Identifier of the instantiated Purchase Fragment. For the scope of uniqueness see the semantics of serviceID in section 5.4.2

5.9 Purchase Channel Fragment

A purchase channel is the interface through which the terminal or user can interact with a purchase system. The purchase channel is described in the Purchase Channel Fragment by the name, the logo, the description, the contact information of the purchase channel operator and the parameters specific for the purchase system. These parameters are not specified in this document but if required should be specified by the purchase system.

In some deployments of a Mobile Broadcast platform, there may be multiple purchase channels. A certain end-user might have a “preferred” purchase channel (e.g. his/her mobile operator) to which all purchase requests should be directed (the preferred purchase channel may even be the only channel that an end-user can use).

5.9.1 Purchase Channel Fragment Syntax

```
<complexType name="PurchaseChannelType">
    <sequence>
        <element name="Name" type="mpeg7:TextualType" maxOccurs="unbounded"/>
        <element name="Description" type="mpeg7:TextualType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="PortalURL" type="anyURI" minOccurs="0"/>
        <element name="ContactInfo" type="anyURI" minOccurs="0" maxOccurs="unbounded"/>
        <element name="MediaTitle" type="mpeg7:TitleMediaType" minOccurs="0"
            maxOccurs="unbounded"/>
        <element name="PrivateData"   type="esg:PrivateDataType" minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
    <attribute name="purchaseChannelID" type="anyURI" use="required"/>
</complexType>
```

5.9.2 Purchase Channel Fragment Semantics

Field	Semantics
Name	Name of the Purchase Channel. Several names in different languages can be specified by instantiating the xml:lang attribute.
Description	Specifies a Description of the Purchase Channel which can be displayed to the user.
PortalURL	Specifies a reference to a web site of the purchase channel.
ContactInfo	Contact information like phone number to contact the Purchase Channel Operator.
MediaTitle	Specifies the reference to media representing the logo of the described Purchase Channel. The media can be of type image, audio or video. Note: This can take the form of a reference for instance to an image or a sound available externally to the metadata, or the media data can be provided inline.
PrivateData	Generic element instantiated to add private data. This is used e.g. for specific parameters of the Purchase Channel.
purchaseChannelID	Specifies a unique identifier for the described Purchase Channel. This identifier may just have local scope i.e. within the network from which the Purchase Channel Fragment was acquired. Alternatively provided there is agreement between a number of network operators or the Purchase Channel provider, the identifier may have global scope, where a common identifier is used for identifying a Purchase Channel no matter which network the Purchase Channel Fragment is acquired from.

5.10 Acquisition Fragment

The Acquisition Fragment specifies information to access a service or content. The Acquisition Fragment contains information displayed to the end user such as ComponentsCharacteristic, information relevant for the ESG application such as contentType and SessionDescription used by the media player for initialization.

5.10.1 Referencing Described Content

The Acquisition Fragment mandates the instantiation of the SessionDescription element to enable the identification of a session carrying content described by the ESG XML Fragments. The session is described by an SDP File.

At a given time, a given Service Fragment may be linked to any number of Acquisition Fragments (so-called Service-related Acquisition Fragments). At the same time, one Schedule Event associated to the given Service may also be linked to any number of Acquisition Fragments (so-called Schedule-related Acquisition Fragments).

Note that, in case a given session is identified in both Service-related Acquisition Fragment and Schedule-related Acquisition fragment, the terminal is not expected to reload the SDP file associated to the given session.

For instance Figure 5.2 and 5.3 provide an illustration of two use cases.

In Figure 5.2 the identification of sessions transporting described content is illustrated based on a use case of a TV service with two temporarily audio tracks. Most of the time the depicted TV service has a video stream and an English audio stream. The respective sessions are described by SDP file “SDP1” which is referred to by Acquisition Fragment “Acquisition1”. However during “Event 2” in the TV service a French audio stream is also available. All streams available in this “Event 2” are described by SDP file “SDP2” which is referred to by Acquisition Fragment “Acquisition2”.

Also in the references to the Acquisition fragments a distinction exists. The Acquisition Fragment “Acquisition1” is referenced as generic Acquisition Fragment directly from the Service Fragment. On the other side the Acquisition

Fragment “Acquisition2” is referenced as a specific Acquisition Fragment from the Schedule Event Fragment “Schedule Event 2”.

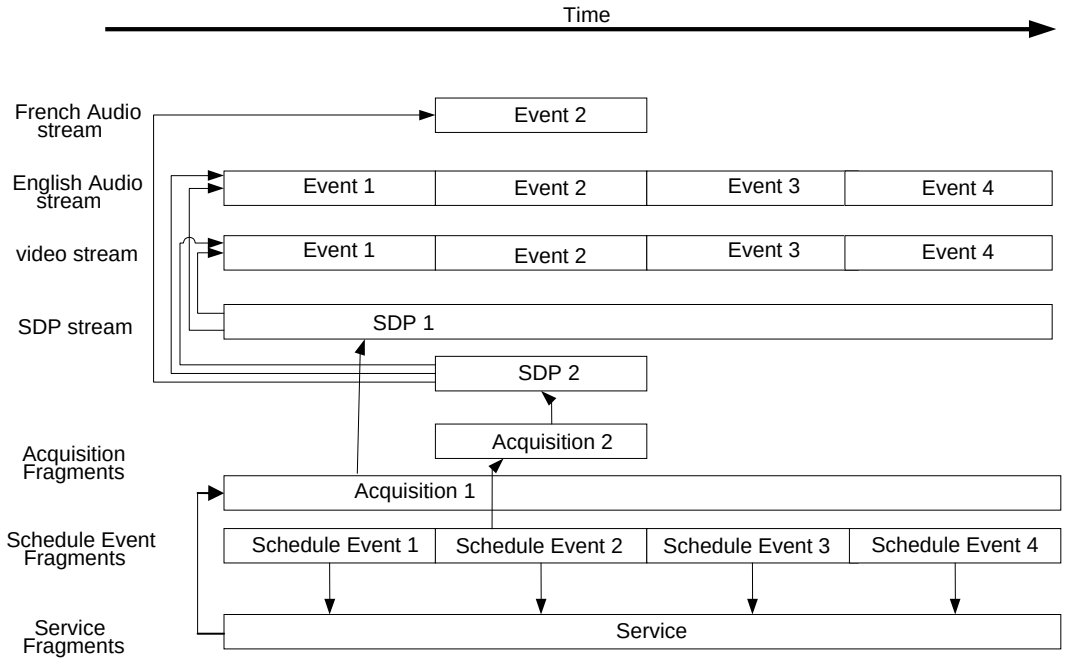


Figure 5.2: Relationship between described content streams, SDP files, Acquisition Fragments and other ESG XML Fragments.

The depicted identification of described content allows the user to tune to the TV Service in a continuous mode consuming the video and English audio track. However when the Schedule Event “Event 2” occurs the terminal may signal to the user that an additional French audio track is now available.

In Figure 5.3 the identification of sessions transporting described content is illustrated based on a use case of a TV service which regularly provides two audio tracks (French and English). The depicted TV service has a video stream and English audio stream which is expected to be the audio track that is always delivered. The respective session is described by the SDP file "SDP" which is referred to by Acquisition Fragment Acquisition1. The Acquisition Fragment Acquisition1 describes only the English audio component. In the "SDP" file, both English and French audio streams are described as audio media (in that case the attribute a=lang may be used in accordance to RFC 2327 SDP-Session Description Protocol). During event 2 and 3 the TV service English and French audio tracks are available. The Acquisition Fragment “Acquisition2” describes the English and French audio tracks and refers to the SDP file "SDP". “Acquisition2” overrides the information from “Acquisition1”. This way, the terminal and the users may be indicated that the French language is available during event 2 and 3 in addition to the English language. However the terminal player doesn't have to load a new SDP file.

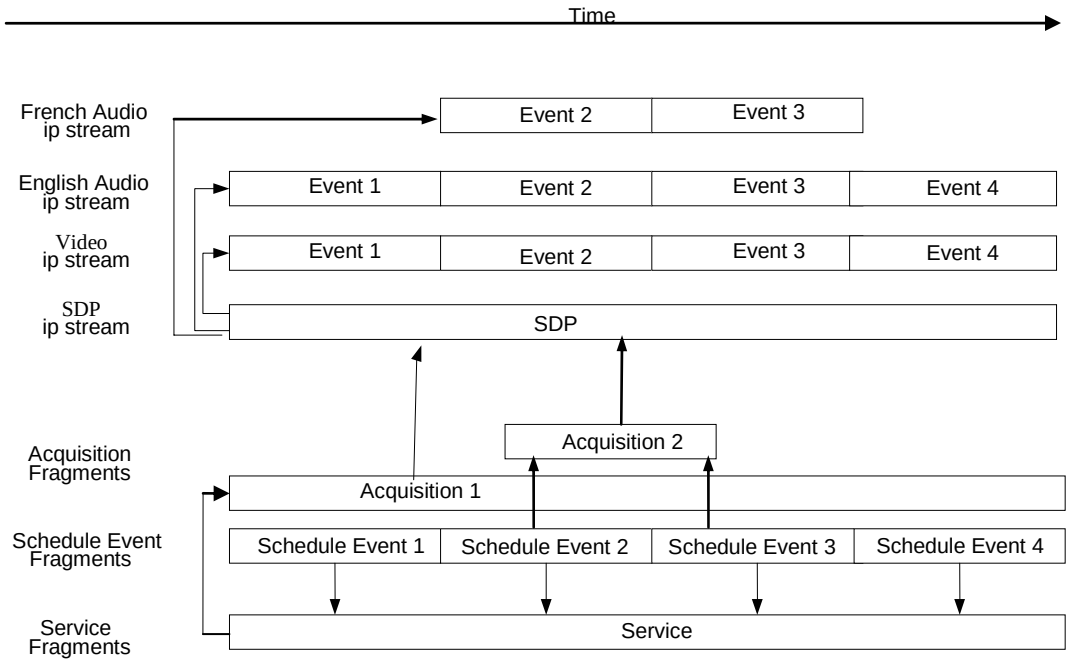


Figure 5.3: Relationship between described content streams, SDP files, Acquisition Fragments and other ESG XML Fragments in the second use case

5.10.2 Acquisition Fragment Syntax

```
<complexType name="AcquisitionType">
    <sequence>
        <element name="ComponentDescription" type="esg:ComponentDescriptionType"
            maxOccurs="unbounded"/>
        <element name="ZappingSupport" type="esg:ZappingSupportType" minOccurs="0"/>
        <element name="KeyStream" type="esg:KeyStreamBaseType" minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
    <attribute name="contentMimeType" type="mpeg7:mimeType" use="required"/>
    <attribute name="acquisitionID" type="anyURI" use="required"/>
</complexType>

<complexType name="ComponentDescriptionType">
    <sequence>
        <element name="ComponentCharacteristic" type="esg:ComponentCharacteristicType"
            minOccurs="0" maxOccurs="unbounded"/>
        <element name="SessionDescription" type="esg:SessionDescriptionBaseType"/>
    </sequence>
</complexType>
```

5.10.3 Acquisition Fragment Semantics

Field	Semantics
ComponentDescription	Describes a component of a service with respect to the characteristic of the component and the session in which the component is available. Note that an acquisition fragment can contain multiple ComponentDescriptions. For instance an indicated application can consume streaming components such as audio and video as well as download components such as interactive applications. In this case the streaming components are described by one ComponentDescription with a SessionDescription and two ComponentCharacteristic fields, one for audio and another one for video. A second ComponentDescription is instantiated for the download component describing the download session and the characteristic of the download components.
ComponentCharacteristic	Specifies the description of the component characteristic specific to the instance accessible by the instantiated acquisition information.
SessionDescription	Contains inlined SDP file which either directly describes the content session or the session carrying an SDP file that describes the content session (see section 5.10.4). The transport of the SDP files in the latter case is specified in section 8.5.
ZappingSupport	If specified indicates that zapping support of a specified Type is available for the acquisition described in the Acquisition Fragment. The field provides a specification of the type of zapping support and a reference to the session description (see section 5.10.5).
KeyStream	Signals all available key streams for a given Acquisition Fragment required for decryption as described in section 5.10.6. The mapping of key streams to media streams is described in [5].
acquisitionID	Specifies a unique identifier of the instantiated Acquisition Fragment. For the scope of uniqueness see the semantics of serviceID in section 5.4.2.
contentMimeType	Specifies the content type from which the terminal can determine the consuming application of the service.

5.10.4 Session Description

The session description files contain information that the terminal needs in order to be able to receive and consume the content of a service. Every session description file relates to a service or a schedule event of a service. A session description file contains application configuration information. The ESG XML Fragments include the information that helps the user to choose what service sessions he/she is interested in.

There are two methods of conveying the content referencing SDP file:

- **Inline**, where the session description file is inlined in the Acquisition fragment as element content. This is useful for situations where the contents of the SDP file are fixed for the time the Acquisition Fragment is signaled and known at the time the Acquisition Fragment is generated. In this case the SessionDescriptionType below contains an SDP element.

- **Out of Band**, where the session description files are delivered independently of the Acquisition Fragment. This is useful for situations where, either the contents of the SDP file are not necessarily fixed for the time the Acquisition Fragment is signaled, or in cases where the Acquisition Fragment is signaled before the SDP file is available. In this case the SessionDescriptionType below contains an SDPRef element.

Note that the SDPRefType comprises

- an SDPURI element, giving the URI of the content referencing SDP file. The URI is e.g. specified in the Content-Location attribute within the FDT of the FLUTE session.

- an SDPStream element, which inlines an SDP file. The SDP file tells the terminal how to join the session to receive the content referencing SDP file.

The AssociatedProcedureDescription file is always delivered out-of-band (i.e. not in the ESG Fragment Stream), either in SDP delivery session or as an object of the service content. In order to be able to identify this configuration file, a URI of the file can be instantiated within the session description datatypes. In case of the existence of an SDP delivery session, the AssociatedProcedureDescription file may be delivered over that session, which is referenced by the inlined SDP file. The AssociatedDeliveryProcedure may be delivered as component of the service content. This is not specified here. However, the URI SHALL be used to address the AssociatedDeliveryProcedure configuration file.

5.10.4.1 Session Description Syntax

```
    <complexType name="SessionDescriptionBaseType" abstract="true" />

    <complexType name="InlinedSDPType">
        <complexContent>
            <extension base="esg:SessionDescriptionBaseType">
                <sequence>
                    <element name="SDP" type="esg:SDPType"/>
                    <element name="AssociatedDeliveryProcedure" type="anyURI" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

    <complexType name="SDPRefType">
        <complexContent>
            <extension base="esg:SessionDescriptionBaseType">
                <sequence>
                    <element name="SDPStream" type="esg:SDPType"/>
                    <element name="SDPURI" type="anyURI" />
                    <element name="AssociatedDeliveryProcedure" type="anyURI" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>

<simpleType name= "SDPType">
    <!--- Note: the InlinedSDP below must be embedded in a CDATA section -->
    <restriction base="string"/>
</simpleType>
```

5.10.4.2 Session Description Semantics

Field	Semantics
SDP	Inlined content referencing SDP file containing information that the terminal needs in order to be able to receive and consume the content of the service session
SDPStream	Identifies a stream of SDP files by an inlined SDP file. This SDP file contains information that the terminal needs to open a FLUTE session which carries the content referencing SDP file identified by SDPURI.
SDPURI	URI referring to a content referencing SDP file containing information that the terminal needs in order to be able to receive and consume the content of the service session. This content referencing SDP file is transported in a FLUTE session signaled by the SDPStream element.
AssociatedDeliveryProcedure	URI referring to the AssociatedDeliveryProcedure configuration file. For details about the AssociatedDeliveryProcedure itself and the transport of the configuration file see [5].This element is optional as the service may choose not to provide this functionality.

5.10.5 Zapping Support

Any normal streamed service may be complemented by an associated zapping support.

Zapping support can be provided to the user with two options:

a) Dynamic Zapping, where the Zapping Support is provided not as part of the ESG data so it can be dynamically changing,

b) Static Zapping, where the Zapping Support is provided inlined.

Both options are described in the following.

Dynamic zapping

The dynamic zapping support is provided as a stream, which contains content to support the zapping to the sessions described by this Acquisition Fragment, e.g. a copy of the audio/video content with reduced performance, a still picture showing the latest snapshot out of the current video or dynamic text such as subtitles. Also a combination of the aforementioned is possible. The zapping support can provide data of different types. The type or types of zapping data can be obtained from the Zapping Support Classification Scheme as declared in Annex C.2. The session description of the Zapping Support Session can be obtained from the ZappingSupportSessionDescription Element.

Static Zapping

The static zapping support is provided in form of inlined data, which contains content to support the zapping to the sessions described by this Acquisition Fragment, such as a still picture giving the impression of the current A/V service, graphics or simple text. Availability of a complementing static zapping support for a particular streaming session is signalled in the Acquisition Fragment by the presence of the MediaLocator element. The type or types of zapping data can be obtained from the Zapping Support Classification Scheme as declared in Annex C.2.

5.10.5.1 Zapping Support Syntax

```
<complexType name="ZappingSupportType">
        <sequence maxOccurs="unbounded">
            <element name="Type" type="tva:ControlledTermType" minOccurs="0"
                maxOccurs="unbounded"/>
            <choice>
                <element name="MediaLocator" type="mpeg7:MediaLocatorType"/>
                <element name="ZappingSDP" type="esg:SessionDescriptionBaseType" minOccurs="0"/>
            </choice>
        </sequence>
    </complexType>
```

5.10.5.2 Zapping Support Semantics

Field	Semantics
Type	Specifies the type of zapping support as a controlled term. A classification scheme of available types e.g. video or image is specified in Annex C.2.
MediaLocator	In case of a static zapping support this field specifies the inlined content for zapping support.
ZappingSDP	In case of a dynamic zapping support this field specifies the session description of the Zapping Support session. The session description is defined in section 5.10.4.

5.10.6 Key stream

Key streams carry key material used to descramble encrypted traffic. The format of the key stream is dependent on the KMS and is identified by the IPDCKMSId.

If required the esg:KeyStreamType can be extended to signal additional information

5.10.6.1 Key Stream Syntax

```
<complexType name="KeyStreamBaseType" abstract="true"/>

<complexType name="KeyStreamType">
    <complexContent>
        <extension base="esg:KeyStreamBaseType">
            <attribute name="IPDCKMSId" type="unsignedShort" use="required" />
            <attribute name="IPDCOperatorId" type="string" use="required" />
        </extension>
    </complexContent>
</complexType>
```

5.10.6.2 Key Stream Semantics

Field	Semantics
IPDCKMSId	The ID of the key management system (KMS) as defined and registered by DVB [18][25].
IPDCOperatorId	The ID of the entity who operates this key stream.

5.10.7 Component Characteristic

In this section a datatype is defined to describe the media components of the service or schedule event. In this description video, audio and download components can be distinguished.

5.10.7.1 Component Characteristic Syntax

```
<complexType name="ComponentCharacteristicType" abstract="true">
    <sequence>
        <element name="Bandwidth" type="tva:BitRateType" minOccurs="0"/>
    </sequence>
    <attribute name="purpose" type="string" use="optional"/>
</complexType>

<complexType name="VideoComponentType">
    <complexContent>
        <extension base="esg:ComponentCharacteristicType">
            <sequence>
            <element name="CodecCharacteristic" type="esg:VideoCodecCharacteristicType"
            minOccurs="0"/>
            <element name="FrameRate" type="tva:FrameRateType" minOccurs="0"/>
            <element name="OpenCaptionLanguage" type="language" minOccurs="0"/>
            <element name="SignLanguage" type="tva:SignLanguageType" minOccurs="0"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="VideoCodecCharacteristicType">
    <sequence>
        <element name="Codec" type="tva:ControlledTermType" minOccurs="0"/>
        <element name="ProfileLevelIndication" type="tva:ControlledTermType" minOccurs="0"/>
    </sequence>
</complexType>

<complexType name="AudioComponentType">
    <complexContent>
        <extension base="esg:ComponentCharacteristicType">
            <sequence>
            <element name="Codec" type="tva:ControlledTermType" minOccurs="0"/>
            <element name="Mode" type="tva:ControlledTermType" minOccurs="0"/>
            <element name="Language" type="mpeg7:ExtendedLanguageType" minOccurs="0"
            maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="FileDownloadComponentType">
```

```
        <complexContent>
            <extension base="esg:ComponentCharacteristicType">
                <sequence>
                <element name="FileFormat" type="string" minOccurs="0" maxOccurs="unbounded"/>
                <element name="Storage" type="unsignedInt" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

```
</schema>
```

5.10.7.2 ComponentCharacteric Semantics

ComponentCharacteristic

Field	Semantics
Bandwidth	Signals the bandwidth consumed by the described component. The specified bandwidth can be classified to be the maximal, average or minimal bandwidth.
purpose	Signals the purpose of the component in the context of the service. E.g. an audio component can be the audio track of a video or the audio track for visually impaired persons.

AudioComponentCharacteristic

Field	Semantics
Codec	Signals which audio codec is used to represent the audio. For this purpose terms of a defined classification scheme are referenced.
Mode	Signals in which mode of the audio streams e.g. if the audio stream is a representation of two or more audio channels and their constellation. For this purpose terms of a defined classification scheme are referenced.
Language	Signals the language of speech information in an audio stream.

VideoComponentCharacteristic

Field	Semantics
Codec	Signals which video codec is used to represent the video. For this purpose terms of a defined classification scheme are referenced.
ProfileLevelIndication	Signals which profileLevel is used to represent the video. For this purpose terms of a defined classification scheme are referenced.
FrameRate	Signals the frame rate of the video.
OpenCaptionLanguage	Signals the language of an open caption contained in a video.
SignLanguage	Signals the sign language visualized in a video.

FileDownloadComponentCharacteristic

Field	Semantics
FileFormat	Specifies the file formats of the files contained in the download component. The value of FileFormat shall be a valid string representation of a Mime Type [2].
Storage	Size of the available files signaled in megabytes.

6 ESG Representation

6.1 Introduction

ESG Fragments may be represented in three ways. Firstly, ESG Fragments may be uncompressed, secondly, ESG Fragments may be compressed with GZIP [7], and thirdly, ESG Fragments may be compressed with BiM specified in ISO/IEC 15938-1 [3] as adopted by DVB-GBS in the document "Carriage and signaling of TV-Anytime information in DVB transport streams" [8]. This document specifies amendments in order to correspond to the requirements of compression and simplicity defined for the transport of ESG fragments.

Note: The complete transport object carrying an ESG Container can be compressed with GZIP based on content encoding signaled in ALC/FLUTE as specified for file delivery in [5].

To signal how the ESG XML Fragments are represented the ESG Init Message is defined in subsection 6.2. The representation and encapsulation of the ESG XML Fragments itself are defined in subsection 6.3. Finally the processing rules of those Encapsulated ESG XML Fragments are specified in subsection 6.4.

6.2 ESG Init Message

The ESG Init Message serves the purpose to initialize the reception of the ESG. For this purpose it signals the representation of the ESG, the presence of an index and the DecoderInit. The ESG Init Message is transported in the ESG Init Container (see section 8.1.1) in the ESG FLUTE session in the single stream mode (see section 8.3) or in the FLUTE Session of the Announcement Carousel in the Multiple Stream mode (see section 8.4).

The syntax of the ESG Init Message is defined as follows:

Syntax	No. of bits	Mnemonic
ESG Init Message{		
EncodingVersion	8	uimsbf
IndexingFlag	1	bslbf
reserved	7	
DecoderInitptr	8	bslbf
if(IndexingFlag) {		
IndexingVersion	8	uimsbf
}		
if(EncodingVersion == '0xF1') {		
BufferSizeFlag	1	bslbf
PositionCodeFlag	1	bslbf
reserved	6	
CharacterEncoding	8	uimsbf
if (BufferSizeFlag == '1') {		
BufferSize	24	uimsbf
}		
}		
if(EncodingVersion == '0xF2' || EncodingVersion == '0xF3') {		
CharacterEncoding	8	uimsbf
}		
Reserved	0 or 8+	
DecoderInit( )		bslbf
}		

The semantics of all fields of the ESG Init Message are as defined for the DVB TVA-init message defined in [8], except for the EncodingVersion and DecoderInit fields which are specified in this section.

Field	Semantics
EncodingVersion	This field indicates the method of encoding used to represent the ESG XML Fragments. This field shall be encoded according to Table 6.1.

Table 6.1: Extension of the Encoding version

Value	Encoding version
0x00 to 0xEF	TVA reserved
0xF0	DVB reserved
0xF1	DVB profile of TVA MPEG_7 profile (BiM) ISO/IEC 15938-1 [3] as defined in this specification.
0xF2	GZip encoded
0xF3	No Encoding i.e. raw XML
0xF4 to 0xF7	DVB reserved
0xF8 to 0xFF	User defined

6.2.1 DecoderInit and default ESGMain Element fragment

The DecoderInit is used to configure parameters required for the decoding and/or parsing of the ESG XML fragments and to transmit the initial state of the ESG Document.

The syntax of the DecoderInit is dependent on the encoding used for ESG XML Fragment representation. The encoding used is signaled by the EncodingVersion field within the ESG Init Message. In the case where the EncodingVersion is set to '0xF2' or '0xF3', the reader shall refer to section 6.2.2 for the definition of the DecoderInit syntax. In the case where the EncodingVersion is set to '0xF1', the reader shall refer to section 6.2.3 of this document for the definition of the DecoderInit syntax.

The ESGMain Element fragment defines the initial description of the ESG Document. The transmission of this fragment is not mandatory. If the ESGMain Element fragment is not delivered to the decoder, the decoder is initialised with the default ESGMain Element fragment. The default ESGMain Element fragment is defined in section 5.2.4.

If the ESGMain Element fragment is delivered to the decoder, it shall be encapsulated in the ESG Init Container as defined in the section 7 of this present document, and this ESG Init Container (see 8.1.1) shall be transported in the ESG FLUTE session in the single stream mode (see section 8.3) or in the FLUTE Session of the Announcement Carousel in the Multiple Stream mode (see section 8.4).

6.2.2 Textual DecoderInit

The DecoderInit specified in this section is used to transmit initialisation information required by the textual decoder in the case the EncodingVersion is set to ‘0xF2’ or ‘0xF3’ in the ESG Init Message.

This initialization information is split into two parts. The first part defines the set of namespaces and their prefixes used within the ESG XML Fragments to be delivered. The second part allows the declaration of ESG XML Fragment Types. The numeric value identifies the ESG XML Fragment Type of ESG XML Fragments being delivered within an Encapsulated Textual ESG XML Fragment (see section 6.3.1). Table 6.2 defines values to be used for the ESG XML Fragment Types declared within this specification. It is not a requirement to describe these ESG XML Fragment Types within the DecoderInit(). However if private extensions have been made to the ESG data model, which result in the delivery of new ESG XML Fragment Types, then these shall be declared within the DecoderInit().

The syntax of the DecoderInit is as follows:

Syntax	No. of Bits	Mnemonic
DecoderInit () {		
version	8	uimsbf
length	8+	vluimsbf8
num_namespace_prefixes	8	uimsbf
for(i=0; i<num_namespace_prefixes; i++) {		
prefix_string_ptr	16	uimsbf
namespace_URI_ptr	16	uimsbf
}		
num_fragment_types	16	uimsbf
for(i=0; i<num_fragment_types; i++) {		
xpath_ptr	16	uimsbf
ESG_XML_fragment_type	16	uimsbf
}		
}		

Field	Semantics
version	Specifies the version of the Textual Decoder Init. The value shall be set to "1". Note: • this version is incremented if the specification of ESG Entry is changed in a not forward compatible way; • a receiver should only decode Textual Decoder Inits which it complies to.
length	The length of the Textual Decoder Init in bytes excluding the Version and Length fields. Note: this allows forward compatible implementations even if fields are added in the future to the Textual Decoder Init.
num_namespace_prefixes	This specifies the number of namespace prefixes declared.
prefix_string_ptr	This is the offset in bytes from the start of the string repository of this container to the first byte of the prefix.
namespace_URI_ptr	This is a pointer to a string within the string repository which contains the declared namespaces.
num_fragment_types	This specifies the number of different ESG XML fragment types carried within the ESG.
xpath_ptr	This is the offset in bytes from the start of the string repository of this container to the first byte of the xPath string which identifies the root element of the ESG XML Fragment.
ESG_XML_fragment_type	A 16 bit value which is used within the ESG_XML_fragment_type field of the Encapsulated Textual ESG XML Fragment specified in section 6.3.1 to identify the ESG XML Fragment Type.

6.2.3 BiM DecoderInit

The DecoderInit specified in this section is used to transmit initialisation information required by the BiM Decoder in the case where the EncodingVersion is set to '0xF1' in the ESG Init Message. The Syntax of the DecoderInit is specified in [3].

At least one schema URI shall be transmitted in the DecoderInit. Consequently, the field NumberOfSchemas of the DecoderInit shall be greater than or equal to 1 and the field SchemaURI[0] of the DecoderInit shall be set to 'urn:dvb:ipdc:esg:2005', indicating the use of the schema defined in this specification.

Additionally, in order to support the transmission of supplemental ESG data additional BiM code spaces and schemaIDs shall be reserved by the following settings in the DecoderInit according to [3]: the flag NoAdvancedFeatures shall be set to "0", the flag AdditionalSchemaFlag shall be set to "1", the NumberOfAdditionalSchemas shall be set to "256", NumberOfKnownAdditionalSchemas shall be set to "0", the flag IsThereExternallyCastableType shall be set to "0", and the flag IsThereExternallySubstitutableType shall be set to "0",

The initial state of a BiM decoder for a binary description tree is given by an initial description. In an ESG fragment stream, the initial description is given by the ESG Main fragment according to section 6.2.1.

As a consequence, the InitialDescription() field of the DecoderInit message as specified in [3], shall always be empty.

The table 52 ContextPathCode extends the DVBContextPath of [8] with the values specified in Table 6.2.

Table 6.2: Values of ESG_XML_fragment_type (in the case of textual representation) and ContextPathCode (in the case of BiM representation) to signal ESG XML Fragment Types

Value	ESG XML Fragment Type	XML Data Type
0x0020	esg:ESGMain Fragment	esg:ESGMainType
0x0021	esg:Content Fragment	esg:ContentType
0x0022	esg:ScheduleEvent Fragment	esg:ScheduleEventType
0x0023	esg:Service Fragment	esg:ServiceType
0x0024	esg:ServiceBundle Fragment	esg:ServiceBundleType
0x0025	esg:Acquisition fragment	esg:AcquisitionType
0x0026	esg:Purchase Fragment	esg:PurchaseType
0x0027	esg:PurchaseChannel Fragment	esg:PurchaseChannelType

To support ESG extensions additionally a mode of variable length ContextPathCode is defined below.

6.2.3.1 ContextPathCode with variable length

The variable length ContextPathCodes are signalled in the ContextPathTable in the DecoderInit as specified in section 7.2 of [3]. In the case variable length ContextPathCodes are signalled the ContextPathTableFlag shall be set to '1' in the DecoderInit according to [3].

6.2.3.2 DVB Datatype Codecs

The following defines a set of codecs that shall be used by default for the encoding of ESG XML Fragments if BiM is used for the representation of ESG XML Fragments.

In the MPEG-7 framework, the use of a specific codec for a specific type is signalled using the codec configuration mechanism defined in [3]. This mechanism associates a codec using its URI with a list of schema types. For that purpose, a URI is assigned to each codec in a classificationScheme, which defines the list of the specific codecs.

In this present document, this list is composed of the following codecs:

- dvbStringCodec (as it is specified in [8])
- dvbDateTimeCodec (as it is specified in [8])
- dvbDurationCodec (as it is specified in [8])
- dvbControlledTermCodec (as it is specified in[8])

The following figure lists the codec definitions of the standard ClassificationScheme used by this present document.

```
<ClassificationScheme uri="urn:tva:metadata:2004:cs:CodecTypeCS ">
    <Term termID="1">
        <Name xml:lang="en">dvbStringCodec</Name>
        <Definition xml:lang="en">Encodes string by using an external string
             buffer</Definition>
    </Term>
    <Term termID="2">
        <Name xml:lang="en">dvbDateTimeCodec</Name>
        <Definition xml:lang="en">Encodes date using Modified Julian Date & Time in
            Millisecond and differential encoding</Definition>
    </Term>
    <Term termID="3">
        <Name xml:lang="en">dvbDurationCodec</Name>
        <Definition xml:lang="en">Encodes duration using strings or
    approximation with an accuracy of 1 minute </Definition>
    </Term>
    <Term termID="5">
        <Name xml:lang="en">dvbControlledTermCodec</Name>
        <Definition xml:lang="en">Encodes Controlled Terms using indices</Definition>
    </Term>
</ClassificationScheme>
```

Figure 6.1: DVB codec classification

6.3 Encapsulated ESG XML Fragment

In this section the Encapsulated ESG XML Fragments carried in the ESG Data Repository as described in section 7.4 is specified.

6.3.1 Encapsulated Textual ESG XML Fragment

The Encapsulated Textual ESG XML Fragment shall be used for the representation of Textual or GZipped ESG XML Fragments within the ESG Data Repository. The use of Encapsulated Textual ESG XML Fragments is mandated when the EncodingVersion field within the ESG Init Message is set to "0xF2" or "0xF3".

Syntax	No. of Bits	Mnemonic
EncapsulatedTextualESGXMLFragment () {		
ESG_XML_fragment_type	16	uimsbf
Data_length	8+	vluimsbf8
for(i=0; i<Data_length; i++) {		
data_byte[i]		
}		
}		

Field	Semantics
ESG_XML_fragment_type	An identifier which enables a terminal to know what type of ESG XML Fragment this is (e.g. Content Fragment, Service Fragment etc). These values may be declared within the DecoderInit as specified in section 6.2.2 or may be well known by the terminal as specified in table 6.2.
Data_length	Signals the number of data_byte bytes.
data_byte	A single byte forming a part of the GZipped ESG XML fragment in the case EncodingVersion is set to "0xF2" or Textual ESG XML fragment in the case EncodingVersion is set to "0xF3".

6.3.2 Encapsulated BiM ESG XML Fragment

The Encapsulated BiM ESG XML Fragment shall be used for the representation of BiM ESG XML fragments within the ESG Data Repository. The use of Encapsulated BiM ESG XML Fragments is mandated when the EncodingVersion field within the ESG Init Message is set to "0xF1".

Syntax	No. of Bits	Identifier
EncapsulatedBiMESGXMLFragment () {		
external_string_buffer_ptr	16	uimsbf
Fragment_Length	8+	vluimsbf8
ContextPathCode	ContextPathCode_Length	uimsbf
FragmentUpdatePayload(startType)		
nextByteBoundary()		
}		

Field	Semantics
external_string_buffer_ptr	The zero-based offset in bytes from the start of the string repository within this container to the first byte of the external string buffer for the fragment
Fragment_Length	This field specifies in bytes the sum of the length of the ContextPathCode and the length of the FragmentUpdatePayload.
ContextPathCode	This field identifies the element representing the signalled ESG XML Fragment. From this field the startType is derived as specified in [3].

	If no ContextPathTable is transmitted, the field shall be encoded according to Table 6.2. If a ContextPathTable is transmitted, the field shall be encoded according to this table (see section 6.2.3.1).
ContextPathCode_Length	If no ContextPathTable is transmitted, this variable is set to 16 bits. If a ContextPathTable is transmitted, this variable is set to the value ContextPathCode_Length transported in the ContextPathTable (see section 6.2.3.1).
startType	This variable identifies the XML data type signalled by the ContextPathCode according to the XML schema definition. The value of this variable is deduced from the ContextPathCode field. If no ContextPathTable is transmitted, this field is set to the XML data type associated to the value of the ContextPathCode field as defined in Table 6.2. If a ContextPathTable is transmitted, this field is deduced from this table and the ContextPathCode field (see section 6.2.3.1).
FragmentUpdatePayload	Signals the payload of the ESG XML Fragment of XML data type startType as defined in [3], subclause 8.3.

6.4 Processing of ESG XML Representation

6.4.1 Introduction

An ESG metadata system includes a common core set of metadata as defined in section 5 of the present document, to ensure a minimum level of interoperability. Extensibility is a key ESG feature. Backward and possibly forward compatibility shall be maintained when the ESG Schema is extended:

- Forward compatibility means an ESG application that is only aware of a previous version of a schema is able to partially decode a description conformant to an updated version of that schema.
- Backward compatibility means an ESG application that is only aware of a new version of the schema is able to partially decode a description conformant to a previous version of that schema.

The XML Schema [17] that is used as the ESG data model representation language for metadata is the main instrument for this extensibility.

Extension rules to allow the extension of the specification with new ESG data model definitions, or for private extensions are defined in Annex D of the present document.

The namespace of the schemas are used by the application processing the ESG XML Fragments to identify whether they are accessing XML elements or attributes defined in the original ESG schema or the extended schema.

6.4.2 Decoder behaviour with XML Encoding

The XML syntax signals to the ESG decoder which schema each element or attribute belongs to. Parts of ESG XML Fragments defined in the original schema or in the extended schema are identified in ESG XML Fragments by using schema namespaces to fully qualify element names, possibly by using namespace prefixes. This allows the ESG Decoder to avoid processing parts of the ESG XML fragments being parsed and which are related to an unknown schema.

The Textual DecoderInit, as defined in section 6.2.2 of the present document, identifies schema versions which are used in the ESG XML fragment stream.

6.4.3 Decoder behaviour with BiM Encoding

With BiM, backward compatibility is provided by the unique reference of the used schema in the DecoderInit. Forward compatibility is ensured by a specific syntax defined in sections 7 and 8 of [3].

The binary format allows one to keep parts of a description related to different schemas in separate chunks of the binary description stream, so that parts related to an unknown schema may be skipped by the decoder. The DecoderInit, as

defined in the section 6.2.3 of the present document, identifies schema versions with which compatibility is preserved by listing their Schema URIs. A decoder that knows at least one of the Schema URIs will be able to decode at least part of the binary description stream.

7 ESG Fragment Encapsulation

7.1 Overview

Fragmentation is the generic decomposition mechanism of the ESG into self-consistent units of data.
In this context self-consistency capability of an ESG Fragment means that:

- ESG Fragments can be obtained in a random order.
- Each ESG Fragment can be transmitted and updated independently.

Different types of fragments may exist:

- ESG XML fragments
- ESG Auxiliary data
- Private Auxiliary data

Note: The transport of ESG Fragments is only specified inside ESG Containers as described in this section with one exception: the transport of SDP files describing sessions. The transport of those SDP files is motivated in section 5.10.1 and specified in section 8.5.

The encapsulation of ESG Fragments serves three purposes

- Aggregation: to reduce the overhead of fragment management information and to support the processing and transmission of ESG information of considerable size, ESG Fragments are aggregated into ESG Containers.

- Fragment Management: to enable the management of fragment creation, deletion and updates over time the fragment management information is signaled for each fragment. This allows a terminal to identify new versions of a fragment without actually reading and comparing the content of the fragments.

- Processing Support: To support a fast processing of ESG Fragments on the terminal redundant data is added into the ESG Container to support the fast random access to the content of ESG Fragments.

The specification of the ESG Fragment Encapsulation is divided into these three parts: ESG Container, ESG Fragment Management Information and ESG Data Repository.

Note: This section specifies the encapsulation of ESG Fragments for the case of unidirectional communication scenarios. The communication of ESG Fragments in point to point communication scenarios is not described in this version of the specification.

7.2 ESG Container

The ESG Containers are transport objects delivered by the transport layer. They aggregate ESG Fragments to enable the efficient transport and processing of ESG data. Beside the specification of the syntax and the semantics of the data fields of an ESG Container also the notion of container identity is specified.

7.2.1 ESG Container Identity and Versioning

The ESG Container is provided a unique identifier (Container_ID) and versioning information. Both Container_ID and versioning information shall be conveyed at the transport layer. Container_ID shall be a unique number within the scope of the ESG Fragment Stream. It is valid to re-use a Container_ID value, provided that sufficient time has elapsed since the Container_id value was last used. Signaling the Container_ID and versioning information of a container enables the terminal to determine if an update action has to be taken on a set of fragments cached in the terminal. Operations signaled by incrementing the versioning information of the container are:

a) an already contained fragment is updated

b) a new fragment is added to the container.

c) a contained fragment is removed from the container.

Consequently an ESG Container has the same identity as long as

d) a fragment is not moved from or to the identified container.

7.2.2 ESG Container Syntax

Syntax	No. of Bits	Mnemonic
container () {		
container_header {		
num_structures	8	uimsbf
for (j=0; j<num_structures; j++) {		
structure_type[j]	8	uimsbf
structure_id[j]	8	uimsbf
structure_ptr[j]	24	uimsbf
structure_length[j]	24	uimsbf
}		
}		
for (j=0; j<byte_count; j++) {		
structure_body[j]		
}		
}		

Note: Within the container the fragment management information (fragment_id, fragment_version, fragment_reference, see section 7.3) is aggregated together, separated from the fragments and it is possible to consume the management information before the fragments in the container.

7.2.3 ESG Container Semantics

Field	Semantics
num_structures	This field specifies the number of structures contained within this container. A value of 0x00 is invalid
structure_type[j]	This field specifies the type of structure being referenced, according to Table 7.1
structure_id[j]	This field identifies multiple occurrences of a specific `structure_type`. In some cases this is just an instance identifier and in other cases it is used to distinguish the type of data carried within the structure (e.g. data repository) as specified in Table 7.2.
structure_ptr[j]	This field specifies the offset in bytes from the start of this container to the first byte of the identified structure.
structure_length[j]	This field specifies the length in bytes of the structure pointed to by `structure_ptr[j]`.
structure_body[j]	Data forming one or more structures within this container.

It is recommended that entries within the container_header are ordered in ascending structure_type and structure_id. For example all structures of type data_repository shall be grouped together and items within the group ordered in ascending structure_ id. This enables a device to efficiently locate a particular structure of interest.

Table 7.1: structure_type assignments

Value	Description
0x00	Reserved
0x01	Fragment Management Information (see section 7.3)
0x02	Data Repository
0x03	Index List
0x04	Index
0x05	Multi Field Sub Index
0xE0	ESG Data Repository
0xE1	ESG Session Partition Declaration (see section 8.4.2.1)
0xE2	ESG Init Message (see section 6.2)

Table 7.2: structure_type and their matching valid structure_id

Structure_type	structure_id	Description
0x01	0x00	Fragment Management Information (see section 7.3)
0x02	0x00	String repository: Data Repository of type String (see section 4.8.4.1 in [21])
0x03	0x00-0xFF	Reserved
0x04	0x00-0xFE	Used to identify a specific instance of an index structure, within a container.
0x05	0x00-0xFE	Used to identify a specific instance of an multi_filed_sub_index structure, within a container.
0xE0	0x00	ESG Data Repository (see section 7.4)
0xE1	0xFF	ESG Session Partition Declaration (see section 8.4.2.1)
0xE2	0x00	ESG Init Message (see section 6.2)

7.3 ESG Fragment Management Information

The Fragment Management Information provides the encapsulation mechanism for a set of ESG fragments, by providing the ability to assign a unique identifier (`fragment_id`) for the lifetime of an ESG fragment and indicating the current version of an ESG fragment.

Each entry references a single ESG fragment carried within the same container.

Note: There shall only ever be one Fragment Management Information structure defined within a single container.

7.3.1 ESG Fragment Management Information Syntax

The ESG Fragment Management Information syntax is provided by the table defined below.

Syntax	No. of Bits	Mnemonic
encapsulation_structure () {		
encapsulation_header {		
reserved_other_use	2	bslbf
reserved	6	bslbf
fragment_reference_format	8	uimsbf
}		
for(j=0; j<fragment_count; j++) {		
encapsulation_entry {		
fragment_reference()[i]		
fragment_version[i]	8	uimsbf
fragment_id[i]	24	uimsbf
}		
}		
}		

7.3.2 ESG Fragment Management Information Semantics

Field	Semantics
reserved_other_use	This field shall be set to "11".
fragment_reference_format	This 8 bit value defines the format and interpretation of the `fragment_reference` field. For semantics of the value see Table 7.3.
fragment_reference()[i]	This field specifies a reference to an encapsulated ESG Fragment. The interpretation of this field is dependent on `the fragment_reference_format`. Please refer to Table 7.3 to determine how this field should be interpreted.
fragment_version[i]	An 8 bit value which identifies the version of the encapsulated ESG Fragment referenced by this entry. When the data for the fragment identified by the `fragment_id` changes, the fragment_version shall increment modulo 255.
fragment_id[i]	A 24 bit value which uniquely identifies an ESG Fragment within the ESG Fragment Stream. The value assigned to an ESG Fragment shall be persistent for the life of that ESG Fragment so long as it is transmitted in the ESG Fragment Stream. It is valid to re-use a fragment_id value, provided that sufficient time has elapsed since the fragment_id value was last used. All entries within the `encapsulation_structure` shall be ordered by ascending `fragment_id`. This enables the efficient location of a fragment by using a binary search algorithm.

Table 7.3: Valid fragment_reference_formats

Value	Meaning
0x00-0x20	Reserved
0x21	Generic ESG Fragment Reference (see section 7.3.3)
0x22 – 0xE0	Reserved
0xE1- 0xFF	User defined

7.3.3 Generic ESG Fragment Reference Syntax

Syntax	No. of Bits	Mnemonic
fragment_reference () {		
esg_fragment_type	8	bslbf
esg_data_repository_offset	24	bslbf
}		

7.3.4 Generic ESG Fragment Reference Semantics

Field	Semantics
esg_fragment_type	This field specifies the Encapsulated ESG Fragment Type referred to by this reference. This field is coded as specified in Table 7.4.
esg_data_repository_offset	This field specifies the offset of the referenced Encapsulated ESG fragment from the start of the ESG Data Repository.

Table 7.4: Valid esg_fragment_type values

Value	Meaning
0x00	Encapsulated ESG XML Fragment (see section 7.4.3)
0x01	Encapsulated ESG Auxiliary Data (see encapsulated _aux_data() in section 7.4.5)
0x02	Encapsulated Private Auxiliary Data (see encapsulated _aux_data() in section 7.4.5)
0x03 – 0xE0	Reserved
0xE1- 0xFF	User defined

Note: ESG Auxiliary Data and Private Auxiliary data are represented by the same data structure.

7.4 ESG Data Repository

In this section the ESG Data Repository is specified. The ESG Data Repository can hold any type of ESG Fragment. The type of the ESG Fragment and the position inside the ESG Data Repository is signalled by the Fragment Management Information (see section 7.3).

7.4.1 ESG Data Repository Syntax

Syntax	No. of Bits	Identifier
esg_data_repository() {		
for (i=0; i<fragment_count; i++) {		
if(esg_fragment_type==0x00){		
encapsulated_esg_xml_fragment()		
}		
if(esg_fragment_type==0x01 || esg_fragment_type==0x02){		
encapsulated_aux_data()		
}		
}		
}		

7.4.2 ESG Data Repository Semantics

Field	Semantics
encapsulated_esg_xml_fragment()	This field specifies an Encapsulated ESG XML Fragment as defined in section 6.3.
encapsulated_aux_data()	This field specifies an Encapsulated ESG or Private Auxiliary Data Fragment (see section 7.4.3).

7.4.3 Encapsulated ESG XML Fragment Syntax

Syntax	No. of Bits	Identifier
encapsulated_esg_xml_fragment() {		
if(EncodingVersion == '0xF1'){		
EncapsulatedBiMESGXMLFragment()		
}		
if(EncodingVersion == '0xF2' || EncodingVersion == '0xF3'){		
EncapsulatedTextualESGXMLFragment()		
}		
}		

7.4.4 Encapsulated ESG XML Fragment Semantics

Field	Semantics
EncodingVersion	The parameter is signalled in the ESG Init Message specified in section 6.2.
EncapsulatedBiMESGXMLFragment	The Encapsulated BiM ESG XML Fragment is specified in section 6.3.2.
EncapsulatedTextualESGXMLFragment	The Encapsulated Textual ESG XML Fragment is specified in section 6.3.1.

7.4.5 Encapsulated Auxiliary Data Syntax

Syntax	No. of Bits	Mnemonic
encapsulated_ aux_data {		
Binary_header {		
Encoding metadataURI Mimetype	16	uimsbf
anyAttribute()		
}		
nextByteBoundary()		
AuxDataLength	n*8	vluimsbf8
AuxData	AuxDataLength*8	
}		

Syntax	No. of Bits	Mnemonic
anyAttribute {		
hasExtension	1	
while(hasExtension ==1) {		
versionId	8	uimsbf
length	5+	vluimsbf5
extension	length	uimsbf
hasExtension	1	bslbf
}		
}		

Note: this binary format defines the format for the transport of ESG Auxilary Data.This format is compatible with a BiM representation of the XML Envelope. The BiM profile is the one defined in section 6.2.3, the XML Schema definition of the XML Envelope is the schema defined in section 7.4.7.

7.4.6 Encapsulated Auxiliary Data Semantics

Field	Semantics
Encoding **metadataURI** **Mimetype**	This field specifies an offset in the String Repository to a list of three null separated strings: • the first string specifies the Content-Encoding according to [19][20], • the second string the URI of the auxiliary data and • the third string the Mime type of the auxiliary data. It is encoded as 3 strings according to the DVBStringCodec as specified in [8].
AuxDataLength	Specifies the length of the following AuxData field in number of bytes.
AuxData	The content of the file

Field	Semantics
anyAttribute	This function allows the encoding of the wildcard (AnyAttribute) defined in the XML Schema definition of the XML Envelope in section 7.4.7. Basically, this encoding is a loop of extensions. For each extension, a versionId allows the decoder to know if the extension is known or not. If the versionId is unknown the decoder should skip the extension field.
hasExtension	This field specifies if an extension is coded or not.
versionId	This field specifies the versionId of the extension. This field shall be encoded as specified in Table 7.5. This Id identifies the schema definition for these new attributes.
length	The length in bits of the encoding of new attributes
extension	This field is the encoding of new attributes, which belongs to the schema identified by the versionId field. It should be decoded by a decoder, which knows the schema represented by the versionId and skipped by the other decoders using the length field. From this schema identified by the versionId and the BiM specification, the binary representation of these attributes can be deduced. .

Table 7.5: VersionId Values

Value	Meaning
0x00 - 0xEF	DVBReserved.
0xF0 - 0xFF	UserPrivate

7.4.7 XML envelope

In this section the XML Schema Definition of a metadata envelope is specified to illustrate the mapping between the elements and attributes of the metadata envelope as specified in [15] and the binary representation specified in section 7.4.5.

```
<schema xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
                attributeFormDefault="unqualified">
    <element name="metadataEnvelope">
        <complexType>
            <sequence>
                <element name="AuxData" type ="AuxDataType" minOccurs="1" maxOccurs="1"/>
            </sequence>
            <attribute name="metadataURI" type="anyURI" use="required"/>
            <attribute name="encoding" type="string" use="required"/>
            <attribute name="mimetype" type="unsignedShort" use="required"/>
            <anyAttribute processContents="skip"/>
        </complexType>
    </element>
    <!-- the type of data carried in this document is encoded in base 64in the format defined by the encoding
                attribute -->
    <complexType name="AuxDataType">
        <simpleContent>
            <restriction base="base64Binary"/>
        </simpleContent>
    </complexType>
<schema>
```

8 ESG Transport

In this section the transport of ESG Containers is described. This specification supports two modes:

A) the single stream mode,

B) the multiple stream mode.

The mode used is signalled by the MultipleStreamTransport field in the ESGAccess Descriptor (see section 9.1.2). In both modes ESG Containers are transported in FLUTE dynamic file delivery carousel sessions as described for file delivery in [5]. The transport of ESG Containers is specified in section 8.1.

To enable a terminal to track changes to fragments without having to acquire all the ESG Containers within an ESG session, a fragment indexing structure has been specified in section 8.2.

In the single stream mode the ESG Containers are transported as transport objects in a single FLUTE session. The FLUTE Session for the transport of ESG Container is based on the file delivery specified in [5] in the mode as described in section 8.3.

In the multiple stream mode the ESG Containers are transported in multiple FLUTE sessions which are distributed over several IP streams. This mode is specified in section 8.4.

In addition the transport of SDP files for the acquisition of content is specified in section 8.5.

8.1 Transport of ESG

In this section the transport of ESG Containers in FLUTE sessions is described as it applies to both modes of ESG transport, the single stream and the multiple stream mode. Section 8.1.1 specifies how ESG Containers are carried in Transport Objects, which relies on the FDT for signaling of ESG Container IDs and version changes. Section 8.1.2 specifies an optional signaling of ESG Container IDs and Versions based on the Split TOI mechanism. In Section 8.1.3 signaling of ESG consistency and completeness is described.

8.1.1 Transport of ESG Containers

In both modes the single stream mode and the multiple stream mode ESG Containers are transported as files in Transport Objects in FLUTE sessions. Files that are ESG Containers are signaled in the FDT by setting the attribute Content-Type=”application/vnd.dvb.esgcontainer”. The ESG Containers are identified based on the URI signaled in the “Content-Location” attribute of the “File” elements in the FDT (see section 8.1.2). The version change of the ESG Containers are signaled based on the TOI and the FDT Instance ID as described in section 8.1.2.

The ESG Containers may be compressed with GZIP [7]. To signal that the ESG Container is compressed with GZIP the Content-Encoding attribute in the FDT shall be set to “gzip”. The encoding of transport objects shall not change for a particular ESG Container.

Initialisation information for the processing of ESG data shall be carried in one ESG Container called the ESG Init Container. In this specification the initialisation information is composed of

- one ESG Init Message specified in section 6.2,
- an optional ESGMain Fragment as specified in section 6.2.1,
- an optional Index List Structure as specified in section 8.2,
- an optional Index Structure as specified in section 8.2,
- and in case the multiple stream mode one ESG Session Partition Declaration specified in section 8.4.2.1.

The Transport Object carrying the ESG Init Container shall be identified by the containerID 1 (see section 8.1.2).

Note: The FDT has to be processed to identify the transport object carrying the ESG Init Container. Then the ESG Init Container has to be processed to initialize the ESG processing.

8.1.2 ESG Container Identification and Version Information using ALC/FLUTE

ESG Containers are used to carry a number of different types of ESG data. This ranges from ESG initialization information (e.g. ESG Init Message, ESG Session Partition Declaration, Index data) to ESG fragments.

An ESG Container is identified by a 16bit integer value (Container_ID) that shall be unique within an ESG Fragment Stream. When containers are delivered in the multiple stream mode, the Container_ID shall be unique across all IP streams forming that ESG.

An ESG Container is delivered as a file within a FLUTE session (see section 8.1.1). Since an ESG Container is identified using its Container_id value, this shall be used to form a unique URI, that is used for the "Content-Location" attribute of the "File" element within the ESGs FDT.

The URI shall conform to the following format:

<context>:<Container_ID>

Where the "<Container_ID>" tag is replaced by the actual ESG Container ID, represented as a Alphanumeric representation of its decimal value.

Where it is recommended that the “<context>” tag is replaced by “urn:dvb:ipdc:cid”.

An example of an instantiation for an ESG Container with Container_ID = 23 is:

urn:dvb:ipdc:cid:23

Section 7.2.1 specifies the requirement on the transport to signal version changes of an ESG Container.

The version change of the ESG Containers are signaled based on the TOI and the FDT Instance ID as described for file delivery in section 6.1.12 of [5].

In addition when using the Split TOI mechanism described in section 8.1.3 a container Version_ID is carried as part of the TOI, and so version changes can be directly inferred by the terminal from the TOI (see section 8.1.3.1).

8.1.3 Version Signaling in the Split TOI field

While using FLUTE as the transport protocol for the ESG, it is mandatory to signal ESG Container ID and version changes in the FDT as described in section 8.1.2. Additionally the ESG Container ID and version may be signaled by the mechanisms described in this section, which define how the TOI field provides the required versioning information.

FLUTE is built on Asynchronous Layered Coding (ALC), version 1. ALC is itself a protocol instantiation of Layered Coding Transport building block (LCT) [9][10][11].

The LCT TOI field is 32*O + 16*H bits in length where the Transport Object Identifier flag (O) length is 2 bits and the Half-word flag (H) length is 1 bit. The maximal length of the TOI is therefore 112 bits (i.e. 14 bytes).

The present specification provides the possibility to use the TOI field in order to indicate the identifier of the transported object, and the version of this latter as well.

When a version identifier is assigned to a transported object through the LCT header, the TOI field is split into two parts: the first part (Most Significant Bits) is allocated to the actual object identification (e.g. a Container_ID), the second part (Less Significant Bits) is allocated to the version identifier (e.g. a container Version_ID).

The receiver detects the fact that the TOI is split (or not) thanks to out-band signalling (these mechanisms are outside the scope of this specification) or thanks to the FLUTE FDT of the actual delivery session. To carry this information, a new attribute "Version-ID-Length" is defined within the FLUTE FDT Instance. This attribute is used to define the structure of the TOI field. This attribute MAY be common for all the delivered objects of a given FDT Instance, or MAY be provided for individual files in the "File" elements of the FDT Instance. Where this attribute appears in both the "FDT-Instance" and the "File" elements, the value of the attribute provided in the "File" element takes precedence. The "Version-ID-Length" attribute is optional. The receiver SHALL understand that a given TOI is not split when the "Version-ID-Length" attribute is neither present within the given "File" element attributes of the FDT, nor the FDT common attributes and when no out-band signalling indicates that this given TOI is split.

The use of the Split TOI shall not change for the identified object e.g. for a particular ESG Container.

Note: The Split TOI can not be used for the FDT itself. The TOI of the FDT is fixed to “0”. The FDT Instance ID is carried in the LCT EXT_FDT Header.

Note: The Split TOI reserves a TOI value range for an identified object. A TOI of a reserved TOI value range shall not be used by another identified object. As the TOI of the FDT is fixed to “0” the object identifier “0” (e.g. the Container_ID) shall not be assigned in the case the Split TOI is used.

The XML Schema definition of the "Version-ID-Length" attribute is the following:

```
<schema targetNamespace="urn:dvb:ipdc:esg_flute_ext:2005" xmlns:ef="urn:dvb:ipdc:esg_flute_ext:2005"
                xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
                attributeFormDefault="unqualified">
    <attribute name="Version-ID-Length"
        type="unsignedLong"
        use="optional"/>
```

8.1.3.1 Expected Receiver behavior while using the FDT

When a terminal receives a transport object in a session for the first time, it detects through the FDT whether the object TOI is split or not, and in the case it is split, the length of the Version_ID.

For example:
Within a 32bit TOI field, if the "Version-ID-Length" equals to 16, the 16 less significant bits are allocated to the Version_ID, and the 16 most significant bits are allocated to the Container_ID of the given transported object (in case the transported object is a ESG Container).

This way, the terminal can distinguish based on the TOI field updated transport objects from completely new objects. The terminal MAY locally replace a given object whose version is older than the delivered version, immediately after the reception of the new-version object (i.e. without waiting to receive the FDT).

8.1.3.2 Example of FDT Instance that carries TOI splitting information (informative)

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance
xmlns ="urn:dvb:ipdc:cdp:flute:fdt:2005" xmlns:ef="urn:dvb:ipdc:esg_flute_extension:2005"
Expires="2890842807">
    <File
        Content-Location="urn:dvb:ipdc:esg:cid:1"
        ef:Version-ID-Length="16"
        TOI="65537"/>
    <File
        Content-Location="urn:dvb:ipdc:esg:cid:2"
        ef:Version-ID-Length="16"
        TOI="131073"/>
    <File
        Content-Location="urn:dvb:ipdc:esg:cid:3"
        TOI="196608"/>
</FDT-Instance>
```

In this example, the objects which TOI are "65537" and "131073" are for example ESG Containers for which the Split TOI is declared. For these two transported objects, the associated LCT TOI field is split into two parts: the first part (Most Significant Bits) of the field is allocated to the Container_ID and the second part (Less Significant Bits) is allocated to the 16bit Version_ID.

For the first object, the TOI field, in a binary format, is: "0100000000000000001"
Thus, the Container_ID is equal to "1" and the current Version_ID is equal to "1".

For the second object, the TOI field in a binary format is: "1000000000000000001"
Thus, the Container_ID is equal to "2" and the current Version_ID is equal to "1".

The transported object which TOI is "196608" (TOI field equals to "1100000000000000000" in a binary format) is a ESG Container for which no Split TOI is declared.

8.1.4 ESG consistency

For transport, an instance of the ESG Data Model is decomposed into ESG XML Fragments. It is mandated that the instance of the ESG Data Model is a valid instance according to the XML Schema Definition [17]. In addition the ESG XML Fragments contain references to other ESG XML Fragments. The instance of the ESG Data Model is called consistent, if the referenced ESG XML Fragments exist in the instance of the ESG Data Model.

For example by adding a particular ESG XML Fragment the consistent instance of the ESG Data Model can become inconsistent if this ESG XML Fragment refers to an ESG XML Fragment which is not contained in the instance. This instance can become consistent again as soon as the referred ESG XML Fragment is added to the instance. The ESG provider should only provide consistent instances of the ESG Data Model.

As the instance of the ESG Data Model is decomposed into ESG XML Fragments those ESG XML Fragments building up an Instance at a certain point in time have to be indicated. To enable a fast determination of the Transport Objects containing ESG XML Fragments building the most recent instance of the ESG Data Model the following signaling in the FDT is specified.

The ESG Containers are delivered over FLUTE dynamic file delivery carousel sessions, as defined in [5].

At each given time, and within each FLUTE session that is used for the transport of ESG Containers, the receivers are proposed a set of Transport Objects by the sender. Note that this is up to each receiver to filter the Transport Objects of its interest within the set of Transport Objects.

A new attribute "FullFDT" is created within the element "FDT-Instance" of the FDT to indicate to the receivers that the FDT Instance contains the exact set of Transport Objects that are currently scheduled for transmission by the sender, in the actual FLUTE session.

This attribute differs from the existing “Complete” attribute in that the “Complete” attribute indicates that no new objects description will be provided in future FDT Instances within this session.

The XML syntax of the "FullFDT" attribute within the FLUTE FDT is the following:

```
    <attribute name="FullFDT"
        type="boolean"
        use="optional" default="false" />
</schema>
```

A FDT instance in which the attribute "FullFDT" is set to TRUE, describes all the Transport Objects that are currently scheduled for transmission, at a given time, within the actual FLUTE session.

No assumption SHALL be made about the fact that a given FDT instance for which the attribute "FullFDT" is absent or set to FALSE, contains the exact set of Transport Objects that are currently scheduled for transmission by the sender, in the actual FLUTE session.

When two FDT instances with attribute "FullFDT" is equal to TRUE are received by a receiver and valid in a given time (that is to say they have not expired), the FDT instance with the highest FDT Instance ID SHALL be used by the terminal.

When one FDT instance with attribute "FullFDT" is equal to TRUE, and another FDT instance where the attribute "FullFDT" is absent or set to FALSE, are received and valid in a given time, the terminal should not make any assumptions about the set of transport objects currently scheduled for transmission unless the FDT Instance with attribute "FullFDT" equals to TRUE has a higher Instance ID than the valid FDT instance which attribute "FullFDT" is absent or set to FALSE.

Note that this mechanism applies to each individual ESG FLUTE session that contributes to the actual ESG Fragment Stream.

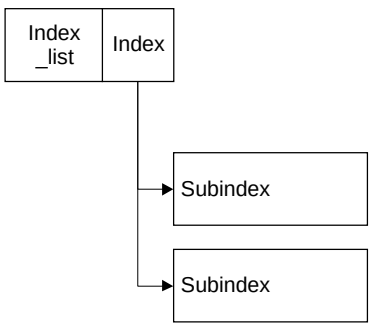
8.2 Indexing overview

To enable a terminal to track changes to fragments without having to acquire all the fragment containers within an ESG session, a fragment indexing structure has been specified. This structure consists of index list, index and one or more optional sub indices. This indexing allows a terminal to monitor for changes to ESG Fragments from one structure.

When a fragment index is transmitted in an ESG multi stream mode, the index shall be carried on the Announcement Carousel of the ESG. In the case of the ESG delivered as a single stream mode, the index shall be carried on the same IPFlow as that of the ESG fragments

The index structure is based on that defined within [21].

The presence of the fragment index is announced using the Index_list structure (see [21]), which is carried within the ESG Init Container (see section 8.1.1).



The diagram depicts the Index structure when separating a single index into multiple Sub Indexes. In the case of a multi stream mode ESG this division point is based on the IPFlow. Therefore the Index is keyed on IPFlow, Fragment_id and

Fragment_version, and a single SubIndex will have all entries from one IPFlow (Session). In the case of a single stream mode ESG there is only one IPFlow in which case there will be one SubIndex, which contains all fragment entries.

8.2.1 IPDC Index specification

8.2.1.1 Index List Structure

The purpose of the Index list structure is to announce the set of available Indexes within the ESG stream. A definition of the Index List structure can be found in section 4.8.5.2 of [21].

A number of parameters are used to announce an Index:

fragment_type – This field identifies the type of fragment that is being referenced by an Index entry. This field may take the value of 0x00, which signals to the terminal that an index entry does not reference any or any specific type of fragment.

num_fields – This field identifies the number of index keys forming the index. For example if we have an index keyed on Service Id and Schedule Event start date/time, we would have two keys i.e. Service Id and Schedule Event start date/time.

field_identifier – This field identifies a key on which the index is based. Typically this is a well known value for a specific field value. In this specification a number of values for this field has been specified (see Table 8.1)

field_encoding – This field signals how the data value associated with the field_identifier are represented/encoded within the sub index structure. A set of valid values has been defined in Table 8.3 and Table 8.4.

index_container – This field signals the identifier of the ESG Container which carries the announced Index.

index_identifier – This field signals the indexes structure_id value within the identified container. This provides the ability to carry multiple Index structures (for different indexes) within the same container.

Table 8.1: Valid field_identifier values.

Value	Description
0x0000	Reserved
0x0001	IPFlowID as declared within the ESG Session Partition declaration
0x0002	fragment_id as declared within the fragment management information
0x0003	fragment_version as declared within the fragment management information
0x0004 - 0xFFFF	Reserved

8.2.1.2 Index Structure

The Index structure is used to declare global settings for the Index and also to declare the set of sub indexes that make up the index and the range of values that can be found within a given sub index. A definition of the Index structure can be found in section 4.8.5.4 of [21].

The container in which each declared sub index is carried is signalled by the sub_index_container field, and the sub index structure within that container is signalled via the sub_index_identifier.

In this specification the overlapping_subindex flag is not supported. Only single_layer_sub_indexes are supported.

8.2.1.3 Sub Index Structure

The Sub Index structure hold the actual index entries where all entries are ordered in increasing order based on each index key. In one ESG fragment stream, only single_layer_sub_index is allowed. Use of multi_layer_sub_index is not supported. A full definition of the single_layer_sub_index structure can be found in section 4.8.5.5 of [21].

8.2.1.4 Fragment Locator References

Fragment Locator references are used with an index entry to reference an ESG fragment.

The following fragment locator type is used within the IPDC ESG fragment index to signal the container in which the declared fragment can be found.

Syntax	No. of Bits	Mnemonic
container_fragment_locator() {		
Container_ID	16	uimbsf
}		

Field	Semantics
Container_ID	The Container in which the indexed fragment can be found.

The use of this container_fragment_locator shall be signalled within the index structure by setting the fragment_locator_format field to a value of 0xE1.

8.2.1.5 ESG Fragment Index

The ESG fragment Index is used to enable a terminal to know in which container a particular fragment is located and which is the current version of that fragment.

8.2.1.6 Index List Structure Instantiation

The following table defines the set of values that the field_identifier and field_encoding fields shall take to announce the fragment index within the Index_list structure. These fields shall be declared in the order in which they appear in the following table.

field_identifier	field_encoding
0x0001 (IPFlowID)	0x0206 (unsigned Byte)
0x0002 (fragment Id)	0x0201 (unsigned long)
0x0003 (fragment_version)	0x0101 (unsigned short)

In addition the fragment_type field shall be set to '0x00'.

8.2.1.7 Index Structure Instantiation

The Index structure for the fragment index shall have the following settings:

Index field	value
overlapping_subindexes	'0'
single_layer_sub_index	'1'
fragment_locator_format	0xE1

The fragment Index is split into a number of sub indexes based on the IPFlowID field. There shall be at least one sub index for each IPFlowID value, where the IPFlowID value corresponds to the declaration within the ESG Session Partition Declaration. In the case of a single stream mode ESG, where there is no session partition strategy declarartion the IPFlowID field shall always take the value of 0x00. Typically a sub index will contain all fragment entries for fragments carried on that IPFlow (identified using IPFlowID). However if the sub index becomes too big then it may be split into further sub indexes, as required.

8.2.2 Multi Field Sub Index Structure Instantiation

The sub index structure for the fragment index shall take the form of a single_layer_sub_index structure. The multi field header shall have the following settings.

Index field	value
leaf_field	'0'
multiple_locators	'0'

Each entry within the single layer sub index structure shall be formed of an IPFlowID, fragment Id, followed by a fragment version. As has been signalled in the Index structure the fragment_locator field shall contain a container_fragment_reference, which signals the container in which the indexed fragment is located.

8.3 ESG Single Stream Transport

In the single stream mode the ESG Containers are transported as transport objects in a single FLUTE session. The FLUTE Session for the transport of ESG Container is based on the file delivery specified in [5]. The FLUTE session is described as specified in section 9.1.2 for the ESGAccessDescriptor.

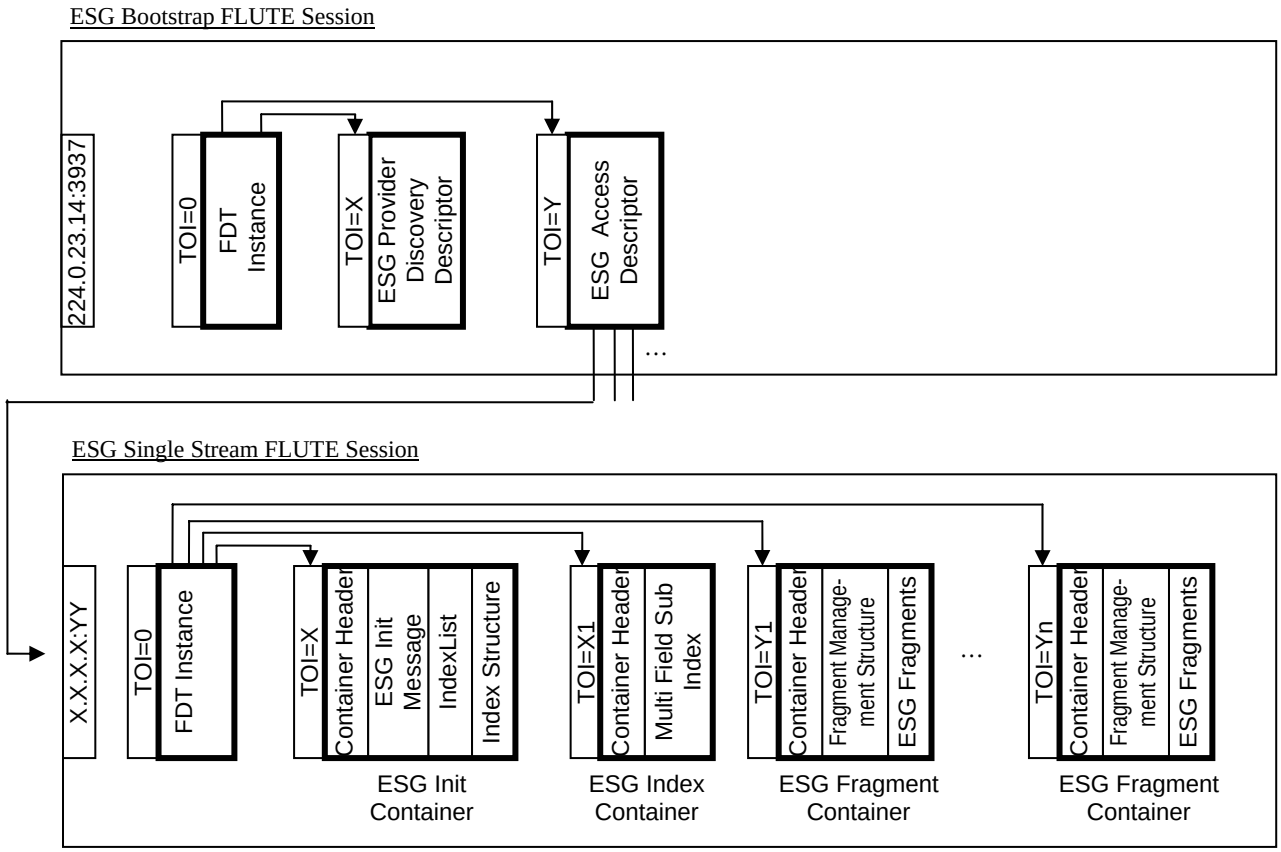


Figure 8.1: Diagram of the Single Stream Transport of the ESG.

8.4 ESG Multiple Stream Transport

In this section the transport of the ESG on multiple IP streams is specified. In the next section an introduction to multiple stream transport is given. The multiple stream transport is divided into FLUTE Sessions transporting parts of the ESG and a particular FLUTE session called Announcement Channel. The Announcement Channel transports initialisation information as specified in section 8.4.2. The remaining FLUTE sessions transporting the ESG are specified in section 8.4.3.

8.4.1 Introduction

The set of ESG Fragments making up an ESG metadata stream, may amount to a large set of data, which could overwhelm the terminal, and/or consume valuable resources (battery power, CPU, memory). It is also envisaged that there will be a wide diversity of terminals deployed with varying functionality, and features. Some terminals will only require a very small subset of the possible ESG data to be able to function, where as others may require richer metadata to support more advanced functionality.

Therefore it is advantageous to be able to split the ESG metadata into a number of separate IP streams. The set of data carried on each IP stream will be based on some partitioning strategy. One common strategy may for example be based on when the content is available (based on its scheduled data time), where one IP stream contains all fragments for content and services available within the next 12 hours. Another IP stream may then contain all fragments for content and services available beyond 12 hours. It is quite possible that fragments are relevant for multiple IP streams, in which case they shall be delivered within the first one that is appropriate and may be delivered in others if needed for consistency of the data.

A partitioning strategy may be based on more than one criterion. A possible strategy may for example be based on Service and scheduled date, time. In this case, one IP stream carries data for Service X available within the next 7 days. Another IP stream may then carry data for Service X available beyond 7 days.

The partitioning strategy used and the set of IP streams that form this partitioned ESG are signalled within an ESG Session Partition Declaration. The ESG Fragments contained in each session are listed in fragment indices.

Both the ESG Session Partition Declaration (defined in this section) and the Fragment Indices (see section 8.2) are carried in the Announcement Carousel on a known IP stream that is signalled within the ESG Access descriptor (see section 9.1.2).

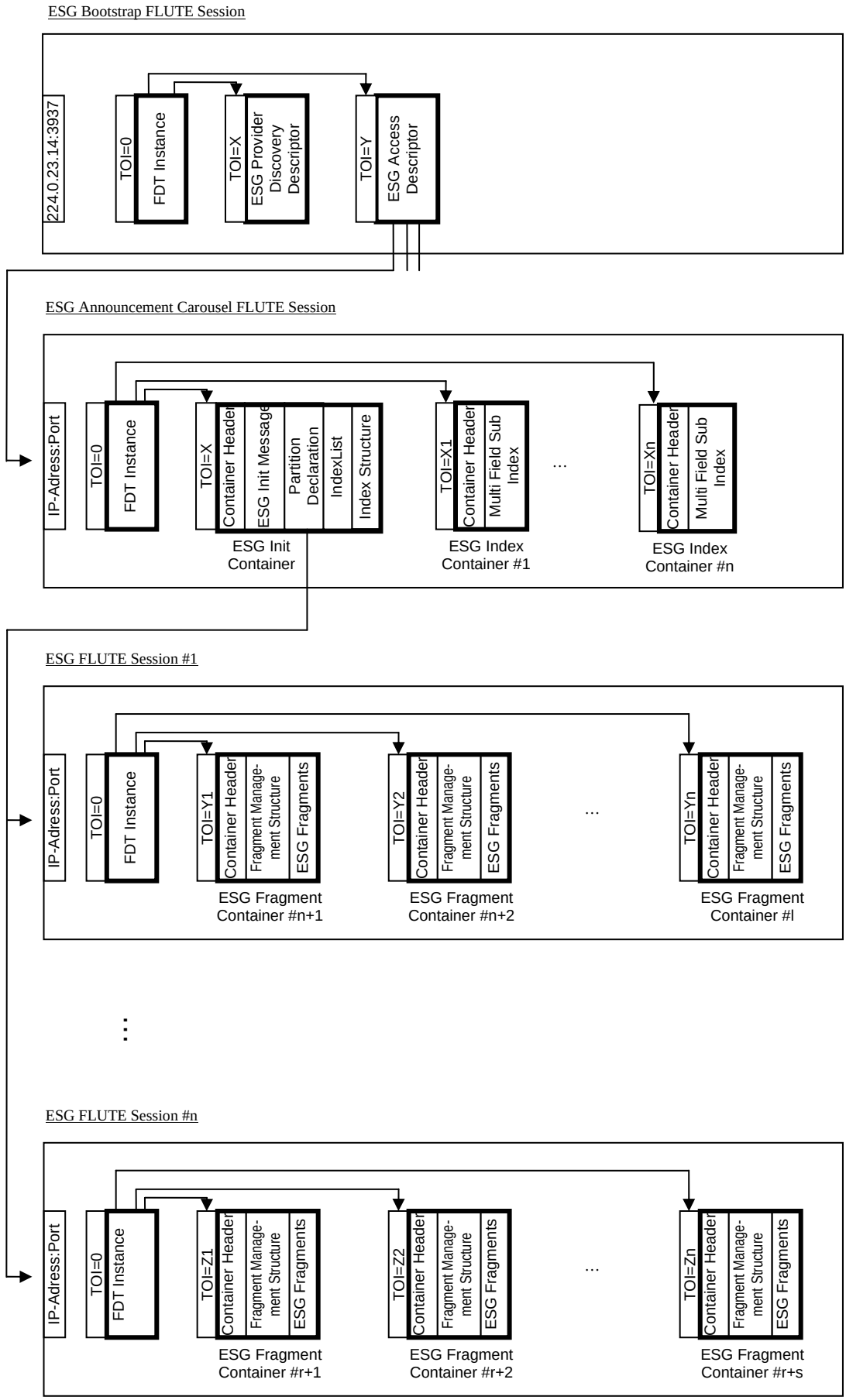

ESG Bootstrap FLUTE Session
224.0.23.14:3937
TOI=0
FDT Instance
TOI=X
ESG Provider Discovery Descriptor
TOI=Y
ESG Access Descriptor
ESG Announcement Carousel FLUTE Session
IP-Adress:Port
TOI=0
FDT Instance
TOI=X
Container Header
ESG Init Message
Partition Declaration
IndexList
Index Structure
ESG Init Container
TOI=X1
Container Header
Multi Field Sub Index
ESG Index Container #1
...
TOI=Xn
Container Header
Multi Field Sub Index
ESG Index Container #n
ESG FLUTE Session #1
IP-Adress:Port
TOI=0
FDT Instance
TOI=Y1
Container Header
Fragment Manage-ment Structure
ESG Fragments
ESG Fragment Container #n+1
TOI=Y2
Container Header
Fragment Manage-ment Structure
ESG Fragments
ESG Fragment Container #n+2
...
TOI=Yn
Container Header
Fragment Manage-ment Structure
ESG Fragments
ESG Fragment Container #l
ESG FLUTE Session #n
IP-Adress:Port
TOI=0
FDT Instance
TOI=Z1
Container Header
Fragment Manage-ment Structure
ESG Fragments
ESG Fragment Container #r+1
TOI=Z2
Container Header
Fragment Manage-ment Structure
ESG Fragments
ESG Fragment Container #r+2
...
TOI=Zn
Container Header
Fragment Manage-ment Structure
ESG Fragments
ESG Fragment Container #r+s

Figure 8.2: Diagram of the Multiple Stream Transport of the ESG and the references between the streams.

8.4.2 Announcement Carousel

The announcement carousel carries the ESG Init Container as specified in section 8.1.1 and optional ESG Containers carrying the Fragment Index (see section 8.2). The ESG Init Container in the Announcement Carousel contains the ESG Session Partition Declaration, which describes a particular partitioning strategy. The ESG Session Partition Declaration is identified within the Container by setting the structure_type field to 0xE1 and the structure_id field to 0xFF within the container header.

The set of containers forming an IP stream shall be delivered using FLUTE as described in section 8.1. The FLUTE Session transporting the Announcement Carousel is described as specified in section 9.1.2 for the ESGAccessDescriptor. The FLUTE Session transporting the ESG Containers are described as specified in section 8.4.2.1 for the ESG Session Partition Declaration.

8.4.2.1 ESG Session Partition Declaration

For signalling of partition strategy the declaration specified in this section is instantiated. The ESG Session Partition Declaration tells the terminal, how the ESG is partitioned, what are the partitioning criteria for each session. ESG Session Partition Declaration is also the entry point to the optional fragment indices that may be transported in the same carousel.

8.4.2.1.1 ESG Session Partition Declaration Syntax

Syntax	No. of Bits	Mnemonic
ESG Session Partition Declaration{		
num_fields	8	uimsbf
overlapping	1	bslbf
reserved	7	bslbf
for(k=0; k<num_fields; k++) {		
field_identifier[k]	16	bslbf
field_encoding[k]	16	bslbf
field_length[k]	8	uimsbf
}		
n_o_IPStreams	8	uimsbf
IPVersion6	1	bslbf
Reserved	7	bslbf
for(i=0; i<n_o_IPStreams; i++){		
IPStreamID[i]	8	uimsbf
if(IPVersion6){		
ESGSourceAddress[i]	128	bslbf
IPAddress[i]	128	bslbf
}else{		
ESGSourceAddress[i]	32	bslbf
IPAddress[i]	32	bslbf
}		
Port[i]	16	uimsbf
SessionID[i]	16	uimsbf
for(k=0; k<num_fields; k++) {		
if(field_length[k]==0){		
length[i][k]	8+	vluimsbf8
}		
if(overlapping){		
start_field_value[i][k]		bslbf
}		
end_field_value[i][k]		bslbf
nextByteBoundary()		
}		
}		

8.4.2.1.2 ESG Session Partition Declaration Semantics

Field	Semantics
num_fields	The number of criteria (fields) that describe the partitioning strategy. For example the strategy may be based on service and broadcast time.
overlapping	When set to '1' signals that identifies if the partitions are overlapping according to the value ranges of the identified fields.
field_identifier[k]	Identifies a criteria (field) on which the fragments are partitioned. This may be a conceptual field, as well as one from the data model (see Table 8.2).
field_encoding[k]	Signals how the field data is encoded. e.g. String, Integer, float (see Table 8.3 and 8.4).
field_length[k]	Specifies the size of the start_value_field[k] and end_value_field[k] in number of bytes. The value of '0x00' is used to indicate that the field is variable in length.
n_o_IPStreams	Signales the number of IP Streams described in this ESG Session Declaration.
IPVersion6	This Flag when set to '1' signals that IP version 6 Addresses are used within the declaration. This Flag when set to '0' signals that IP version 4

	Addresses are used within the declaration.
IPStreamID[i]	Identifies the IP Stream in the ESG Fragment Stream. The IPStreamID is used to key the IP stream in the Fragment Index.
ESGSourceAddress[i]	The ESG IP Source address. The use of this Source Address along with the IPAddress field uniquely identifies the IP Stream.
IPAddress[i]	Destination IP address of indexed delivery session.
Port[i]	The IP port on which the ESG Containers shall be delivered.
SessionID[i]	The ALC Session ID which when combined with the ESGSourceAddress and IPAddress uniquely identifies the ALC session on which the ESG data shall be delivered
length[i][k]	Specifies in Bytes the sum of the lengths of the start_field_value[i][k], end_field_value[i][k] and the nextByteBoundary().
start_field_value [i][k]	Specifies the minimum value of a range of field values. The range of field values characterizes fragments contained in this partition k according to the criteria signaled in the field_identifier[k]. The encoding and size of this field is specified in table 8.4.
end_field_value [i][k]	Specifies the maximum value of a range of field values. The range of field values characterizes fragments contained in this partition k according to the criteria signaled in the field_identifier[k].. The encoding and size of this field is specified in table 8.4.

Table 8.2: Semantics of field_identifier values

Value	Encoding	Meaning
0x00	0x0101 (unsigned short)	The number of hours for which the fragments are valid. This may be used to split the ESG into various schedule depths
0x01	0x0000 (string)	The URI of the Service fragments ServiceId, This may be used to carry all fragments relevant to a particular service
0x02 - 0xEF		DVB Reserved
0xF0 - FE		User Defined
0xFF		Reserved

Table 8.3: Semantics of field_encoding values

Value	Meaning
0x0000	Field containes an inline string in UTF-8 [14]. For this field_encoding the field_length shall be set to ‘0x00’.
0x0001 - 0x00FF	Field is an offset in bytes from the start of the string data repository structure.
0x0100 - 0x01FF	Field contains an inline 2-byte value.
0x0200 – 0x0201 0x0300 0x0401	Field contains an inline 4-byte value.
0x0204 – 0x0206	Field contains an inline 1-byte value.
0x0202 – 0x0203	Field is variable length representation of an integer. For this field_encoding the field_length shall be set to ‘0x00’.
0x0302 0x0400	Field contains an inline 8-byte value.

0x0204 – 0x02FF 0x0402-0x04FF	Undefined
0x0500-0xFFFF	Reserved for future use.

Table 8.4: encoding types and their respective sizes

field_encoding	Description	Encoding	Size in bits
0x0000	string type	Null-terminated string	variable (8+)
0x0001 - 0x00FF	Reserved for custom string types		
0x0100	signed short	two's complement - Big-Endian	16
0x0101	unsigned short	unsigned binary - Big-Endian	16
0x0102-0x01FF	Reserved for custom 2-byte types		16
0x0200	signed long	two's complement - Big-Endian	32
0x0201	unsigned long	unsigned binary - Big-Endian	32
0x0202	variable length signed integer	one bit to indicate sign (0: positive, 1: negative), followed by abs(value) using vluimsbf5.	variable (6+)
0x0203	variable length unsigned integer	vluimsbf8	variable (8+)
0x0204	boolean	0:False 1:True	8
0x0205	signed byte	two's complement	8
0x0206	unsigned byte	unsigned binary	8
0x0207-0x02FF	Reserved for custom integer types		
0x0300	signed float	IEEE standard 754-1985 [16] Big-Endian	32
0x0301	Reserved		
0x0302	signed double	IEEE standard 754-1985 [16] Big-Endian	64
0x0303-0x03FF	Reserved for custom rational types		
0x0400	dateTime	Modified Julian Date and Milliseconds (TVA BiM codec, clause 4.4.2.4.2 in [21])	64
0x0401	date	Modified Julian Date (TVA BiM codec, clause 4.4.2.4.3 in [21])	32
0x0402-0x04FF	Reserved for custom binary fragments		
0x0500-0xFFFF	Reserved for future use		

The ESG Session Partition Declaration shall be carried as a structure within the ESG Init Container. The structure shall be signalled by setting the structure_type field within the container header to 0xE1. The corresponding structure_id field shall be set to 0xff (see section 7.2.3).

Note: The FLUTE Sessions transporting the ESG Containers are described as specified in this section for the ESG Session Partition Declaration. The parameters fixed for the description of the FLUTE session transporting the announcement carousel in section 9.1.2 also apply to all FLUTE Sessions described within the Partition Declaration.

8.5. Transport of SDP Files for Acquisition

SDP Files for the acquisition of content can be transported separate from the ESG Fragments as described in section 5.10.1. These SDP files are transported in ALC/FLUTE sessions as described for file delivery in [5].

Note: According to section 5.10.4 the SDP file may alternatively be inlined in the Acquisition Fragment.

Note: The number of different SDP files transported in one ALC/FLUTE session is not restricted.

Note: The transport of SDP Files in an ALC/FLUTE session MAY use the mechanism to split the LCT TOI field as described in chapter 8.1

9 ESG Bootstrapping

In this section the ESG bootstrap process is specified. As introduced in section 4 several ESGs can be transported in parallel on an IP Platform. To indicate to the receiving terminal the availability of ESGs the specification consists of the following parts:

a) The ESG Bootstrap Descriptors are specified in section 9.1. The Bootstrap Descriptors provide information about the ESG Provider and the Acquisition of available ESGs.

b) The Transport of the Bootstrap Descriptors is specified in section 9.2.

9.1 The ESG Bootstrap Descriptors

Two kinds of Bootstrap Descriptors are signalled: the ESGProviderDiscovery descriptor and the ESGAccessDescriptor.

The ESGProviderDiscovery descriptor specifies the ESG providers that deliver ESGs in a given IP Platform. [1] The ESGProviderDiscovery descriptor is represented as a textual XML file. Based on the ESGProviderDiscovery descriptor the user or the terminal may select the ESG to boot with. Based on the ESGProviderID associated to each described ESGProvider the terminal may parse the related ESGAccessDescriptor to boot the ESG.

The ESGAccessDescriptor is a binary representation of ESG Acquisition information. The ESGAccessDescriptor specifies the Acquisition information related to a particular ESGProviderID signaled in the ESGProviderDiscoveryDescriptor.

9.1.1 ESGProviderDiscovery Descriptor

ESGProviderDiscovery Descriptor Syntax

```
<schema targetNamespace="urn:dvb:ipdc:esgbs:2005" xmlns:bs="urn:dvb:ipdc:esgbs:2005"
                xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns="http://www.w3.org/2001/XMLSchema"
                elementFormDefault="qualified" attributeFormDefault="unqualified">
    <import namespace="urn:mpeg:mpeg7:schema:2001" />

    <complexType name="ESGProviderType">
        <sequence>
            <element name="ProviderURI" type="anyURI"/>
            <element name="ProviderName" type="mpeg7:TextualType"/>
            <element name="ProviderLogo" type="mpeg7:TitleMediaType" minOccurs="0"/>
            <element name="ProviderID" type="positiveInteger"/>
            <element name="ProviderInformationURL" type="anyURI" minOccurs="0"/>
            <element name="PrivateAuxiliaryData" type="anyType" minOccurs="0"/>
        </sequence>
    </complexType>

    <element name="ESGProviderDiscovery">
    <complexType>
        <sequence>
            <element name="ServiceProvider" type="bs:ESGProviderType" maxOccurs="unbounded"/>
        </sequence>
        </complexType>
    </element>
</schema>
```

ESGProviderDiscovery Descriptor Semantics:

[1] It is assumed that a given ESG Provider delivers only one ESG at a given time, in a given IP Platform

Field	Semantics
ProviderURI	Specifies a URI uniquely identifying the ESG provider. For instance this URI can be an internet DNS domain name registered by the ESG Service Provider that uniquely identifies the Service Provider
ProviderName	Name of the ESG Service Provider in a textual format. This name can for instance be displayed to the user.
ProviderLogo	Specifies a representation of the ESG Provider promotional logo.
ProviderID	This ID is used to identify uniquely the ESG provider in the ESG Access Descriptor. The ESG provider must register the ProviderID at the authority that manages the bootstrapping channel to guarantee uniqueness.
ProviderInformationURL	Specifies a URL of more detailed information about the service provider.
PrivateAuxiliaryData	Specifies auxiliary data in a private format. This is an extension point which can be used by the ESG provider for private data.

9.1.2 ESGAccessDescriptor

ESGAccessDescriptor Syntax

Syntax	No. of Bits	Mnemonic
ESG Access Descriptor{		
n_o_ESGEntries	16	uimsbf
for(i=0; i<n_o_ESGEntries; i++){		
ESGEntry[i]()		
}		
}		

Syntax	No. of Bits	Mnemonic
ESGEntry{		
ESGEntryVersion	8	uimsbf
ESGEntryLength	8+	vluimsbf8
MultipleStreamTransport	1	bslbf
IPVersion6	1	bslbf
Reserved	6	bslbf
ProviderID	16	uimsbf
if(IPVersion6){		
SourceIPAddress	128	bslbf
DestinationIPAddress	128	bslbf
}else{		
SourceIPAddress	32	bslbf
DestinationIPAddress	32	bslbf
}		
Port	16	uimsbf
TSI	16	uimsbf
}		

ESGAccessDescriptor Semantics

Field	Semantics
n_o_ESGEntries	Specifies the number of ESGEntries in which access information to ESGs is signaled.
ESGEntryVersion	Specifies the version of the ESG Entry Specification. The value shall be set to "1". Note: - this version is incremented if the specification of ESG Entry is changed in a not forward compatible way; - a receiver should only decode ESG Entries which it complies to.
ESGEntryLength	The EntryLength specifies the length of the ESGEntry in Bytes excluding the ESGEntryVersion and EntryLength fields. Note: this allows forward compatible implementations even if fields are added in the future to the ESGAccessDescriptor.
MultipleStreamTransport	If set to '1' specifies that a FLUTE session is described which transports an Announcement Carousel Session (see section 8.4). If set to '0' specifies that a FLUTE session is described which contains all ESG Containers of that ESG (see section 8.3).
IPVersion6	If set to '1' specifies that the SourceIPAddress and the DestinationIPAddress are signaled according to IP version 6. If set to '0' specifies that the SourceIPAddress and the DestinationIPAddress are signaled according to IP version 4.
ProviderID	This ID is used to identify uniquely the ESG provider in the ESGProviderDiscovery Descriptor. The ESG provider must register the ProviderID at the authority that manages the bootstrapping channel to guarantee uniqueness.
SourceIPAddress	Specifies the source IP address of the FLUTE session transporting the ESG. The IPVersion is signaled by the IPVersion6 field.
DestinationIPAddress	Specifies the destination IP address of the FLUTE session transporting the ESG. The IPVersion is signaled by the IPVersion6 field.
Port	Specifies the port number of the IP Stream of the FLUTE session in which the ESG is transported.
TSI	Specifies the transport session identifier (TSI) of the FLUTE session in which the ESG is transported.

According to the file delivery specification in [5] the following information is required to launch a FLUTE agent in the terminal. The listed fields are specified in the ESGAccessDescriptor except the ones printed italic. For those printed italic default values are assumed as listed.

1. The source IP address
2. *The number of channels in the session is fixed to be **1**.*
3. The destination IP Address of the only channel of the session
4. The port number of the only channel of the session
5. The Transport Session Identifier
6. *The start and end time of the session is fixed to be **0 – 0**.*
7. *The protocol is fixed to be **FLUTE/UDP***
8. *The media type is assumed to be "**application**" and the format list contains only one item "**0**"*

9.2 Transport of ESG Bootstrap Descriptors

The ESG Bootstrap Descriptors are transported on ALC/LCT as specified in [5] for FLUTE sessions.

According to the file delivery specification in [5] the following information is required to launch a FLUTE agent in the terminal. As there is no explicit signaling of the parameters the following parameters are assumed:

1. The source IP address is not fixed
2. The number of channels in the session is fixed to be 1.
3. The destination IP Address of the only channel of the session is fixed to 224.0.23.14 for IP Version 4 or FF0X:0:0:0:0:0:0:12D for IP Version 6 as it is registered for “DvbServDisc” in [13] respectively [23].
4. The port number of the only channel of the session is the port for “ipdcesgbs” as it is registered in [22].
5. The Transport Session Identifier is not fixed.
6. The start and end time of the session is fixed to be 0 – 0.
7. The protocol is fixed to be FLUTE/UDP
8. The media type is assumed to be “application” and the format list contains only one item “0”

Additionally the following restrictions apply:

The FEC scheme is fixed to be the compact no-FEC code as it is specified in [5].

The FLUTE bootstrap session shall be delivered to destination IP address as it is registered for “DvbServDisc” in [13] respectively [23] on the port for ipdcesgbs as it is registered in [22] and shall be the only FLUTE session on that multicast group/port. Thus, the receiver may assume that the first FLUTE session detected (Source IP Address/TSI) on that destination address/port is the bootstrap session.

The ESG Bootstrap Descriptors are transported as files in Transport Objects in FLUTE sessions. The ProviderDiscovery Descriptor file is signaled in the FDT by setting the attribute Content-Type=”text/xml”. The ProviderDiscovery Descriptor shall be represented in UTF-8 character encoding [14]. The ESGAccessDescriptor file is signaled in the FDT by setting the attribute Content-Type=”application/vnd.dvb.ipdcesgaccess”.

Note: The mechanism of Split TOI described in section 8.1.3 may also be applied in the case of ESG Bootstrap Descriptor transport.

Annex A TV-Anytime Datatypes

In this section the datatypes of the TV_Anytime [12] namespace referenced in the IPDC datatype definitions are listed.

```
<complexType name="ServiceInformationNameType">
    <complexContent>
        <extension base="mpeg7:TextualType">
            <attribute name="length" type="tva:ServiceInformationNameLengthType" use="optional"/>
        </extension>
    </complexContent>
</complexType>

<simpleType name="ServiceInformationNameLengthType">
    <restriction base="string">
        <enumeration value="short"/>
        <enumeration value="medium"/>
        <enumeration value="long"/>
    </restriction>
</simpleType>
```

```
<complexType name="SynopsisType">
    <complexContent>
        <extension base="mpeg7:TextualType">
            <attribute name="length" type="tva:SynopsisLengthType" use="prohibited"/>
        </extension>
    </complexContent>
</complexType>

<simpleType name="SynopsisLengthType">
    <restriction base="string">
        <enumeration value="short"/>
        <enumeration value="medium"/>
        <enumeration value="long"/>
    </restriction>
</simpleType>
```

```
<complexType name="GenreType">
    <complexContent>
        <extension base="tva:ControlledTermType">
            <attribute name="type" use="optional" default="main">
            <simpleType>
                <restriction base="string">
                    <enumeration value="main"/>
                    <enumeration value="secondary"/>
                    <enumeration value="other"/>
                </restriction>
            </simpleType>
            </attribute>
        </extension>
    </complexContent>
</complexType>
```

```
<complexType name="KeywordType">
    <simpleContent>
        <extension base="mpeg7:TextualType">
            <attribute name="type" use="optional" default="main">
            <simpleType>
```

```
                    <restriction base="NMTOKEN">
                        <enumeration value="main"/>
                        <enumeration value="secondary"/>
                        <enumeration value="other"/>
                    </restriction>
                </simpleType>
                </attribute>
            </extension>
        </simpleContent>
    </complexType>
```

```
    <complexType name="ControlledTermType">
        <sequence>
            <element name="Name" type="tva:TermNameType" minOccurs="0"/>
            <element name="Definition" type="mpeg7:TextualType" minOccurs="0"/>
        </sequence>
        <attribute name="href" type="mpeg7:termReferenceType" use="required"/>
    </complexType>

    <complexType name="TermNameType">
        <complexContent>
            <extension base="mpeg7:TextualType">
                <attribute name="preferred" type="boolean" use="optional"/>
            </extension>
        </complexContent>
    </complexType>
```

```
    <complexType name="CaptionLanguageType" >
        <simpleContent>
            <extension base="language">
                <attribute name="closed" type="boolean" use="optional" default="true"/>
                <attribute name="supplemental" type="boolean" use="optional" default="false"/>
            </extension>
        </simpleContent>
    </complexType>

    <complexType name="SignLanguageType" >
        <simpleContent>
            <extension base="language">
                <attribute name="primary" type="boolean" use="optional"/>
                <attribute name="translation" type="boolean" use="optional"/>
                <attribute name="type" type="string" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
```

```
    <simpleType name="TVAIDType">
        <restriction base="string">
            <whiteSpace value="collapse"/>
        </restriction>
    </simpleType>
```

```
    <complexType name="ClassificationSchemeTableType">
        <sequence>
            <element name="CSAlias" type="tva:CSAliasType" minOccurs="0" maxOccurs="unbounded" />
            <element name="ClassificationScheme" type="tva:ClassificationSchemeType" minOccurs="0"
                maxOccurs="unbounded" />
```

```
    </sequence>
</complexType>

<complexType name="CSAliasType" >
    <complexContent>
        <extension base="mpeg7:ClassificationSchemeAliasType">
            <attributeGroup ref="tva:fragmentIdentification"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="ClassificationSchemeType" >
    <complexContent>
        <extension base="mpeg7:ClassificationSchemeType">
            <attributeGroup ref="tva:fragmentIdentification"/>
        </extension>
    </complexContent>
</complexType>

<attributeGroup name="fragmentIdentification">
    <attribute name="fragmentId" type="tva:TVAIDType" use="optional"/>
    <attribute name="fragmentVersion" type="unsignedLong" use="optional"/>
</attributeGroup>

<complexType name="CreditsItemType">
    <complexContent>
        <extension base="tva:TVAAgentType">
            <sequence>
            <element name="Character" type="mpeg7:PersonNameType" minOccurs="0"
            maxOccurs="unbounded"/>
            </sequence>
            <attribute name="role" type="mpeg7:termReferenceType" use="required"/>
        </extension>
    </complexContent>
</complexType>
<complexType name="CreditsListType">
    <sequence>
        <element name="CreditsItem" type="tva:CreditsItemType" minOccurs="0"
            maxOccurs="unbounded"/>
    </sequence>
</complexType>
<complexType name="TVAAgentType">
    <sequence>
        <choice minOccurs="0" maxOccurs="unbounded">
            <element name="PersonName" type="mpeg7:PersonNameType"/>
            <element name="PersonNameIDRef">
            <complexType>
                <attribute name="ref" type="tva:TVAIDRefType" use="required"/>
            </complexType>
            </element>
            <element name="OrganizationName" type="mpeg7:TextualType"/>
            <element name="OrganizationNameIDRef">
            <complexType>
                <attribute name="ref" type="tva:TVAIDRefType" use="required"/>
            </complexType>
            </element>
        </choice>
    </sequence>
</complexType>
<simpleType name="TVAIDRefType">
    <restriction base="string">
```

```
        <whiteSpace value="collapse"/>
    </restriction>
</simpleType>
<complexType name="FlagType">
    <attribute name="value" type="boolean" use="required"/>
</complexType>

<complexType name="BitRateType">
    <simpleContent>
        <extension base="nonNegativeInteger">
            <attribute name="variable" type="boolean" use="optional" default="false"/>
            <attribute name="minimum" type="unsignedLong" use="optional"/>
            <attribute name="average" type="unsignedLong" use="optional"/>
            <attribute name="maximum" type="unsignedLong" use="optional"/>
        </extension>
    </simpleContent>
</complexType>

<simpleType name="FrameRateType">
    <restriction base="string">
        <pattern value="([0-9]{1,3}(.[0-9]{1,3})?)|([0-9]{1,3}/1.001)"/>
    </restriction>
</simpleType>
```

Annex B MPEG-7 Datatypes

In this section the datatypes of the ISO/IEC 15938-5 MPEG-7 [4] namespace referenced in the IPDC datatype definitions are listed.

```
    <complexType name="MediaLocatorType">
        <sequence>
            <choice minOccurs="0">
                <element name="MediaUri" type="anyURI"/>
                <element name="InlineMedia" type="mpeg7:InlineMediaType"/>
            </choice>
            <element name="StreamID" type="nonNegativeInteger" minOccurs="0"/>
        </sequence>
    </complexType>

    <complexType name="InlineMediaType">
        <choice>
            <element name="MediaData16" type="hexBinary"/>
            <element name="MediaData64" type="base64Binary"/>
        </choice>
        <attribute name="type" type="mpeg7:mimeType" use="required"/>
    </complexType>

    <simpleType name="mimeType">
        <restriction base="string">
            <whiteSpace value="collapse"/>
        </restriction>
    </simpleType>

    <complexType name="TextualType">
        <simpleContent>
            <extension base="mpeg7:TextualBaseType"/>
        </simpleContent>
    </complexType>

<complexType name="TextualBaseType" abstract="true">
    <simpleContent>
        <extension base="string">
            <attribute ref="xml:lang" use="optional"/>
            <attribute name="phoneticTranscription" use="optional">
                <simpleType>
                    <list itemType="mpeg7:PhoneType"/>
                </simpleType>
            </attribute>
            <attribute name="phoneticAlphabet" type="mpeg7:phoneticAlphabetType" use="optional"
                default="sampa"/>
        </extension>
    </simpleContent>
</complexType>

    <simpleType name="PhoneType">
        <restriction base="string"/>
    </simpleType>

    <simpleType name="phoneticAlphabetType">
        <restriction base="NMTOKEN">
            <enumeration value="sampa"/>
            <enumeration value="ipaSymbol"/>
            <enumeration value="ipaNumber"/>
            <enumeration value="other"/>
```

```
    </restriction>
</simpleType>
```

```
<complexType name="TitleMediaType">
    <sequence>
        <element name="TitleImage" type="mpeg7:ImageLocatorType" minOccurs="0"/>
        <element name="TitleVideo" type="mpeg7:TemporalSegmentLocatorType" minOccurs="0"/>
        <element name="TitleAudio" type="mpeg7:TemporalSegmentLocatorType" minOccurs="0"/>
    </sequence>
</complexType>
```

```
<complexType name="ImageLocatorType">
    <complexContent>
        <extension base="mpeg7:MediaLocatorType">
            <choice minOccurs="0">
            <element name="MediaTimePoint" type="mpeg7:mediaTimePointType"/>
            <element name="MediaRelTimePoint" type="mpeg7:MediaRelTimePointType"/>
            <element name="MediaRelIncrTimePoint" type="mpeg7:MediaRelIncrTimePointType"/>
            <element name="BytePosition">
                <complexType>
                    <attribute name="offset" type="nonNegativeInteger" use="required"/>
                    <attribute name="length" type="positiveInteger" use="optional"/>
                </complexType>
            </element>
            </choice>
        </extension>
    </complexContent>
</complexType>
```

```
<complexType name="MediaTimeType">
    <sequence>
        <choice>
            <element name="MediaTimePoint" type="mpeg7:mediaTimePointType"/>
            <element name="MediaRelTimePoint" type="mpeg7:MediaRelTimePointType"/>
            <element name="MediaRelIncrTimePoint" type="mpeg7:MediaRelIncrTimePointType"/>
        </choice>
        <choice minOccurs="0">
            <element name="MediaDuration" type="mpeg7:mediaDurationType"/>
            <element name="MediaIncrDuration" type="mpeg7:MediaIncrDurationType"/>
        </choice>
    </sequence>
</complexType>

<simpleType name="mediaTimePointType">
    <restriction base="mpeg7:basicTimePointType">
        <pattern value="(\-?\d+(\-\d{2}(\-\d{2})?)?)? (T\d{2}(:\d{2}(:\d{2}(:\d+)?)?)?)?(F\d+)?"/>
    </restriction>
</simpleType>

<complexType name="MediaRelTimePointType">
    <simpleContent>
        <extension base="mpeg7:mediaTimeOffsetType">
            <attribute name="mediaTimeBase" type="mpeg7:xPathRefType" use="optional"/>
        </extension>
    </simpleContent>
</complexType>

<simpleType name="mediaTimeOffsetType">
```

```
    <restriction base="mpeg7:basicDurationType">
      <pattern value="\-?P(\d+D)?(T(\d+H)?(\d+M)?(\d+S)?(\d+N)?)?(\d+F)?"/>
    </restriction>
  </simpleType>

  <complexType name="MediaRelIncrTimePointType">
    <simpleContent>
      <extension base="integer">
        <attribute name="mediaTimeUnit" type="mpeg7:mediaDurationType" use="optional"/>
        <attribute name="mediaTimeBase" type="mpeg7:xPathRefType" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

  <simpleType name="mediaDurationType">
    <restriction base="mpeg7:basicDurationType">
      <pattern value="\-?P(\d+D)?(T(\d+H)?(\d+M)?(\d+S)?(\d+N)?)?(\d+F)?"/>
    </restriction>
  </simpleType>

  <complexType name="MediaIncrDurationType">
    <simpleContent>
      <extension base="integer">
        <attribute name="mediaTimeUnit" type="mpeg7:mediaDurationType" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>

  <simpleType name="basicTimePointType">
    <restriction base="string">
      <pattern value="\-?(\d+(\-\d{2}(\-\d{2})?)?)?(T\d{2}(:\d{2}(:\d{2}(:\d+(\.\d{2})?)?)?)?)?(F\d+)?((\-
        |\+)\d{2}:\d{2})?"/>
    </restriction>
  </simpleType>

  <simpleType name="basicDurationType">
    <restriction base="string">
      <pattern value="\-?P(\d+D)?(T(\d+H)?(\d+M)?(\d+S)?(\d+N)?(\d{2}f)?)?(\d+F)?((\-
        |\+)\d{2}:\d{2}Z)?"/>
    </restriction>
  </simpleType>

  <simpleType name="xPathRefType">
    <restriction base="mpeg7:xPathType">
      <pattern
        value="/?((((child::)?((\i\c*:)?(\i\c*)(\[\d+\])?))|\.|(\.\.))/)*((((child::)?((\i\c*:)?(\i\c*)(\[\d+\])?))|\.)|((attri
        bute::|@)((\i\c*:)?(\i\c*|\*))))"/>
    </restriction>
  </simpleType>

  <simpleType name="xPathType">
    <restriction base="token"/>
  </simpleType>
```

```
  <complexType name="TemporalSegmentLocatorType">
    <complexContent>
      <extension base="mpeg7:MediaLocatorType">
        <choice minOccurs="0">
        <element name="MediaTime" type="mpeg7:MediaTimeType"/>
        <element name="BytePosition">
```

```
                <complexType>
                    <attribute name="offset" type="nonNegativeInteger" use="required"/>
                    <attribute name="length" type="positiveInteger" use="optional"/>
                </complexType>
            </element>
            </choice>
        </extension>
    </complexContent>
</complexType>
```

```
<complexType name="TitleType">
    <simpleContent>
        <extension base="mpeg7:TextualBaseType">
            <attribute name="type" use="optional" default="main">
                <simpleType>
                    <union>
                        <simpleType>
                            <restriction base="NMTOKEN">
                                <enumeration value="main"/>
                                <enumeration value="secondary"/>
                                <enumeration value="alternative"/>
                                <enumeration value="original"/>
                                <enumeration value="popular"/>
                                <enumeration value="opusNumber"/>
                                <enumeration value="songTitle"/>
                                <enumeration value="albumTitle"/>
                                <enumeration value="seriesTitle"/>
                                <enumeration value="episodeTitle"/>
                            </restriction>
                        </simpleType>
                        <simpleType>
                            <restriction base="mpeg7:termReferenceType"/>
                        </simpleType>
                    </union>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
```

```
    <simpleType name="termReferenceType">
        <union>
            <simpleType>
                <restriction base="NMTOKEN">
                <whiteSpace value="collapse"/>
                <pattern value=":[^:]+:[^:]+"/>
                </restriction>
            </simpleType>
            <simpleType>
                <restriction base="anyURI"/>
            </simpleType>
        </union>
    </simpleType>
```

```
    <complexType name="PersonNameType">
        <sequence>
            <choice maxOccurs="unbounded">
                <element name="GivenName" type="mpeg7:NameComponentType"/>
```

```
            <element name="FamilyName" type="mpeg7:NameComponentType" minOccurs="0"/>
            <element name="Title" type="mpeg7:NameComponentType" minOccurs="0"/>
            <element name="Numeration" type="string" minOccurs="0"/>
        </choice>
    </sequence>
    <attribute name="dateFrom" type="mpeg7:timePointType" use="optional"/>
    <attribute name="dateTo" type="mpeg7:timePointType" use="optional"/>
    <attribute name="type" use="optional">
        <simpleType>
            <restriction base="NMTOKEN">
            <enumeration value="former"/>
            <enumeration value="variant"/>
            <enumeration value="main"/>
            </restriction>
        </simpleType>
    </attribute>
    <attribute ref="xml:lang" use="optional"/>
</complexType>

<simpleType name="timePointType">
    <restriction base="mpeg7:basicTimePointType">
        <pattern value="(\-?\d+(\-\d{2}(\-\d{2})?)?)?(T\d{2}(:\d{2}(:\d{2}(:\d+)?)?)?)?(F\d+)?((\-
            |\+)\d{2}:\d{2})?"/>
    </restriction>
</simpleType>

<complexType name="NameComponentType">
    <simpleContent>
        <extension base="mpeg7:TextualBaseType">
            <attribute name="initial" type="string" use="optional"/>
            <attribute name="abbrev" type="string" use="optional"/>
        </extension>
    </simpleContent>
</complexType>
```

```
<complexType name="ParentalGuidanceType">
    <sequence>
        <choice>
            <element name="ParentalRating" type="mpeg7:ControlledTermUseType"/>
            <element name="MinimumAge" type="nonNegativeInteger"/>
        </choice>
        <element name="Region" type="mpeg7:regionCode" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
</complexType>
```

```
<complexType name="ControlledTermUseType">
    <complexContent>
        <extension base="mpeg7:InlineTermDefinitionType">
            <attribute name="href" type="mpeg7:termReferenceType" use="required"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="InlineTermDefinitionType" abstract="true">
    <sequence>
        <element name="Name" minOccurs="0" maxOccurs="unbounded">
            <complexType>
            <simpleContent>
                <extension base="mpeg7:TextualType">
```

```
                    <attribute name="preferred" type="boolean" use="optional"/>
                </extension>
            </simpleContent>
            </complexType>
        </element>
        <element name="Definition" type="mpeg7:TextualType" minOccurs="0"
            maxOccurs="unbounded"/>
        <!-- Term element removed -->
    </sequence>
</complexType>
```

```
<simpleType name="regionCode">
    <restriction base="string">
        <whiteSpace value="collapse"/>
        <pattern value="[a-zA-Z]{2}(-[a-zA-Z0-9]{1,3})?"/>
    </restriction>
</simpleType>
```

```
<complexType name="ExtendedLanguageType">
    <simpleContent>
        <extension base="language">
            <attribute name="type" use="optional" default="original">
            <simpleType>
                <restriction base="NMTOKEN">
                    <enumeration value="original"/>
                    <enumeration value="dubbed"/>
                    <enumeration value="background"/>
                </restriction>
            </simpleType>
            </attribute>
            <attribute name="supplemental" type="boolean" use="optional" default="false"/>
        </extension>
    </simpleContent>
</complexType>
```

```
<complexType name="ClassificationSchemeBaseType" abstract="true">
    <complexContent>
        <extension base="mpeg7:DSType">
            <sequence>
            <element name="Import" type="mpeg7:ReferenceType" minOccurs="0"
            maxOccurs="unbounded"/>
            </sequence>
            <attribute name="uri" type="anyURI" use="required"/>
            <attribute name="domain" use="optional">
            <simpleType>
                <list itemType="mpeg7:xPathAbsoluteSelectorType"/>
            </simpleType>
            </attribute>
        </extension>
    </complexContent>
</complexType>

<complexType name="ClassificationSchemeAliasType">
    <complexContent>
        <extension base="mpeg7:HeaderType">
            <attribute name="alias" type="NMTOKEN" use="required"/>
            <attribute name="href" type="anyURI" use="required"/>
        </extension>
```

```
    </complexContent>
</complexType>

<!-- Definition of ClassificationScheme DS (ISO/IEC 15938-5: 7.3.2) -->
<complexType name="ClassificationSchemeType">
    <complexContent>
        <extension base="mpeg7:ClassificationSchemeBaseType">
            <sequence>
            <element name="Term" type="mpeg7:TermDefinitionType" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>

<complexType name="DSType" abstract="true">
    <complexContent>
        <extension base="mpeg7:Mpeg7BaseType">
            <sequence>
            <element name="Header" type="mpeg7:HeaderType" minOccurs="0"
            maxOccurs="unbounded"/>
            </sequence>
            <attribute name="id" type="ID" use="optional"/>
            <attributeGroup ref="mpeg7:timePropertyGrp"/>
            <attributeGroup ref="mpeg7:mediaTimePropertyGrp"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="Mpeg7BaseType" abstract="true">
    <complexContent>
        <restriction base="anyType"/>
    </complexContent>
</complexType>

<attributeGroup name="timePropertyGrp">
    <attribute name="timeBase" type="mpeg7:xPathRefType" use="optional"/>
    <attribute name="timeUnit" type="mpeg7:durationType" use="optional"/>
</attributeGroup>

<simpleType name="durationType">
    <restriction base="mpeg7:basicDurationType">
        <pattern value="\-?P(\d+D)?(T(\d+H)?(\d+M)?(\d+S)?(\d+N)?)?(\d+F)?((\-|\+)\d{2}:\d{2}Z)?"/>
    </restriction>
</simpleType>

<attributeGroup name="mediaTimePropertyGrp">
    <attribute name="mediaTimeBase" type="mpeg7:xPathRefType" use="optional"/>
    <attribute name="mediaTimeUnit" type="mpeg7:mediaDurationType" use="optional"/>
</attributeGroup>

<complexType name="HeaderType" abstract="true">
    <complexContent>
        <extension base="mpeg7:Mpeg7BaseType">
            <attribute name="id" type="ID" use="optional"/>
        </extension>
    </complexContent>
</complexType>

<simpleType name="xPathAbsoluteSelectorType">
    <restriction base="mpeg7:xPathSelectorType">
```

```
        <pattern
            value="(/|(((//)|/)(((child::)?((\i\c*:)?(\i\c*|\*)))|\.))*)(\|(/|(((//)|/)(((child::)?((\i\c*:)?(\i\c*|\*)))|\.))*))*"/>
    </restriction>
</simpleType>

<simpleType name="xPathSelectorType">
    <restriction base="mpeg7:xPathType">
        <pattern
            value="(/|(((//)|/)?(((child::)?((\i\c*:)?(\i\c*|\*)))|\.|\.\.)(((//)|/)(((child::)?((\i\c*:)?(\i\c*|\*)))|\.|\.\.))*))(\|/|(((//)|/
            )?(((child::)?((\i\c*:)?(\i\c*|\*)))|\.|\.\.)(((//)|/)(((child::)?((\i\c*:)?(\i\c*|\*)))|\.|\.\.))*)*"/>
    </restriction>
</simpleType>

<complexType name="ReferenceType">
    <attributeGroup ref="mpeg7:referenceGrp"/>
</complexType>

<attributeGroup name="referenceGrp">
    <attribute name="idref" type="IDREF" use="optional"/>
    <attribute name="xpath" type="mpeg7:xPathRefType" use="optional"/>
    <attribute name="href" type="anyURI" use="optional"/>
</attributeGroup>

<complexType name="TermDefinitionBaseType" abstract="true">
    <complexContent>
        <extension base="mpeg7:DSType">
            <sequence>
            <element name="Name" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                    <simpleContent>
                        <extension base="mpeg7:TextualType">
                            <attribute name="preferred" type="boolean" use="optional"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="Definition" type="mpeg7:TextualType" minOccurs="0"
            maxOccurs="unbounded"/>
            </sequence>
            <attribute name="termID" type="NMTOKEN"/>
        </extension>
    </complexContent>
</complexType>

<complexType name="TermDefinitionType">
    <complexContent>
        <extension base="mpeg7:TermDefinitionBaseType">
            <sequence>
            <element name="Term" minOccurs="0" maxOccurs="unbounded">
                <complexType>
                    <complexContent>
                        <extension base="mpeg7:TermDefinitionType">
                            <attribute name="relation" type="mpeg7:termRelationQualifierType" use="optional"
            default="NT"/>
                        </extension>
                    </complexContent>
                </complexType>
            </element>
            </sequence>
        </extension>
    </complexContent>
```

```
    </complexType>

    <simpleType name="termRelationQualifierType">
        <union>
            <simpleType>
                <restriction base="NMTOKEN">
                <enumeration value="NT"/>
                <enumeration value="BT"/>
                <enumeration value="RT"/>
                <enumeration value="US"/>
                <enumeration value="UF"/>
                </restriction>
            </simpleType>
            <simpleType>
                <restriction base="mpeg7:termReferenceType"/>
            </simpleType>
        </union>
    </simpleType>
```

Annex C Default Classification Schemes

In this section an initial set of classification schemes is listed.

C.1 ServiceType

The ServiceType fragment defines an element ‘ServiceType’ which is of type tva:ControlledTermType. It is proposed to use a ClassificationScheme as shown below for the ServiceType:

```
<ClassificationScheme uri="urn:dvb:ipdc:esg:cs:ServiceTypeCS">
<Term termID="1.0">  <Name xml:lang="en">ContentType</Name>
        <Definition xml:lang="en">Digital TV service </Definition>
        <Term termID="1.1">
            <Name xml:lang="en">TV Service</Name>
            <Definition xml:lang="en">Digital TV service </Definition>
        </Term>
        <Term termID="1.2">
            <Name xml:lang="en">Radio Service</Name>
            <Definition xml:lang="en">Digital Radio Service</Definition>
        </Term>
        <Term termID="1.3">
            <Name xml:lang="en">Data Service</Name>
            <Definition xml:lang="en">Generic data service</Definition>
            <Term termID="1.3.1">
                <Name xml:lang="en">DRM Service</Name>
                <Definition xml:lang="en">Entitilements/Rights Objects for protected content</Definition>
                <Term termID="1.3.1.1">
                    <Name xml:lang="en">RI Service</Name>
                    <Definition xml:lang="en">18c Rights Issuer service</Definition>
                </Term>
                <Term termID="1.3.1.2">
                    <Name xml:lang="en">EMM Service</Name>
                    <Definition xml:lang="en">OpenFramework EMM service</Definition>
                </Term>
            </Term>
            <Term termID="1.3.2">
                <Name xml:lang="en">download service</Name>
```

```
            <Definition xml:lang="en">File download service </Definition>
        </Term>
    </Term>
</Term>
<Term termID="2.0">  <Name xml:lang="en">TransportType</Name>
    <Definition xml:lang="en">Type of transport, e.g. download, streamed</Definition>
    <Term termID="2.1">
        <Name xml:lang="en">Streamed</Name>
        <Definition xml:lang="en">A streamed service </Definition>
    </Term>
    <Term termID="2.2">
        <Name xml:lang="en">Download</Name>
        <Definition xml:lang="en">A download service </Definition>
    </Term>
</Term>

 </ClassificationScheme>
```

C.2 ZappingSupport

In this classification scheme types of zapping data are declared.

```
<ClassificationScheme uri="urn:dvb:ipdc:cs:ZappingSupportCS:2005">
    <Term termID="1">
    <Name xml:lang="en">Static Zapping Support</Name>
        <Term termID="1.1">
            <Name xml:lang="en">Static Picture</Name>
<Definition xml:lang="en"> This static picture shall be defined in the MediaLocator element of the
                ZappingSupportType.
            </Definition>
        </Term>
        <Term termID="1.2">
            <Name xml:lang="en">Static Text</Name>
<Definition xml:lang="en"> This static text shall be defined in the MediaLocator element of the
                ZappingSupportType.
</Definition>
        </Term>
    </Term>
    <Term termid="2">
    <Name xml:lang="en">Dynamic Zapping Support</Name>
        <Term termID="2.1">
            <Name xml:lang="en">Video</Name>
<Definition xml:lang="en"> This is a reduced copy of the related video track of an AV service. This zapping
                type is transported with the same protocol stack and restrictions as specified for the associated
                service:
                IP, UDP, RTP with RTP payload format video, the streamed video content as payload. For
                synchronization purposes the same RTP timestamps as the related video track should be used.
            </Definition>
        </Term>
<Term termID="2.2">
            <Name xml:lang="en">Audio</Name>
<Definition xml:lang="en"> This is a reduced copy of the related audio track of an AV service. This zapping
                type is transported with the same protocol stack and restrictions as specified for the associated
                service:
```

```
                IP, UDP, RTP with RTP payload format audio, the streamed audio content as payload. For
                synchronization purposes the same RTP timestamps as the related audio track should be used.
            </Definition>
        </Term>
        <Term termID="2.3">
            <Name xml:lang="en">DynamicPicture</Name>
<Definition xml:lang="en"> This zapping type is transported with the following protocol layers:
                IP, UDP, RTP with RTP payload format MJPEG as specified in IETF RFC 2435, JPEG file as
                payload. For synchronization purposes the same RTP timestamps as the related video track
                should be used.
            </Definition>
        </Term>
        <Term termID="2.4">
            <Name xml:lang="en">DynamicText</Name>
<Definition xml:lang="en"> This zapping type is transported with the following protocol layers:
                IP, UDP, RTP with RTP payload format 3GPP timed text as specified in 3GPP TS 26.346
                v6.1.0: Multimedia Broadcast/Multicast Service; Protocols and Codecs (Release 6), 3GPP timed
                text file as payload. For synchronization purposes the same RTP timestamps as the related
                video track should be used.
            </Definition>
        </Term>
    </Term>
</ClassificationScheme>
```

Annex D Extensibility of the ESG Schema

The following clauses define rules to support specification extensions and private extensions of the *ESG Schema* in a backward and forward compatible manner.

The following rules shall be applied so as to define valid extensions of the *ESG Schema*, in particular to allow the compatibility mechanisms described above. They constrain the extensibility of the *ESG schemas*:

- Extension must be defined using the *ESG schema* representation language (i.e. MPEG-7 DDL). The way these extended schemas are transmitted is out of the scope of the present document.
- The module definition must have a prose definition that describes the syntactic and semantic requirements of the elements, attributes and/or content models that it declares.
- Existing element names should never be re-used. New elements names should be defined under their own namespace (e.g. for another version of the ESG specification or for private extensions).
- The module definition's elements and attributes must be part of an XML namespace. If the module is defined by an organization other than DVB, TVA and MPEG (for imported MPEG datatypes and description schemes), this namespace must NOT be the same as the namespace in which other DVB, TVA and MPEG standards are defined.
- The namespace under which extensions are defined will need to be clearly identified.
- Any extensions to existing schema should not obscure existing functionality. Thus existing functionality should not be contained within a new element that an earlier decoder will not understand.
- Wherever possible, an extended schema should only add functionality and not replace existing functionality. This will allow a version 1 decoder to maximally understand a version 2 document.
- An application should ignore any elements or attributes they do not need, do not understand or cannot use.

Table D.1 provides the list of conditions under which the extensions of ESG metadata definitions are supported or not.

Table D.1: Types of extension permitted in future versions of the *ESG data model*

Condition/Type of extension of *ESG* metadata definitions	Status
Condition 1: A new global element of existing type	NOT PERMITTED
Condition 2: A new global attributes added to existing type	PERMITTED
new type (simple or complex - but see below for limitations on derivation etc)	PERMITTED
Polymorphism of existing type by Inheritance with restriction	PERMITTED (But see rules above)
Polymorphism of existing type by Inheritance with extension	PERMITTED (But see rules above)
Polymorphism of existing type by Redefining types during import	NOT PERMITTED
Substitution Groups	NOT PERMITTED. Instead of using substitution groups, explicit derivation can be used. This is safer for future extensions.

Annex E ESG Init Message

E.1 Default ESG Init Message (informative)

The following (Table E.1) is an example ESG Init Message that conforms to the BiM profile specified in this present document.

Table E.1: Example ESG Init Message

Field	No. of bits	Value	Notes
ESG Init Message {			
EncodingVersion	8	'0xF1'	DVB profile of BiM
IndexingFlag	1	0	No indexing used in the current ESG Fragment Stream
reserved	7	1111111	
DecoderInitptr	8	5	Position of the DecoderInit data from the beginning of the ESG init message
{			/* EncodingVersion == '0xF0'*/
BufferSizeFlag	1	0	default buffer size for the ZlibCodec is used
PositionCodeFlag	1	0	position codes are not used
reserved	6	111111	
CharacterEncoding	8	'0x01'	UTF-8 Character Encoding
}			
Reserved	0 or 8+		
DecoderInit( )	8+	[data]	Decoder Initialisation message
}			

The following (Table E.2) is an example ESG Init Message when textual XML representation is used with gzip encoding specified in this document.

Table E.2: Example ESG Init Message

Field	No. of bits	Value	Notes
ESG Init Message {			
EncodingVersion	8	'0xF2'	GZip encoding
IndexingFlag	1	0	No indexing used in the current ESG Fragment Stream
reserved	7	1111111	
DecoderInitptr	8	5	Position of the DecoderInit data from the beginning of the ESG init message
{			/* EncodingVersion == '0xF2' || EncodingVersion == '0xF3'*/
CharacterEncoding	8	'0x01'	UTF-8 Character Encoding
}			
Reserved	0		
DecoderInit( )	8+	[data]	Decoder Initialisation message
}			

E.2 Example of a DecoderInit message (informative)

The following (Table E.3) is an example DecoderInit message that conforms to the BiM profile specified in this document.

Table E.3: Example DecoderInit message

Field	No. of bits	Value	Semantic
DecoderInit {			
SystemsProfileLevelIndication	16	'0x80'	arbitrary value
UnitSizeCode	3	000	default unit size
ReservedBits	5	11111	
NumberOfSchemas	8	'0x01'	Only one schema is used.
{			
SchemaURI_Length[0]	8	'0x10'	16 characters in the URI string
SchemaURI [0]		"urn:dvb:ipdc:esg:2005"	
LocationHint_Length[0]	8	'0x00'	no location hint is provided.
NumberOfTypeCodecs[0]	8	'0x00'	Only default codecs are used.
}			
InitialDescription_Length	8	'0x00'	The initial root description is conveyed in the TVAMain fragment.
}			

E.3 Example of a DecoderInit message (informative)

The following (table E.4) is an example DecoderInit message that conforms to the BiM profile specified in this document including an instantiation of a ContextPathTable.

Table E.4: Example DecoderInit message

Field	No. of bits	Value	Semantic
DecoderInit {			
SystemsProfileLevelIndication	16	'0x0080'	arbitrary value
UnitSizeCode	3	000	default unit size
NoAdvancedFeatures	1	0	Advanced Features configured.
ReservedBits	4	11111	
AdvancedFeaturesFlag_Length	8	‘0x08’	Eight Flags signaled
ReservedBitsZero	7	0000000	
ContextPathTableFlag	1	1	A ContextPathTable is signaled
NumberOfSchemas	8	'0x01'	Only one schema used
{			
SchemaURI_Length[0]	8	'0x10'	16 characters in the URI string
SchemaURI [0]		"urn:dvb:ipdc:esg:2005"	
LocationHint_Length[0]	8	'0x00'	no location hint is provided.
NumberOfTypeCodecs[0]	8	'0x00'	Only default codecs are used.
}			
{			
ContextPathTable_Length	8	‘0x0F’	
ContextPathCode_Length	8	‘0x10’	The ContextPathCode is 16bits long
NumberOfContextPaths	8	‘0x02’	Two ContextPaths are Signaled
CompleteContextPath	1	0	
ContextPath_Length[0]	10	1000101100	
ContextPath[0]	28	…	first ContextPath
ContextPathCode[0]	16	‘0x0101’	
ContextPath_Length[1]	10	1001000000	
ContextPath[1]	32	…	second ContextPath
ContextPathCode[1]	16	‘0x0102’	
ReservedBitsZero	3	000	Stuffing
}			
InitialDescription_Length	8	'0x00'	The initial root description is conveyed in the TVAMain fragment.
}			