ETSI TS 102 471 V1.3.1 (2009-04)

Technical Specification

Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Electronic Service Guide (ESG)

European Broadcasting Union

Union Européenne de Radio-Télévision

EBU·UER

Reference

RTS/JTC-DVB-248

Keywords

broadcasting, data, digital, DVB, IP, mobile, TV, video

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

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

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

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

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

Copyright Notification

No part may be reproduced except as authorized by written permission.
The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 2009.
© European Broadcasting Union 2009.
All rights reserved.

DECTTM, **PLUGTESTS**TM, **UMTS**TM, **TIPHON**TM, the TIPHON logo and the ETSI logo are Trade Marks of ETSI registered for the benefit of its Members.
3GPPTM is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners.
LTE™ is a Trade Mark of ETSI currently being registered
for the benefit of its Members and of the 3GPP Organizational Partners.
GSM® and the GSM logo are Trade Marks registered and owned by the GSM Association.

Contents

Intellectual Property Rights

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

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

Foreword

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

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

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

Founded in September 1993, the DVB Project is a market-led consortium of public and private sector organizations in the television industry. Its aim is to establish the framework for the introduction of MPEG-2 based digital television services. Now comprising over 200 organizations from more than 25 countries around the world, DVB fosters market-led systems, which meet the real needs, and economic circumstances, of the consumer electronics and the broadcast industry.

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

The present document contains information about the Electronic Service Guide (ESG) which describes available services. 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 (EN 302 304 [1]).

2 References

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.
- Non-specific reference may be made only to a complete document or a part thereof and only in the following cases:
  - if it is accepted that it will be possible to use all future changes of the referenced document for the purposes of the referring document;
  - for informative references.

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

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

2.1 Normative references

The following referenced documents are indispensable for the application of the present document. For dated references, only the edition cited applies. For non-specific references, the latest edition of the referenced document (including any amendments) applies.

[1] ETSI EN 302 304: "Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)".

[2] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types".

[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] ETSI TS 102 472: "Digital Video Broadcasting (DVB);IP Datacast over DVB-H: Content Delivery Protocols".

[6] ETSI TS 102 005: "Digital Video Broadcasting (DVB); Specification for the use of Video and Audio Coding in DVB services delivered directly over IP protocols".

[7] IETF RFC 1952 (1996): "GZIP file format specification Version 4.3".

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

[9] IETF RFC 3450: "Asynchronous Layered Coding (ALC) Protocol Instantiation".

[10] IETF RFC 3926: "FLUTE - File Delivery over Unidirectional Transport".

[11] IETF RFC 3451: "Layered Coding Transport (LCT) Building Block".

[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"); Part 3: Metadata; Sub-part 1: Phase 1 - Metadata schemas".

[13] IANA: "Internet Multicast Addresses".

NOTE: See at http://www.iana.org/assignments/multicast-addresses.

[14] IETF RFC 3629: "UTF-8, a transformation format of ISO 10646".

[15] ETSI TS 126 346 (V8.1.0): "Universal Mobile Telecommunications System (UMTS); Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (3GPP TS 26.346, Release 8)".

[16] IEEE 754-1985: "Standard for Binary Floating-Point Arithmetic".

[17] W3C Recommendation (2nd May 2001): "XML Schema".

NOTE: See at http://www.w3.org/XML/Schema.

[18] "DVB CAS System".

NOTE: See at http://www.dvb.org/products_registration/dvb_identifiers/ca_systems/.

[19] IETF RFC 2616: "Hypertext Transfer Protocol - HTTP/1.1".

[20] IANA: "Hypertext Transfer Protocol Parameters".

NOTE: See at http://www.iana.org/assignments/http-parameters.

[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"); Part 3: Metadata; Sub-part 2: System aspects in a uni-directional environment".

[22] IANA: "Port Numbers".

NOTE: See at http://www.iana.org/assignments/port-numbers.

[23] IANA: "Internet Protocol Version 6 Multicast Addresses".

NOTE: See at http://www.iana.org/assignments/ipv6-multicast-addresses.

[24] Void.

[25] ETSI EN 300 468: "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems".

[26] IETF RFC 2327: "SDP: Session Description Protocol".

[27] ISO 4217: "Codes for the representation of currencies and funds".

NOTE: See at http://www.iso.org/iso/en/prods-services/popstds/currencycodes.html.

[28] ISO/IEC 10918-1: "Information technology - Digital compression encoding of continuous-tone still images: Requirements and guidelines".

NOTE: See at http://www.w3.org/Graphics/JPEG/itu-t81.pdf.

[29] JFIF: "JPEG File Interchange Format", Eric Hamilton, C-Cube Microsystems.

NOTE: See at http://www.w3.org/Graphics/JPEG/jfif3.pdf.

[30] W3C Recommendation: "PNG (Portable Network Graphics) Specification Version 1.0".

NOTE: See at http://www.w3.org/TR/REC-png.html.

[31] GIF 89a Specification: "Graphics Interchange Format (sm)", Version 89a, (c) 1987, 1988, 1989, 1990, Copyright CompuServe Incorporated Columbus, Ohio.

NOTE: See at http://www.w3.org/Graphics/GIF/spec-gif89a.txt.

[32] ETSI TS 126 234 (V8.1.0): "Universal Mobile Telecommunications System (UMTS); Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (3GPP TS 26.234, Release 8)".

[33] ETSI TS 102 832: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Notification Framework".

[34] DVB identifiers.

NOTE: See at http://www.dvb.org/products_registration/dvb_identifiers/index.xml.

[35] W3C Recommendation: "HTML 4.01 Specification".

NOTE: See at http://www.w3.org/TR/html401/.

[36] IETF RFC 3987: "Internationalized Resource Identifiers (IRIs)".

NOTE: See at http://www.ietf.org/rfc/rfc3987.txt.

[37] IETF RFC 2782: "A DNS RR for specifying the location of services (DNS SRV)".

NOTE: See at http://www.ietf.org/rfc/rfc2782.txt.

[38] "Enabler Release Definition for OMA Device Management v1.2", OMA-ERELD-DM-V1-2-1.

NOTE: See at http://www.openmobilealliance.org/.

2.2 Informative references

The following referenced documents are not essential to the use of the present document but they assist the user with regard to a particular subject area**.** For non-specific references, the latest version of the referenced document (including any amendments) applies.

[i.1] ETSI TR 102 469: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Architecture".

[i.2] IETF RFC 2435: "RTP Payload Format for JPEG-compressed Video".

[i.3] ETSI TS 126.234: "Universal Mobile Telecommunications System (UMTS); LTE; Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs (3GPP TS 26.234 version 8.1.0 Release 8)".

[i.4] IETF RFC 2326: "Real Time Streaming Protocol (RTSP)".

[i.5] ETSI TS 102 611: "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Implementation Guidelines for Mobility".

[i.6] ISO 3166-1: "Codes for the representation of names of countries and their subdivision; Part 1: Country codes".

[i.7] ISO 3166-2: "Codes for the representation of names of countries and their subdivisions; Part 2: Country subdivision code".

[i.8] ETSI TS 102 034: "Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks".

[i.9] ETSI TS 123.003: "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); LTE; Numbering, addressing and identification (3GPP TS 23.003 version 8.3.0 Release 8)".

3 Definitions, mnemonics, functions and abbreviations

3.1 Definitions

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

AccessPoint: "reference point" of an interactive server or broadcast FLUTE session that is used by terminals to retrieve the ESG Components

NOTE: The AccessPoints are signalled in the ESG Bootstrap and DeliveryList.

BC ESG Init Container: ESG Container carrying initialization information to decode ESG Fragments delivered over Broadcast Channel

BC ESG Init Message: ESG Init Message specific to broadcast delivery

Broadcast Channel: distributes data unidirectionally to all receivers

NOTE: In the present document a Broadcast Channel distributes data over DVB-H network.

Channel: IP connection or a bundle of IP connections between the network and the terminal

NOTE: In the scope of the present document, a Channel can be either a Broadcast Channel or Interactive Channel.

complete ESG: full self-consistent set of fragments available at a certain point in time which can be used to construct a self-contained ESG at the terminal

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

Context Path: in XML terminology, identifies the context of an element and its datatype in an XML Instance by describing the path from the root element to that element

NOTE: The context of an element is defined in the present document by all parent elements of that element and their datatypes.

Current ESG XML Document: instantiation of the ESGMain Element together with all ESG XML Fragments transported at a particular point in time

Current ESG: consistent set of ESG Fragments at a particular point in time

datatype: XML Schema datatype [17]

DeliveryList: consists of one or more SubLists

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

Encapsulated ESG XML Fragment: representation of an ESG XML Fragment, which also contains header information (e.g., the Context Path Code of the ESG XML Fragment)

ESG Auxiliary Data: ESG data, which is referenced from an instance of the XML based Data Model

EXAMPLE: SDP file, HTML page or PNG file.

ESG Component: set of fragments that builds part of a complete ESG

NOTE: A complete ESG can be built of several ESG Components, which may be delivered over different channels.

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

ESG Fragment Stream: stream of ESG Fragments which contributes to the same ESG on receiver end

NOTE: With respect to the transport layer this ESG Fragment stream can be compiled from several transport Streams, e.g. IP Streams.

ESG Fragment Type: category of ESG Fragment

EXAMPLE: ESG XML Fragment, ESG Auxiliary Data or Private Auxiliary Data.

ESG Fragment: fragment of ESG data

NOTE: According to the present document, an ESG Fragment can be ESG XML Fragment, ESG Auxiliary Data or Private Auxiliary data.

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

ESG Init Message: initialization information to decode ESG Fragments

ESG Provider: provider that offers an ESG

ESG XML Fragment Type: category of ESG XML Fragments

NOTE: 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 related datatypes are declared in table 3 or in the DecoderInit (see clause 6.2).

ESG XML Fragment: ESG Fragment of an XML instance which is an instantiation of a datatype

NOTE: A limited set of ESG XML Fragment Types have been defined in the present document.

Fragment Reference: 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.

IA ESG Init Container: ESG Container carrying initialization information to decode ESG Fragments delivered over Interactive Channel

NOTE: The ESG fragments, which are delivered over Interactive channel due to ESG repair, would need BC ESG Init Container to decode.

IA ESG Init Message: ESG Init Message specific to interactive delivery

Interactive Channel: distributes data bidirectionally to each receiver

IPDCKMSId: id which is assigned by DVB to every Key Management System

NOTE: The IPDCKMSId is referred to as CA_System_ID in the DVB registration [18] and [39].

IPDCOperator: characterized uniquely by IPDCOperatorId and IPDCKMSId

NOTE: A physical IPDC Operator may have several IPDCOperatorIds which each map to an IPDC Operator in the defined sense.

legacy terminal: terminals which comply to versions of ESG specifications published by DVB before the version of the present document

Private Auxiliary Data: data of which the format is not specified in the present document

service: offer from a service provider which has media content related to it

SubList: list of ESG Fragments

NOTE: The combination of SubLists describes the full set of Fragments a terminal needs to build a complete and consistent ESG instance.

3.2 Mnemonics

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

bslbf: Bit string, left bit first, where "left" is the order in which bit strings are written in the present document. 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 1.

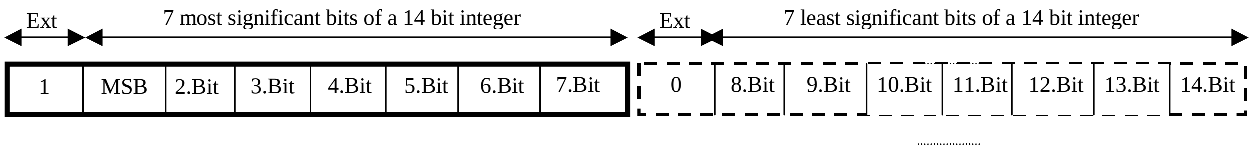

Ext
7 most significant bits of a 14 bit integer
Ext
7 least significant bits of a 14 bit integer
1
MSB
2.Bit
3.Bit
4.Bit
5.Bit
6.Bit
7.Bit
0
8.Bit
9.Bit
10.Bit
11.Bit
12.Bit
13.Bit
14.Bit

Figure 1: Informative example for the vluimsbf8 data type

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 2.

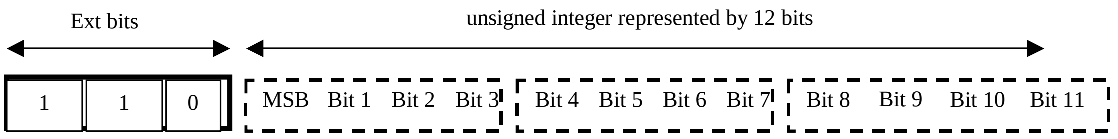


Figure 2: Informative example for the vluimsbf5 data type

3.3 Functions

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

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: 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	3rd Generation Partnership Project
ALC	Asynchronous Layered Coding
BC	Broadcast Channel
BiM	Binary format for Metadata
DL	DeliveryList
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
IA	Interactive Channel
IP	Internet Protocol
IPDC	IP Datacast
KMS	Key Management System
LCT	Layered Coding Transport
MB/MS	Multimedia Broadcast/Multicast Service
MIME	Multipurpose Internet Mail Extensions
PID	Packet IDentifier
PSI	Program Specific Information
RFC	Request For Comments
SDP	Session Description Protocol
SI	Service Information
TOI	Transport Object Identifier (in LCT)
TSI	Transport Session Identifier
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

The ESG operations can be broken down in three main operations:

- ESG bootstrap: the operation through which the terminal knows which ESGs are available and how to acquire them. This operation is described for Broadcast Channel in clause 9.2.1 and Interactive Channel in clause 9.2.2.
- 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. This operation is described in clause 8 and clause 10 for retrieval over broadcast and interactive channel.
- ESG update: the operation through which the terminal refreshes the ESG information stored in the terminal with the latest versions. This operation is described in clause 8.4.

NOTE: Even though the steps are listed sequentially this does not mean that steps cannot be processed in parallel. ESG operations take place after the ESG Bootstrap Entry Point has been discovered as described in clause 9.

4.1.1 ESG Acquisition overview

One of the ESG delivery scenarios is the ESG delivery over Broadcast Channel. In this scenario a whole ESG is delivered over DVB-H bearer utilizing one or several IP streams and using FLUTE protocol as described in clause 8. For the ESGs delivered solely over the BC Channel, observations about consistency of an ESG in clause 8.1.4 apply. Update and versioning are based on Container versioning or Index List.

Note that this scenario can benefit from repair capability provided by an interactive access point as specified in clause 9.1.

An ESG or parts of it can also be delivered over an interactive channel in three scenarios:

a) to repair the containers received over broadcast;

b) to complement the broadcast channel;

c) as a standalone means for transporting a consistent ESG.

In the repair case, the terminal can request for containers that are transmitted over the Broadcast Channel by using their Container IDs, as described in clause 10.5.2. The requests are to be sent to the interactive access point supporting "repair" capability.

It is possible to deliver parts of an ESG over different channels, i.e. over BC and IA Channels. In this complementary case a comprehensive list of the ESG Fragments are listed in the DeliveryList, i.e. at a given time the listed fragments form a consistent ESG. Update and versioning management is handled by tracking changes in the DeliveryList. The repair of the ESG can be performed to its BC delivered parts as in the BC-only delivery case. See figure 3 for examples of these scenarios.

The third scenario is a standalone IA delivery of an ESG. Also in this case the DeliveryList provides a list of consistent set of ESG Fragments. The repair, as specified for the BC delivery, does not apply.

The interactive delivery methods utilizing HTTP protocol are described in more detail in clause 10.

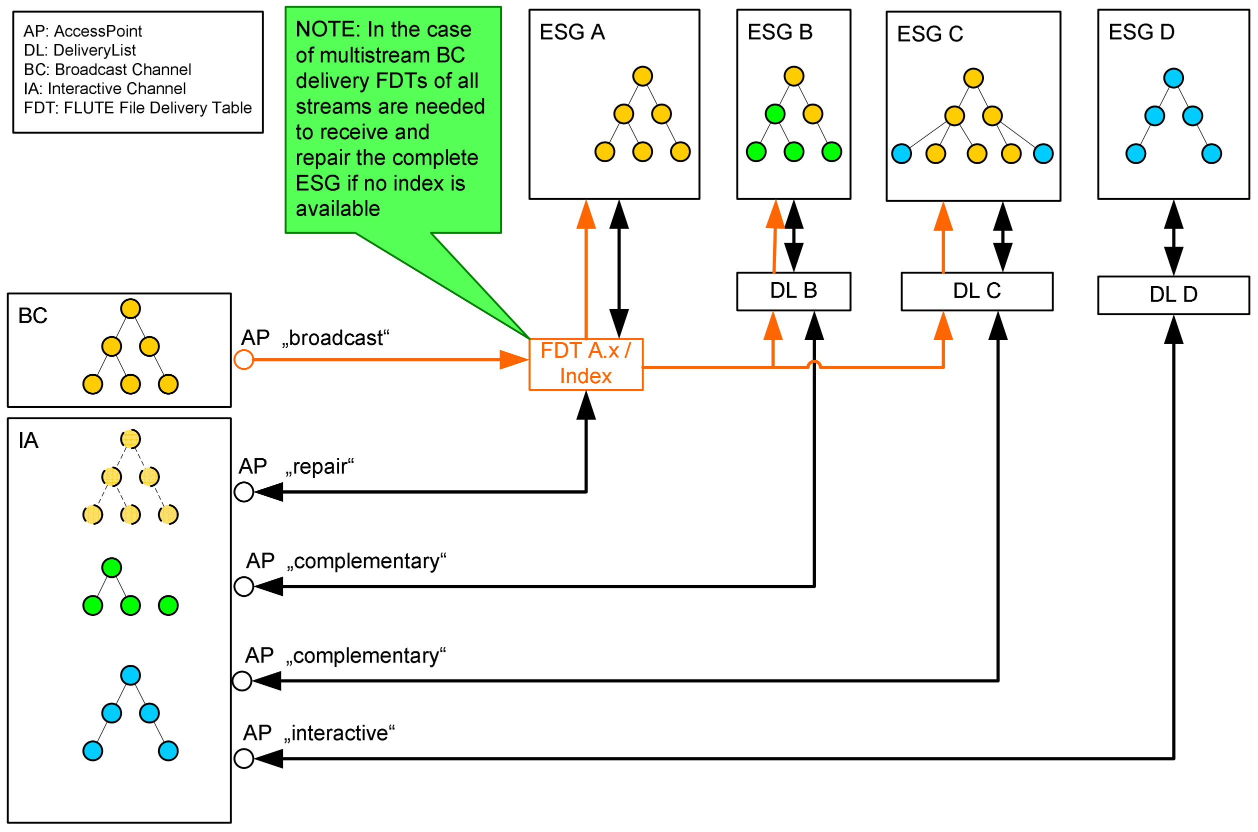


Figure 3: Examples of ESG retrieval over broadcast and interactive channels

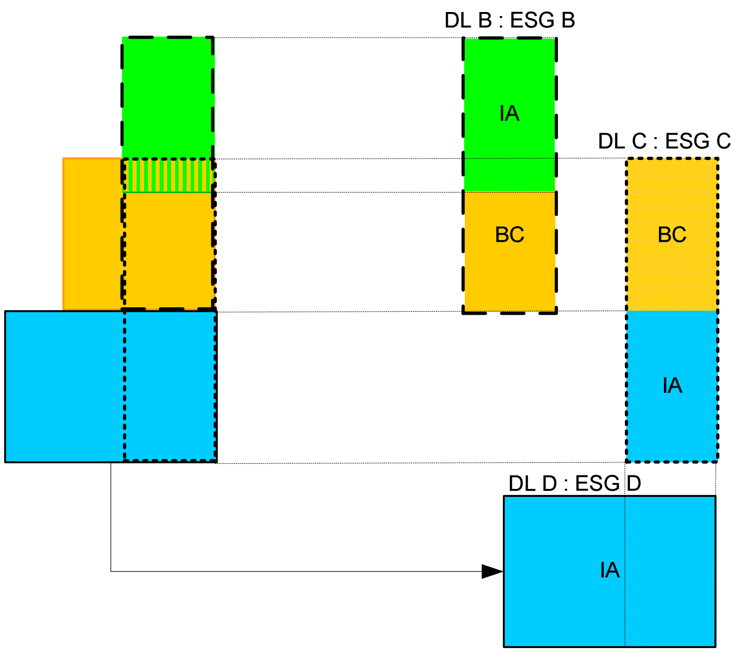


Figure 4: Examples of how an ESG can be build from Fragment sets of various sources

An ESG Provider may offer one or more ESGs. An ESG, uniquely identified by its ESG_URI, is built from ESG Fragments available over one or more channels. For each ESG delivered (partly or fully) over interactive channel, a DeliveryList signals the Fragments that have to be received from each channel. Some Fragments offered by an ESG Provider may be used in multiple ESGs for example broadcast fragments may be complemented by two different Providers or fragments of the same provider maybe used as complements in one ESG and as part of a different ESG fully delivered over interactive channel. Figure 4 shows examples how ESGs are built from different fragments sets and signalled in the DeliveryList.

4.2 Service Discovery

The 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. See figure 5 for an overview of the service discovery.

Based on the ESG information rendered to a user through an ESG application, a specific service can be selected. Both broadcast services and unicast services are supported. The ESG also provides information which enables the terminal to connect to the related IP stream in the DVB-H transport stream or interactive network.

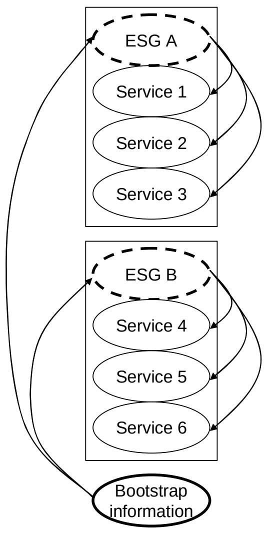


Figure 5: Service Discovery overview

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 and/or by HTTP requests over interactive channel.

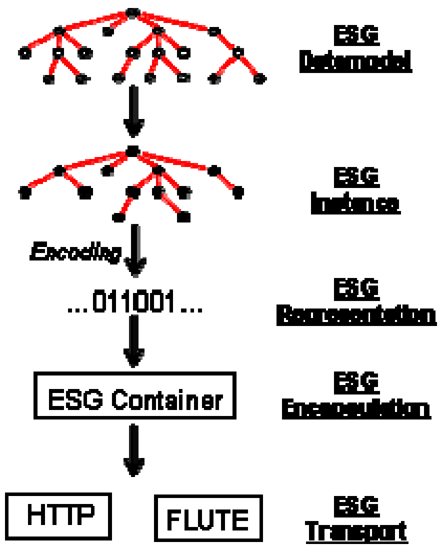


Figure 6

5 ESG Datamodel

5.1 Overview

In this clause 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 7 depicts the ESG Fragments specified in this clause 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.

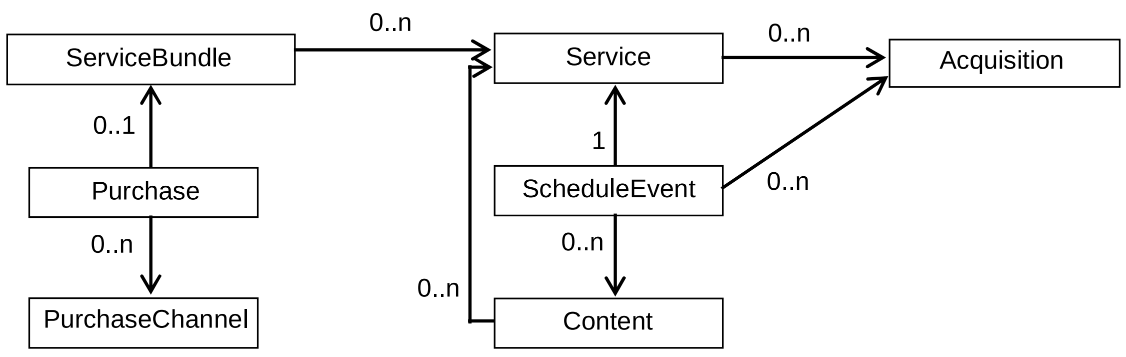


Figure 7: Block diagram of the specified ESG XML Fragments and the relations between them

5.2 ESG Wrapper

In this clause 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 signalled.

5.2.1 ESG Namespace Declaration

In this clause the target namespaces of urn:dvb:ipdc:esg:2005 and urn:dvb:ipdc:esg:2008 are declared. The namespaces contain the data types and the global elements defined respectively declared in the present document. Also the imported namespaces and their namespace aliases used throughout the present document are declared in this clause.

The targetNamespace="urn:dvb:ipdc:esg:2008" applies to the schema extensions defined in the present document. To build a valid ESG instance according to the current document, these extensions, identified by xsi:type, shall be used. Legacy schema definitions of ESG Datamodel Types are included for the sake of better readability and understanding of the present document but remain in the targetNamespace="urn:dvb:ipdc:esg:2005" to enable forward compatibility. The relevant syntax sections will be marked with a comment. For implementation see annex J for the complete schema definitions.

```
<schema targetNamespace="urn:dvb:ipdc:esg:2008"
        xmlns:esg="urn:dvb:ipdc:esg:2005"  xmlns:esg2="urn:dvb:ipdc:esg:2008"
        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:dvb:ipdc:esg:2005" />
   <import namespace="urn:mpeg:mpeg7:schema:2001" />
   <import namespace="urn:tva:metadata:2005" />
```

5.2.2 ESG Main Element

In this clause 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 signalled a default ESG Main Element should be assumed by the decoder as specified in clause 5.2.4.

5.2.2.1 ESG Main Element Syntax

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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:2008.
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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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="esg2:AcquisitionExtensionType" 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	Specifies 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 clause 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.

```
<esg:ESGMain xmlns="urn:dvb:ipdc:esg:2008" xmlns:esg="urn:dvb:ipdc:esg:2005"
            xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:tva="urn:tva:metadata:2005"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="urn:dvb:ipdc:esg:2008
            www.dvb.org/metadata/ipdc/esg2008.xsd">
   <esg:ESG>
      <esg:ContentTable/>
      <esg:ScheduleEventTable/>
      <esg:ServiceTable/>
      <esg:ServiceBundleTable/>
      <esg:PurchaseTable/>
      <esg:PurchaseChannelTable/>
      <esg:AcquisitionTable/>
   </esg:ESG>
</esg:ESGMain>
```

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

5.3 Basic ESG Datatypes

In this clause 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

```
   <!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

   <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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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). When the serviceType is Service related Notification service, it specifies the ID of the referred Service or ScheduleEvent fragment of a regular service.
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.

NOTE: The value of the field HowRelated can use the terms currently available in the classification scheme urn:tva:metadata:cs:HowRelatedCS:2007, e.g. term 24 IsPartOf.

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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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 clause provides an extension point to specify private data. Private data elements in the ESG are supposed to be exposed to external applications. The client terminal should determine the target application using the namespace of the private data (signalled by the xsi:type attribute attached to the private data XML element). The default behaviour in case the namespace is not known by the ESG client application is to ignore the content of the private data element(s). For specific applications, syntax and semantics of this private data are defined in the present document.

5.3.4.1 Private Data Syntax

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

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

5.3.4.2 Private Data Extensions for Notification

The following extension describes the PrivateData specific to Notification applications. The NotificationPrivateData contains information on the ComponentID and NotificationType which can be used by the Notification application to bind Content or Service Description to a specific Notification Component that carries a specific NotificationType. When a user selects notifications described in the Content or Service Fragment (e.g. News Notifications), the application can map the associated ComponentID to the corresponding value in the Acquisition Fragment. The association between Content and Acquisition is made by the sequence of AcquisitionRef and ContentFragmentRef in the ScheduleEvent Fragment, the association between Service and Acquistion fragements is established via AcquisitionRef in the Service Fragment

The following defines the syntax of NotificationPrivateDataType:

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2008" >

<complexType name="NotificationPrivateDataType" mixed="false">
   <complexContent mixed="false">
      <extension base="esg:PrivateDataType">
         <sequence>
            <element name="NotificationBinding">
               <complexType>
                  <sequence>
                     <element name="NotificationType" type="unsignedShort"/>
                     <element name="ComponentID" type="esg2:ComponentIDType"/>
                  </sequence>
               </complexType>
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

The semantics of the newly added fields are as follows:

Field	Semantics
NotificationBinding	Contains information for the Notification application to bind the content or service description to a NotificationType of a NotificationComponent.
NotificationType	The Notification type of the notification messages associated with the Content or Service Fragment.
ComponentID	Unique identifier a Notification component. This identifier shall map to a corresponding ComponentID of the extended ComponentCharacteristic of Acquisition Fragment.

The Notification type of the NotificationPrivateDataType shall map to the NotificationType entry of the NotificationApplicationInit element as described in clause 5.10.7.

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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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 1: 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 characterizes 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 2: 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 clause 5.3.3). EXAMPLE 1: For a TV channel, usually the service provider is the TV channel itself. EXAMPLE 2: 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 3: 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 clause 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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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 1: 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 characterizes the genre of the medias available from the described service bundle.
ServiceRef	Specifies the services contained in the service bundle. NOTE 2: 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 clause 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

```
   <!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

   <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 1: 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 2: 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 3: 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 4: 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 5: 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 clause 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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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	In the case of content over Broadcast Channel, this field signals the URI of the Content-Location, that is used to locate 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. In the case of content over the Interactive Channel, this field is used as a parameter to request the content from a Unicast Access described in the Acquisition Fragment. For the query syntax see clause 7.3.6.1 of CDP [5].
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.

Field	Semantics
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 clause 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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005"-->

<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>

<!-- targetNamespace="urn:dvb:ipdc:esg:2008"-->

<complexType name="PurchaseExtensionType" >
   <complexContent>
      <extension base="esg:PurchaseType" >
         < sequence>
            < element name="privateData" type="esg:PrivateDataType" minOccurs="0"
            maxOccurs="unbounded" />
         </ 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 1: 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. The currency code signalled in the currency attribute is specified in ISO 4217 [27]. NOTE 2: 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 quantization 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 quantization 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. DRMSystems registered by DVB as defined and registered by DVB [18] and [40] are referred to as Key Management Systems and 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 signalled which are specific to the purchase channel.

Field	Semantics
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 3: 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 clause 5.4.2.
PrivateData	Generic element instantiated to add private data, e.g. specific to an SPP profile.

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 the present 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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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 figures 8 and 9 provide an illustration of two use cases.

In figure 8 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".

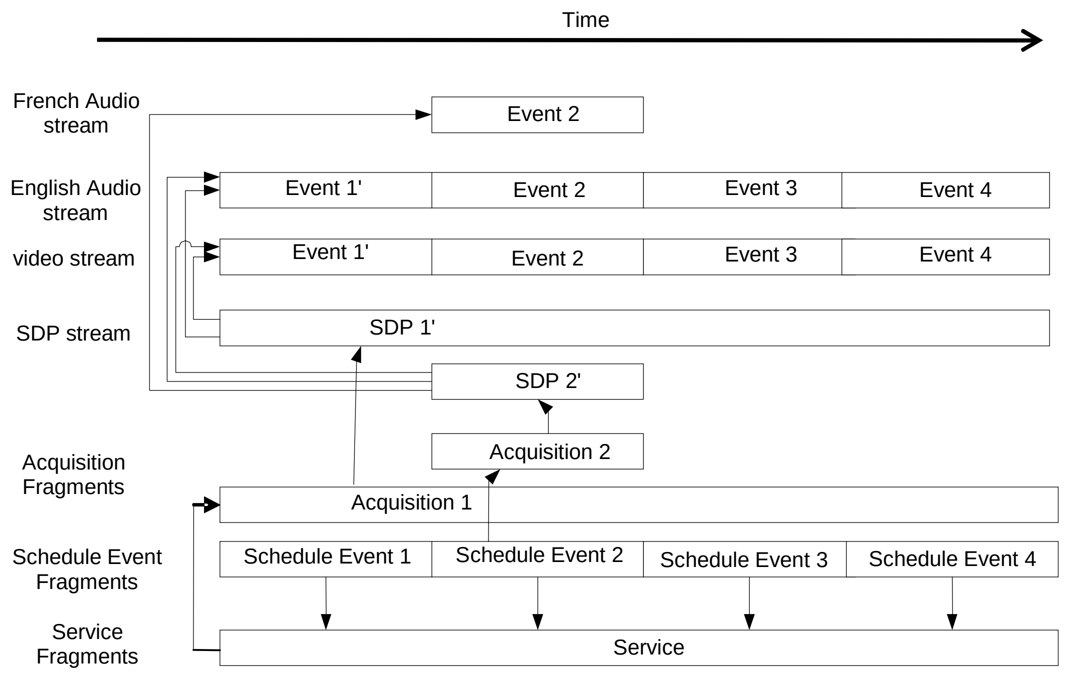


Figure 8: Relationship between described content streams, SDP files, Acquisition Fragments and other ESG XML Fragments

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 9 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 [26]). 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 does not have to load a new SDP file.

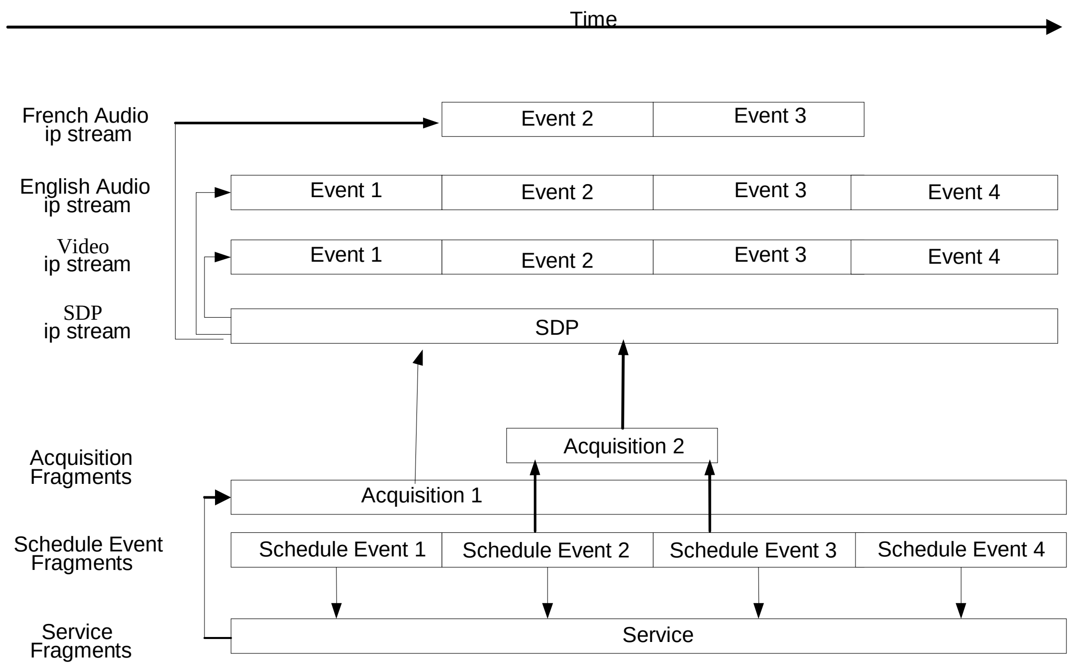


Figure 9: Relationship between described content streams, SDP files, Acquisition Fragments and other ESG XML Fragments in the second use case

5.10.2 Acquisition Fragment Syntax

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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>
```

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2008" >

<complexType name="AcquisitionExtensionType" mixed="false">
   <complexContent mixed="false">
      <extension base="esg:AcquisitionType">
         <sequence>
            <element name="privateData" type="esg:PrivateDataType" minOccurs="0"
         maxOccurs="unbounded" />
         <sequence>
         <attribute name="DeliveryChannel" type="esg2:DeliveryChannelType"/>
      </extension>
   </complexContent>
</complexType>

<simpleType name="DeliveryChannelType">
   <restriction base="string">
      <enumeration value="Broadcast"/>
```

```
        <enumeration value="Interactive"/>
        <enumeration value="Both"/>
     </restriction>
  </simpleType>
```

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	For streaming or download sessions over broadcast channel, the session description contains inlined SDP file which either directly describes the content session or the session carrying an SDP file that describes the content session (see clause 5.10.4). The transport of the SDP files in the latter case is specified in clause 8.5. For streaming or download over the interactive channel, this signals to terminals the way to access the Unicast session i.e. providing a URL or an SDP file.
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 clause 5.10.5).
KeyStream	Signals all available key streams for a given Acquisition Fragment required for decryption as described in clause 5.10.6. The mapping of key streams to media streams is described in TS 102 472 [5].
acquisitionID	Specifies a unique identifier of the instantiated Acquisition Fragment. For the scope of uniqueness see the semantics of serviceID in clause 5.4.2.
contentMimeType	Specifies the content type from which the terminal can determine the consuming application of the service.
DeliveryChannel	This field indicates the delivery method described in the Acquisition Fragment.
Broadcast	This fields indicates that the Acquisition Fragment contains only broadcast session descriptions.
Interactive	This fields indicates that the Acquisition Fragment contains only interactive session descriptions.
Both	This fields indicates that the Acquisition Fragment contains both broadcast and interactive session descriptions.
PrivateData	Generic element instantiated to add private data, e.g. specific to an SPP profile.

5.10.4 Session Description

5.10.4.1 Broadcast 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 signalled 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 signalled, or in cases where the Acquisition Fragment is signalled 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.

5.10.4.1.1 Session Description Syntax

```
   <!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

   <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>
```

5.10.4.1.2 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 signalled 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 TS 102 472 [5].This element is optional as the service may choose not to provide this functionality.
NOTE: White spaces i.e. space characters, carriage returns, line feeds or tabs shall be preserved in the processing of SDPType content.	

5.10.4.2 Unicast Session Description

This clause defines the Unicast Session Description to initiate reception of unicast services over the interactive channel.

5.10.4.2.1 Unicast Access Session Description Syntax

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2008" >

<complexType name="UnicastAccessType" mixed="false">
   <complexContent mixed="false">
      <extension base="esg:SessionDescriptionBaseType">
         <sequence>
            <element name="Profile" type="tva:ControlledTermType" minOccurs="0"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>

<complexType name="PollDownloadType" mixed="false">
   <complexContent mixed="false">
      <extension base="esg2:UnicastAccessType">
         <choice>
            <element name="RegistrationAccess" type="esg2:HTTPAccessType"/>
            <element name="PollAccess">
               <complexType mixed="false">
                  <complexContent mixed="false">
                     <extension base="esg2:PollAccessBaseType"/>
                  </complexContent>
               </complexType>
            </element>
         </choice>
      </extension>
   </complexContent>
</complexType>

<complexType name="PollAccessBaseType" mixed="false">
   <complexContent mixed="false">
      <extension base="esg2:HTTPAccessType">
         <attribute name="pollInterval" type="unsignedInt" use="required"/>
      </extension>
   </complexContent>
</complexType>

<complexType name="PushDownloadType" mixed="false">
   <complexContent mixed="false">
      <extension base="esg2:UnicastAccessType">
         <sequence>
            <element name="RegistrationAccess" type="esg2:HTTPAccessType"/>
         </sequence>
      </extension>
   </complexContent>
</complexType>
```

```
   <complexType name="HTTPAccessType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg2:UnicastAccessType">
            <sequence>
               <element name="HTTPAccessServerURL" type="anyURI"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>

  <complexType name="RTSPAccessType">
    <complexContent mixed="false">
      <extension base="esg2:UnicastAccessType">
        <choice>
          <element minOccurs="0" name="RTSPAccessServerURL" type="anyURI" />
          <sequence>
            <element name="InlineSDP" type="esg:SDPType" />
            <element minOccurs="0" name="SDPURI" type="anyURI" />
          </sequence>
          <element name="SDPURL" type="anyURI" />
        </choice>
      </extension>
    </complexContent>
</complexType>

   <complexType name="PSSAccessType">
      <complexContent>
         <extension base="esg2:RTSPAccessType"/>
      </complexContent>
   </complexType>
```

5.10.4.2.2 Unicast Access Session Description Semantics

The semantics of the unicast access session description fields are described by the following tables.

Field	Semantics
Profile	This controlled term defines the profile of unicast access as defined in clause C.4.

Poll Download Type Semantics

Field	Semantics
RegistrationAccess	This field describes the HTTP Access to register for poll delivery. Information about the AccessURL from the content can be polled is delivered as part of the registration response.
PollAccess	This field describes direct access to poll content over interactive channel.

PollAccessBaseType Semantics

Field	Semantics
PollInterval	The poll interval in seconds.

PushDownloadType Semantics

Field	Semantics
RegistrationAccess	This field describes the HTTP Access to register for push delivery.
NOTE: The content will be send to the terminal via the push mode signalled by UnicastAccess Profile.	

HTTPAccessType Semantics

Field	Semantics
HTTPAccessServerURL	The URL of the HTTP Server.

RTSP Access Type Semantics

Field	Semantics
RTSPAccessServerURL	RTSP URL providing the host name (and optionally port) of the streaming server. The protocol defined by the profile or extension shall be used [PSS, ControlledTermTypes***].
InlineSDP	Inline SDP file containing information that the terminal needs in order to be able to receive and consume the content of the service session.
SDPURI	URI to a content referencing SDP file describing the service session.
SDPURL	A URL from which the SDP file describing the service session can be retrieved.

PSS Access Type Semantics

The semantic of the fields is the same as for RTSP Access Type with exception of the PSS delivery used as defined in PSS specification [41].

5.10.4.3 Notification Session Description

5.10.4.3.1 Default Notification Session Description Syntax

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2008" >

<complexType name="DefaultNotificationSessionType" >
   <complexContent>
      <extension  base="esg:SessionDescriptionBaseType">
         <attribute name="isEDN" type="boolean" use="required"/>
      </extension>
   </complexContent>
</complexType>
```

5.10.4.3.2 Default Notification Session Description Semantics

The semantics of the field are as follows.

Table 1: DefaultNotificationSessionType Semantics

Field	Semantics
isEDN	When its values is "true", it refers to the ESG Default Notification (EDN) session. When its value is "false"; it refers to the Platform Default Notification (PDN) session or Network Default Notification (NDN) session.

NOTE: When it refers to NDN session, it is delivered over the same FLUTE session as PDN.

When session description type is based on DefaultNotificationSessionType, it signals to the terminal that Notification messages are delivered in a default Notification channel. This corresponds to the out-of-band delivery of user-selected Notifications. In this case, the attribute isEDN indicates whether or not the Notification messages are available on the EDN channel.

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 clause 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 clause C.2.

5.10.5.1 Zapping Support Syntax

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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 clause 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 clause 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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005" >

<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>

<!-- targetNamespace="urn:dvb:ipdc:esg:2008" >

   <complexType name="KeyStreamExtType" >
      <complexContent>
         <extension base="esg:KeyStreamBaseType" >
            <element name="ComponentID" type="esg2:ComponentIDType"/>
         </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] and EN 300 468 [25].
IPDCOperatorId	The ID of the entity who operates this key stream.
ComponentID	See clause 5.10.7.

5.10.7 Component Characteristic

In this clause 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

```
<!-- targetNamespace="urn:dvb:ipdc:esg:2005"-->

<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>
```

```
   <!-- targetNamespace="urn:dvb:ipdc:esg:2008"-->

   <complexType name="VideoComponentExtType" mixed="false">
      <complexContent>
         <extension base="esg:VideoComponentType">
            <attribute name="componentID" type="esg2:ComponentIDType"/>
         </extension>
      </complexContent>
   </complexType>

   <complexType name="AudioComponentExtType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg:AudioComponentType">
            <attribute name="componentID" type="esg2:ComponentIDType"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="FileDownloadComponentExtType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg:FileDownloadComponentType">
            <attribute name="componentID" type="esg2:ComponentIDType"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="NotificationComponentType">
      <complexContent>
         <extension base="esg2:ComponentCharacteristicExtType">
            <sequence>
               <element name="NotificationApplicationInit"
                                 type="esg2:NotificationApplicationInitType"
                                 minOccurs="0" maxOccurs="unbounded" />
               <element name="NotificationComponentIDRef"
                                 type="esg2:ComponentIDType" minOccurs="0" />
            </sequence>
            <attribute name="ReplaceSDP" type="boolean" use="required" />
            <attribute name="componentID" type="esg2:ComponentIDType"/>
         </extension>
      </complexContent>
   </complexType>

   <complexType name="NotificationApplicationInitType">
      <sequence>
         <element name="NotificationType" type="unsignedShort"/>
         <element name="NotificationMIMEType" type="mpeg7:mimeType"/>
         <element name="ComponentIDRef" type="esg2:ComponentIDType" minOccurs="0"/>
         <element name="ContentLocation" type="anyURI" minOccurs="0"/>
      </sequence>
   </complexType>

   <simpleType name="ComponentIDType">
      <restriction base="string">
         <pattern value="[!#\$%&amp;'\*\+-\.0-9A-Z\^_`a-z\{\|\}~]+"/>
      </restriction>
   </simpleType>
```

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.

In order to facilitate Notification component reference (and more general use cases, e.g. enable delivering multiple camera angles over the same session), ComponentCharacteristicsType is extended with an attribute ComponentID; The resulting ComponentCharacteristicExtType is used as the base type for NotificationComponentType.

Field	Semantics
ComponentID	Unique identifier a media component (e.g. audio component or Notification component). This identifier shall be unique in the context created by the services designed to be consumed simultaneously.

This component ID SHALL map to the attribute of the "a=label:" line in the SDP file.

AudioComponentCharacteristic

Field	Semantics
Codec	Signals which audio codec is used to represent the audio (see [6] for available codecs). 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 (see [6] for available codecs). 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 (RFC 2046 [2]).
Storage	Size of the available files signalled in megabytes.

NotificationComponentCharacteristic

Field	Semantics
ReplaceSDP	When its value= true, the SDP related to the Notification component (named as Notification SDP) replaces the SDP of the referred regular service or ScheduleEvent (named as referred SDP). When its value= false, the Notification SDP complements the referred SDP.
NotificationApplicationInit	Specifies initialization information for the notification application. This includes the notification type, the MIME type of the application specific message part, and optionally the location of the corresponding notification init container. A NotificationApplicationInit element shall be provided for each notification type carried over this component.
NotificationComponentIDRef	Specifies ComponentID of the Notification component that is complementary to the current Notification component (i.e. a component that carries payload objects over a FLUTE session).

NOTE 1: The NotificationApplicationInit element may not always be present in the Acquisition Fragment. This is for instance the case when the Notification type is a registered value.

When a Notification component is delivered over RTP, ReplaceSDP is set to true. In this case, the Notification SDP describes all media streams (e.g. A/V streams) in the referred SDP and all related Notification components delivered over RTP, including but not limited to the current one. In this case, the terminal shall use the Notification SDP to replace the referred SDP when consuming the Notification component together with its referring regular service.

When a Notification component is delivered over FLUTE, ReplaceSDP is set to false. In this case, in order to consume the Notification component together with its referring regular service, the terminal shall use the referred SDP to retrieve the latter while using Notification SDP or default Notification channel for the former.

When there are two Notification components (one over RTP <ReplaceSDP=true> and the other over FLUTE <ReplaceSDP=false>) for the same service or ScheduleEvent, the terminal shall carry out both manipulations as described above in order to consume them together with the referring regular service.

NOTE 2: The signalling of a Notification component in SDP over RTP is done by an attribute "a=label" for each media stream corresponding to a Notification component (see Notification Framework [42], clause 6.2.2.5). The terminal can thus look up the label in the SDP in order to retrieve the current Notification component and possibly other referring Notification components.

The above extensions allow a Notification component, e.g. over RTP, to refer to another Notification component, e.g. over FLUTE, which, for example, carries payload of the Notification messages of the former. Additionally, a Notification component over FLUTE, whether used for delivering Notification messages or not, can be referred by multiple RTP Notification components.

Field	Semantics
NotificationType	The Notification type to which the NIC is applicable to.
NotificationMimeType	The MIME Type of the corresponding Notification Type.
ComponentIDRef	If it is present, it signifies the componentID of the FLUTE Notification component that carries NIC. The following ContentLocation element announces the Content-location of the NIC. If it is absent, it signifies that the NIC is carried as a file over current ESG stream. In this case, the following element ContentLocation refers to the URI of the NIC file within the ESG flute session.
ContentLocation	This field signals the URI of the content location that is used to signify the NIC file within the FLUTE session.

The NotificationType field of the Notification messages (see Notification Framework [43], clause 6.1.1) of the Notification component SHALL map to the NotificationType element as described above.

NOTE 3: a NIC can be transported as:

1) A file transported by the ESG FLUTE session
2) A file transported by the FLUTE Notification component
3) A notification message transported by the RTP Notification component

For case 3, the Notification Init Container is signalled with a dedicated Notification Payload Format Type (see Notification Framework [44], clause 6.2, table 5).

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 (RFC 1952 [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 signalling of TV-Anytime information in DVB transport streams" (TS 102 323 [8]). The present 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 signalled in ALC/FLUTE as specified for file delivery in TS 102 472 [5].

To signal how the ESG XML Fragments are represented the ESG Init Message is defined in clause 6.2. The representation and encapsulation of the ESG XML Fragments itself are defined in clause 6.3. Finally the processing rules of those Encapsulated ESG XML Fragments are specified in clause 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. Over the broadcast channel, the ESG Init Message for broadcast ESG delivery is transported in the BC ESG Init Container (see clause 8.1.1) in the ESG FLUTE session in the single stream mode (see clause 8.3) or in the FLUTE Session of the Announcement Carousel in the Multiple Stream mode (see clause 8.4). Over the interactive channel, the IA ESG Init Container carrying the Init Message for interactive ESG delivery can be retrieved as specified in clause 10.5.2 via the ESG DeliveryList (see clause 10.2). The IA ESG Init message structure is the same as the BC ESG Init Message as specified in this clause with Indexing Flag always set to 0.

The IA ESG Init Container shall be only carried over the interactive channel.

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 TS 102 323 [8], except for the EncodingVersion and DecoderInit fields which are specified in this clause.

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 2.

Table 2: 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 the present document.
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 signalled 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 clause 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 clause 6.2.3 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 initialized with the default ESGMain Element fragment. The default ESGMain Element fragment is defined in clause 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 clause 7.

Over the broadcast channel, the BC ESG Init Container (see clause 8.1.1) shall be transported in the ESG FLUTE session in the single stream mode (see clause 8.3) or in the FLUTE Session of the Announcement Carousel in the Multiple Stream mode (see clause 8.4).

An IA ESG Init Container may be signalled in the DeliveryList as specified in clauses 10.2 and 10.4. The request of IA Init Container over the interactive channel is specified in clause 10.5.2.

6.2.2 Textual DecoderInit

The DecoderInit specified in this clause is used to transmit initialization 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 clause 6.3.1). Table 3 defines values to be used for the ESG XML Fragment Types declared within the present document. 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 1: This version is incremented if the specification of ESG Entry is changed in a not forward compatible way. NOTE 2: 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 3: 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 clause 6.3.1 to identify the ESG XML Fragment Type.

6.2.3 BiM DecoderInit

The DecoderInit specified in this clause is used to transmit initialization 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 ISO/IEC 15938-1 [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 2 with the field SchemaURI[0] of the DecoderInit set to "urn:dvb:ipdc:esg:2005", and the field SchemaURI[1] set to "urn:dvb:ipdc:esg:2008", indicating the use of the schemas defined in the present document.

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 ISO/IEC 15938-1 [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 clause 6.2.1.

As a consequence, the InitialDescription() field of the DecoderInit message as specified in ISO/IEC 15938-1 [3], shall always be empty.

Table 52 ContextPathCode extends the DVBContextPath of TS 102 323 [8] with the values specified in table 3.

Table 3: 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 clause 7.2 of ISO/IEC 15938-1 [3]. In the case variable length ContextPathCodes are signalled the ContextPathTableFlag shall be set to "1" in the DecoderInit according to ISO/IEC 15938-1 [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 ISO/IEC 15938-1 [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 the present document, this list is composed of the following codecs:

- dvbStringCodec (as it is specified in TS 102 323 [8]);
- dvbDateTimeCodec (as it is specified in TS 102 323 [8]);
- dvbDurationCodec (as it is specified in TS 102 323 [8]);
- dvbControlledTermCodec (as it is specified in TS 102 323 [8]).

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

DVB codec classification

```
<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>
```

6.3 Encapsulated ESG XML Fragment

In this clause the Encapsulated ESG XML Fragments carried in the ESG Data Repository as described in clause 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 clause 6.2.2 or may be well known by the terminal as specified in table 3.
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. NOTE: Fragments may not contain any string data. In such a case no particular value is specified for the external_string_buffer_ptr.
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 ISO/IEC 15938-1 [3]. If no ContextPathTable is transmitted, the field shall be encoded according to table 3. If a ContextPathTable is transmitted, the field shall be encoded according to this table (see clause 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 clause 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 3. If a ContextPathTable is transmitted, this field is deduced from this table and the ContextPathCode field (see clause 6.2.3.1).
FragmentUpdatePayload	Signals the payload of the ESG XML Fragment of XML data type startType as defined in ISO/IEC 15938-1 [3], clause 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 clause 5, 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.

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 clause 6.2.2, 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 clauses 7 and 8 of ISO/IEC 15938-1 [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 clause 6.2.3, 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.

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: The fragment management information enables the management of fragment creation, deletion and updates over time when containers are delivered over the broadcast channel . 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.

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 over Broadcast Channel

For delivery over the broadcast channel the ESG Container is provided a identifier (Container_ID) that is unique within the broadcast channel delivery and versioning information. Both Container_ID and versioning information shall be conveyed at the transport layer as specified in clause 8.1.1.2. Container_ID shall be a unique number within the scope of the broadcast 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. Signalling 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 signalled 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:

- 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]`		
`    }`		
`}`		

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 4.
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 5.
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 4: structure_type assignments

Value	Description
0x00	Reserved.
0x01	Fragment Management Information (see clause 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 clause 8.4.2.1).
0xE2	ESG Init Message (see clause 6.2).

Table 5: structure_type and their matching valid structure_id

Structure_type	structure_id	Description
0x01	0x00	Fragment Management Information (see clause 7.3).
0x02	0x00	String repository: Data Repository of type String (see clause 4.8.4.1 in TS 102 822-3-2 [21]).
0x03	0x0 to 0xFF	Reserved.
0x04	0x0 to 0xFE	Used to identify a specific instance of an index structure, within a container.
0x05	0x0 to 0xFE	Used to identify a specific instance of an multi_filed_sub_index structure, within a container.
0xE0	0x00	ESG Data Repository (see clause 7.4).
0xE1	0xFF	ESG Session Partition Declaration (see clause 8.4.2.1).
0xE2	0x00	ESG Init Message (see clause 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.

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

NOTE: Within the container the fragment management information (fragment_id, fragment_version, fragment_reference) is aggregated together, separated from the fragments and it is possible to consume the management information before the fragments in the 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 6.
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 6 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. For fragment delivery over the broadcast channel the following applies: The fragment_id uniquely identifies an ESG Fragment within the broadcast 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 broadcast 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. For fragment delivery over the interactive channel the following applies: The fragment_id maps to the transportID in the DeliveryList and is used to enable quick access to a fragment in the container structure. The fragment_id should be unique for each IAChannel AccessPoint signalled in the DeliveryList. It is not required that the id is unique across multiple AccessPoints.

Table 6: Valid fragment_reference_formats

Value	Meaning
0x00 to 0x20	Reserved.
0x21	Generic ESG Fragment Reference (see clause 7.3.3).
0x22 to 0xE0	Reserved.
0xE1 to 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.
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: Valid esg_fragment_type values

Value	Meaning
0x00	Encapsulated ESG XML Fragment (see clause 7.4.3).
0x01	Encapsulated ESG Auxiliary Data (see encapsulated _aux_data() in clause 7.4.5) - (see note).
0x02	Encapsulated Private Auxiliary Data (see encapsulated _aux_data() in clause 7.4.5) - (see note).
0x03 to 0xE0	Reserved.
0xE1 to 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 clause 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 clause 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 clause 6.3.
encapsulated_aux_data()	This field specifies an Encapsulated ESG or Private Auxiliary Data Fragment (see clause 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 clause 6.2.
EncapsulatedBiMESGXMLFragment	The Encapsulated BiM ESG XML Fragment is specified in clause 6.3.2.
EncapsulatedTextualESGXMLFragment	The Encapsulated Textual ESG XML Fragment is specified in clause 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 clause 6.2.3, the XML Schema definition of the XML Envelope is the schema defined in clause 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 RFC 2616 [19] and IANA [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 TS 102 323 [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 clause 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 8. 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 8: VersionId Values

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

7.4.7 XML envelope

In this clause 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 TS 126 346 [15] and the binary representation specified in clause 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="string" use="required"/>
         <anyAttribute processContents="skip"/>
      </complexType>
   </element>
   <!-- the format of the data is defined by the encoding attribute and the data is carried in the
            AuxData element encoded in base64 -->
   <complexType name="AuxDataType">
      <simpleContent>
         <restriction base="base64Binary"/>
      </simpleContent>
   </complexType>
<schema>
```

8 ESG Transport over Broadcast Channel

In this clause the transport of ESG Containers over the broadcast channel is described. The present document 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 clause 9.1.2). In both modes ESG Containers are transported in FLUTE dynamic file delivery carousel sessions as described for file delivery in TS 102 472 [5]. The transport of ESG Containers is specified in clause 8.1.1.

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 clause 8.1.2.

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 TS 102 472 [5] in the mode as described in clause 8.1.3.

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 clause 8.1.4.

In addition the transport of SDP files for the acquisition of content is specified in clause 8.1.5.

8.1 Transport of ESG using FLUTE

In this clause 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. Clause 8.1.1 specifies how ESG Containers are carried in Transport Objects, which relies on the FDT for signalling of ESG Container IDs and version changes. Clause 8.1.2 specifies an optional signalling of ESG Container IDs and Versions based on the Split TOI mechanism. In clause 8.1.3 signalling of ESG consistency and completeness is described.

8.1.1 Transport of ESG Containers using FLUTE

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 signalled in the FDT by setting the attribute Content-Type="application/vnd.dvb.esgcontainer". The ESG Containers are identified based on the URI signalled in the "Content-Location" attribute of the "File" elements in the FDT (see clause 8.1.2). The version change of the ESG Containers are signalled based on the TOI and the FDT Instance ID as described in clause 8.1.2.

The ESG Containers may be compressed with GZIP (RFC 1952 [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.

Initialization information for the processing of ESG data shall be carried in one ESG Container called the ESG Init Container. In the present document the initialization information is composed of:

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

The Transport Object carrying the ESG Init Container shall be identified by the containerID 1 (see clause 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 clause 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:esg:cid".

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

urn:dvb:ipdc:esg:cid:23

Clause 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 signalled based on the TOI and the FDT Instance ID as described for file delivery in clause 6.1.12 of TS 102 472 [5].

In addition when using the Split TOI mechanism described in clause 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 clause 8.1.3.1).

If the FDT Split-TOI mechanism is not supported, a MD5 checksum should be used to identify the correct container. The MD5 value should be signalled in the FDT by the Content-MD5 attribute to allow instant identification of the correct container. If no Content-MD5 is signalled in the FDT, the terminal should compile the MD5 checksum of received containers or may only use the container identification as signalled by the FDT Content-Location with the risk of possible inconsistency if a new version has been announced but the old version is still in the FLUTE carousel and picked by the terminal.

8.1.3 Version Signalling 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 clause 8.1.2. Additionally the ESG Container ID and version may be signalled by the mechanisms described in this clause, 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) (RFC 3450 [9], RFC 3926 [10] and RFC 3451 [11]).

The LCT TOI field is $32 \times O + 16 \times H$ 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 document 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 the present document) 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.

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 behaviour 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.

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: "010000000000000001"
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: "100000000000000001"
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 "110000000000000000" 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.

In cases where an ESG is transported, partially or fully, over the Interactive Channel, the DeliveryList shall be used to manage overall consistency (see clause 11).

8.1.4.1 Consistency signalling in FLUTE FDT

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 signalling in the FDT is specified.

The ESG Containers are delivered over FLUTE dynamic file delivery carousel sessions, as defined in TS 102 472 [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" />
```

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 TS 102 822-3-2 [21].

The presence of the fragment index is announced using the Index_list structure (see TS 102 822-3-2 [21]), which is carried within the ESG Init Container (see clause 8.1.1).

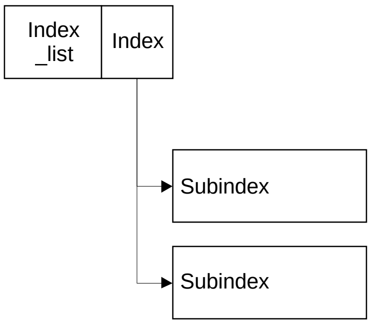


Figure 10: Overview of index structure

Figure 10 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 clause 4.8.5.2 of TS 102 822-3-2 [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.

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 the present document a number of values for this field has been specified (see table 9).

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 tables 11 and 12.

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 9: 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 to 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 clause 4.8.5.4 of TS 102 822-3-2 [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 the present document 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 clause 4.8.5.5 of TS 102 822-3-2 [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 declaration 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 TS 102 472 [5]. The FLUTE session is described as specified in clause 9.1.2 for the ESGAccessDescriptor.

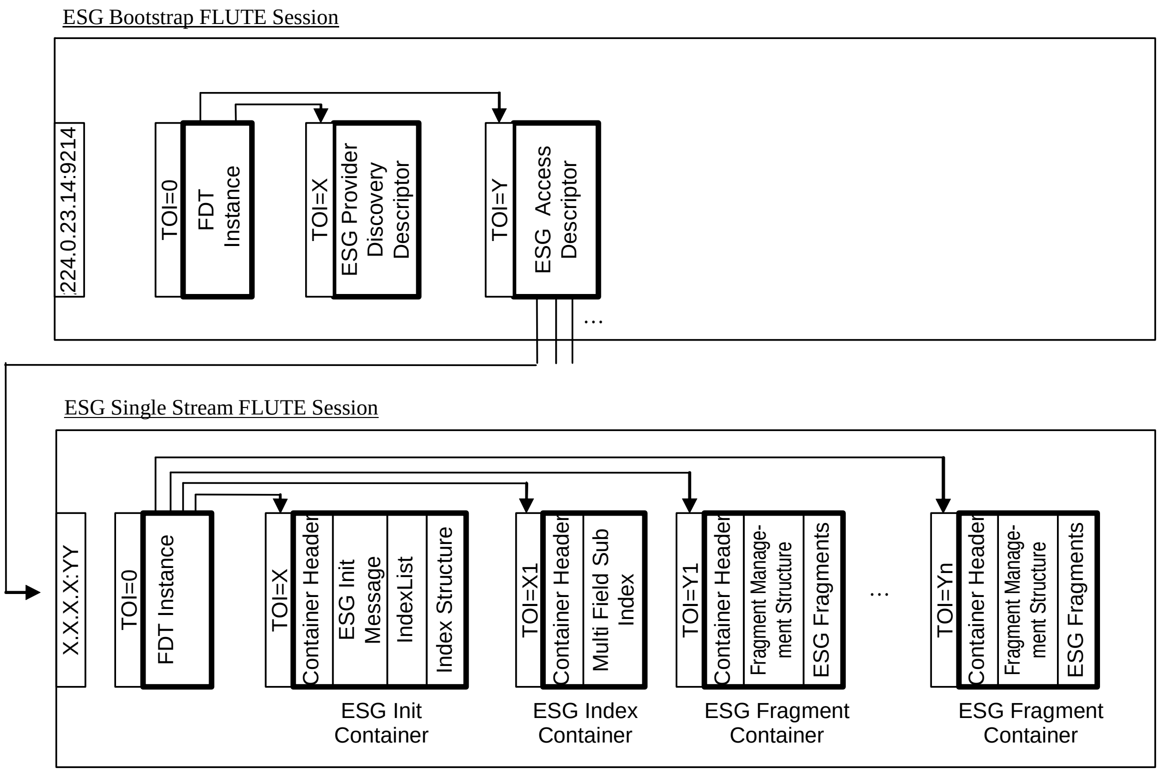


Figure 11: Diagram of the Single Stream Transport of the ESG

8.4 ESG Multiple Stream Transport

In this clause the transport of the ESG on multiple IP streams is specified. In the next clause 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 initialization information as specified in clause 8.4.2. The remaining FLUTE sessions transporting the ESG are specified in clause 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 clause) and the Fragment Indices (see clause 8.2) are carried in the Announcement Carousel on a known IP stream that is signalled within the ESG Access descriptor (see clause 9.1.2).

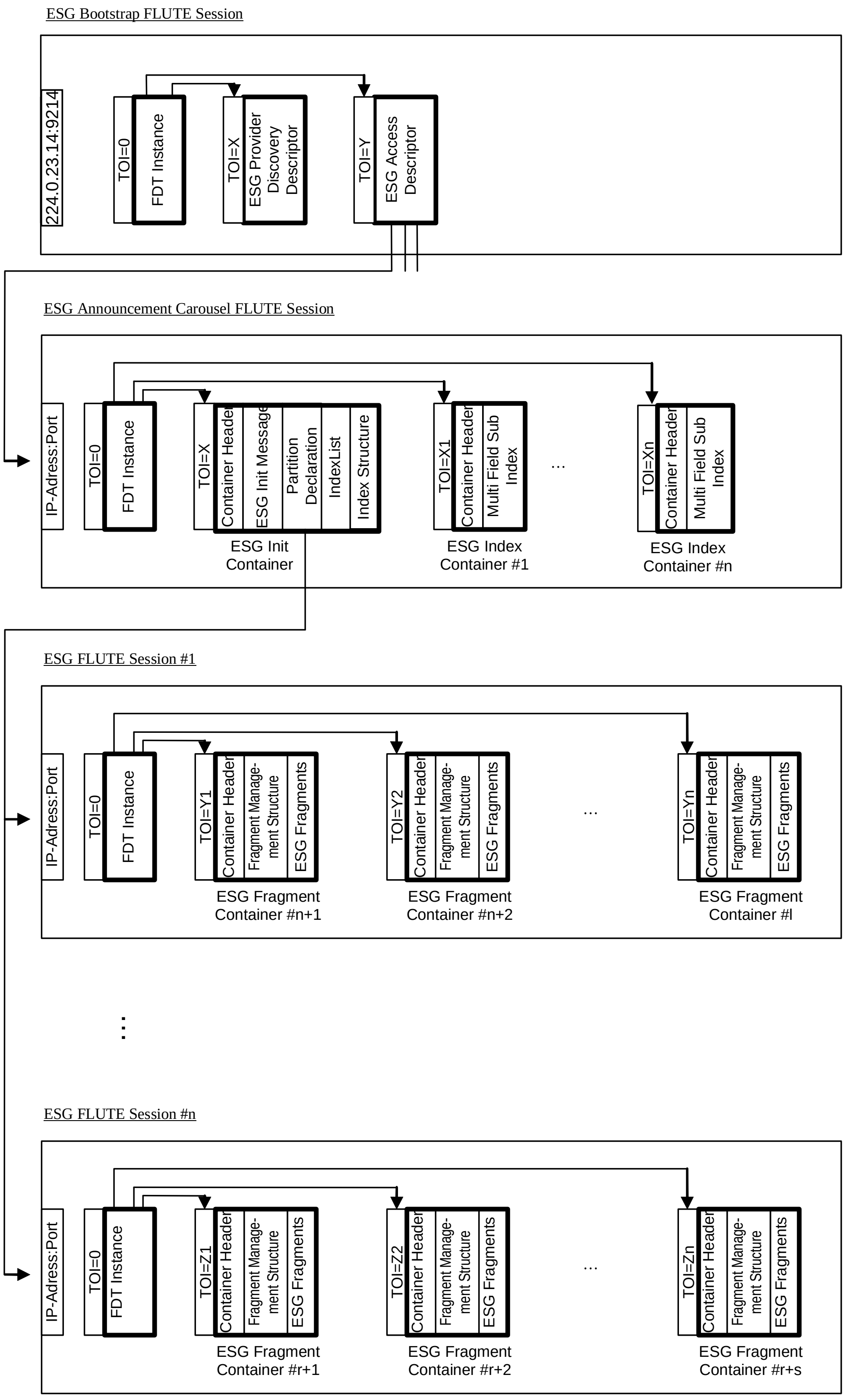


Figure 12: Diagram of the Multiple Stream Transport of the ESG and the references between the streams

8.4.2 Announcement Carousel

The announcement carousel carries the ESG Init Container as specified in clause 8.1.1 and optional ESG Containers carrying the Fragment Index (see clause 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 clause 8.1. The FLUTE Session transporting the Announcement Carousel is described as specified in clause 9.1.2 for the ESGAccessDescriptor. The FLUTE Session transporting the ESG Containers are described as specified in clause 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 clause 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 10).
field_encoding[k]	Signals how the field data is encoded. e.g. String, Integer, float (see tables11 and 12).
field_length[k]	Specifies the size of the start_field[k] and end_ field[k] in number of bytes. The value of "0x00" is used to indicate that the field is variable in length.
n_o_IPStreams	Signals 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 signalled in the field_identifier[k]. The encoding and size of this field is specified in table 12.
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 signalled in the field_identifier[k]. The encoding and size of this field is specified in table 12.

Table 10: 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 to 0xEF		DVB Reserved.
0xF0 to FE		User Defined.
0xFF		Reserved.

Table 11: Semantics of field_encoding values

Value	Meaning
0x0000	Field contains an inline string in UTF-8 (RFC 3629 [14]). For this field_encoding the field_length shall be set to 0x00.
0x0001 to 0x00FF	Field is an offset in bytes from the start of the string data repository structure.
0x0100 to 0x01FF	Field contains an inline 2-byte value.
0x0200 to 0x0201 0x0300 0x0401	Field contains an inline 4-byte value.
0x0204 to 0x0206	Field contains an inline 1-byte value.
0x0202 to 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 to 0x02FF 0x0402 to 0x04FF	Undefined.
0x0500 to 0xFFFF	Reserved for future use.

Table 12: Encoding types and their respective sizes

field_encoding	Description	Encoding	Size in bits
0x0000	String type.	Null-terminated string.	variable (8+)
0x0001 to 0x00FF	Reserved for custom string types.		
0x0100	Signed short.	two's complement - Big-Endian.	16
0x0101	Unsigned short.	unsigned binary - Big-Endian.	16
0x0102 to 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 to 0x02FF	Reserved for custom integer types.		
0x0300	Signed float.	IEEE 754-1985 [16] Big-Endian.	32
0x0301	Reserved.		
0x0302	Signed double.	IEEE 754-1985 [16] Big-Endian.	64
0x0303 to 0x03FF	Reserved for custom rational types.		
0x0400	DateTime.	Modified Julian Date and Milliseconds (TVA BiM codec, clause 4.4.2.4.2 in TS 102 822-3-2 [21]).	64
0x0401	Date.	Modified Julian Date (TVA BiM codec, clause 4.4.2.4.3 in TS 102 822-3-2 [21]).	32
0x0402 to 0x04FF	Reserved for custom binary fragments.		
0x0500 to 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 clause 7.2.3).

NOTE: The FLUTE Sessions transporting the ESG Containers are described as specified in this clause for the ESG Session Partition Declaration. The parameters fixed for the description of the FLUTE session transporting the announcement carousel in clause 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 clause 5.10.1. These SDP files are transported in ALC/FLUTE sessions as described for file delivery in TS 102 472 [5].

NOTE 1: According to clause 5.10.4 the SDP file may alternatively be inlined in the Acquisition Fragment.

NOTE 2: The number of different SDP files transported in one ALC/FLUTE session is not restricted.

NOTE 3: The transport of SDP Files in an ALC/FLUTE session MAY use the mechanism to split the LCT TOI field as described in clause 8.1.

9 ESG Bootstrapping

In this clause the ESG bootstrap process is specified. As introduced in clause 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 clause 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 clause 9.2.

c) The Discovery of Bootstrap Entry Points to receive ESG bootstrap information is specified in clause 9.3.

9.1 The ESG Bootstrap Descriptors

The ESGProviderDiscovery descriptor specifies the ESG providers that deliver ESGs in a given IP Platform. 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 or ESG_URI associated with each described ESG, 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 signalled in the ESGProviderDiscovery descriptor. An ESG Provider can offer several ESGs identified by a unique ESG_URI. For each ESG the AccessPoints from where a DeliveryList and ESG Fragments can be retrieved are signalled in the AccessDescriptor. Each AccessPoint is identified by accessPointID. The "interactive" and "complementary" AccessPoint implies its capability and the context of a specific ESG_URI associated to it, whereas the "repair" AccessPoint always offers the same ESG Fragments as in the broadcast delivery.

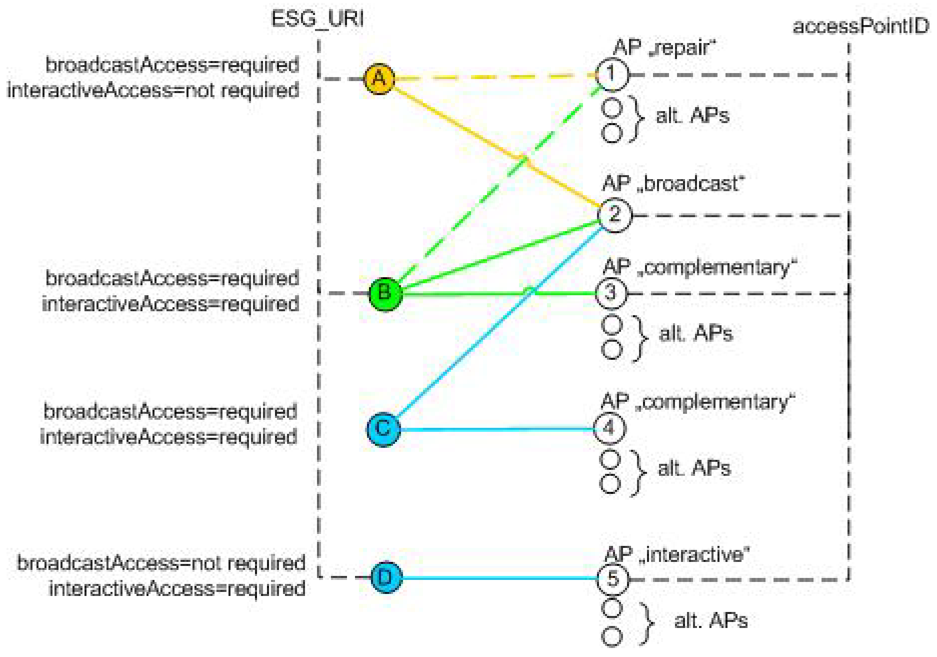


Figure 13: Examples of ESG Bootstrap Access Points

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>
      <attribute name="format" type="anyURI" use="optional" default="urn:dvb:ipdc:esg:2005"/>
   </complexType>

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

ESGProviderDiscovery Descriptor Semantics

Field	Semantics
ServiceProvider	An ESGProvider offering one or more ESGs. NOTE: An instance shall use bs2:ESGProviderExtensionType as specified in clause 9.1.1.1.
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.
format	Specifies the format of the ESG provided by this ESG provider. This value shall be set to urn:dvb:ipdc:esg:2008 for ESG compliant to the present document.

9.1.1.1 ESGProviderType Extension

The following extension enables signalling of multiple ESGs of an ESG Provider, where for each ESG the required broadcast or interactive AccessType is signalled and associated AccessPoints in the ESGAccess descriptor.

Syntax of ESGProviderExtensionType

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns:bs="urn:dvb:ipdc:esgbs:2005" xmlns:tva="urn:tva:metadata:2005"
            xmlns:bs2="urn:dvb:ipdc:esgbs:2008" xmlns="http://www.w3.org/2001/XMLSchema"
            targetNamespace="urn:dvb:ipdc:esgbs:2008" elementFormDefault="qualified"
            attributeFormDefault="unqualified">
   <import namespace="urn:dvb:ipdc:esgbs:2005"/>
   <import namespace="urn:tva:metadata:2005"/>

   <complexType name="ESGProviderExtensionType" mixed="false">
```

```
      <complexContent mixed="false">
         <extension base="bs:ESGProviderType">
            <sequence>
               <element name="ESG" type="bs2:ESGType" maxOccurs="unbounded"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>

   <complexType name="ESGType">
      <sequence>
         <element name="ESG_URI" type="anyURI"/>
         <element name="TargetAudienceType" type="tva:ControlledTermType" minOccurs="0"
            maxOccurs="unbounded"/>
         <element name="AccessPoint" type="bs2:accessPointType" maxOccurs="unbounded"/>
      </sequence>
      <attribute name="broadcastAccess" type="bs2:AccessType" use="optional" default="required"/>
      <attribute name="interactiveAccess" type="bs2:AccessType" use="optional" default="not
            required"/>
   </complexType>

   <simpleType name="AccessType">
      <restriction base="string">
         <enumeration value="required"/>
         <enumeration value="not required"/>
      </restriction>
   </simpleType>

   <complexType name="accessPointType">
      <attribute name="accessPointID" type="unsignedInt"/>
   </complexType>
</schema>
```

Semantics of ESGProviderExtensionType

Field	Semantics
ESG	The ESG element contains information about ESGs offered by a ServiceProvider, and enables globally unique identification of an ESG and 1 to 1 mapping to an ESG entry in the AccessDescriptor Multiple ESGs can be signalled for each one ServiceProvider.
ServiceProvider	The ServiceProvider Element describes the ESG Provider.
ESG_URI	Globally unique identifier of an ESG.
TargetAudienceType	Target audience of the particular ESG, e.g. for roaming or local users. This is a controlled term of a Classification Scheme.
AccessPoint	Each ESG has one or several AccessPoints which can be either a broadcast AccesPoint or interactive AccessPoint.
accessPointID	The identifier for an AccessPoint that maps to an accessPointID in the ESG AccessDescriptor.
broadcastAccess	Defines if broadcast access is needed to acquire the ESG. Default value is "required".
interactiveAccess	Defines if interactive access is needed to acquire the ESG. Default value is "not required".

The interactveAccess and broadcastAccess attributes in the ProviderDiscoveryDescriptor give the terminal some information about the minimum connectivity capability needed to access a specific ESG entry, e.g. a terminal that does not have IA access capability should not select an ESG that requires IA access. When a terminal has both BC and IA channel capabilities and multiple entries could be chosen, it is up to the user preferences which entry to choose (e.g. there is most of the time good DVB-H coverage so the user prefers to retrieve ESG over BC) or the terminal may be preconfigured with a preferred entry by other means (e.g. by the operator).

The following possible combinations of AccessTypes map to the AccessPoints of different capability:

AccessType	BC required	BC not required
IA required	"complementary"	"interactive"
IA not required	"repair" or broadcast only	n/a

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]()`		
`   }`		
`}`		

ESGAccessDescriptor Semantics

Field	Semantics
n_o_ESGEntries	Specifies the number of ESGEntries in which access information to ESGs is signalled.

In order to signal AccessPoints for broadcast and interactive access and link them to an ESG in the ESG ProviderDiscovery, the following Version 2 ESGEntry shall be used.

NOTE: For Version 1 ESG Entry to support legacy terminals see clause H.2.

Version 2 ESGEntry Syntax

Syntax	No. of bits	Mnemonic
`ESGEntry {`		
`   ESGEntryVersion`	8	uimsbf
`   ESGEntryLength`	8+	vluimsbf8
`   AccessPointID`	8	uimsbf
`   descriptor()`		
`}`		

Version 2 ESGEntry Semantics

Field	Semantics
n_o_ESGEntries	Specifies the number of ESGEntries in which access information to ESGs is signalled.
ESGEntryVersion	The ESGEntryVersion SHALL be "2" for the ESGAccessDescriptor as specified in the present document.
ESGEntryLength	The EntryLength specifies the length of the ESGEntry in Bytes excluding the ESGEntryVersion and EntryLength fields.
AccessPointID	The identifier of the accessPoint in the ESGAcessDescriptor. This is referenced from the ProviderDiscovery Descriptor for a given ESG. The accessPointID is unique within the AccessDescriptor.

Table of defined descriptors

DescriptorBroadcast Descriptor	Describes acces over broadcast channelSemantics
Broadcast Descriptor	Describes acces over broadcast channel.
Interactive Descriptor	Describes access over interactive channel.

For a given ESG_URI, multiple AccessPoints with different capabilities may be listed. However each capability shall be available only once in the AccessDescriptor for a given ESG_URI:
Only one accessPointID referencing one broadcast AccessPoint shall be declared.
For one broadcast access it is assumed that only one repair AccessPoint URL and/or complementary AccessPoint URL is given in the scope of a given ESG_URI. Also there is at most one "interactive" AccessPoint for a given ESG_URI.

To enable load balancing and recovery from unavailable APs, for each capability alternative AccessPoint URIs may be present.
An AccessPoint shall be associated with only one ESG_URI and is not reused by different ESGs with exception of the "repair" and "broadcast" AccessPoint.

Each AP implies a certain context: i.e. each AccesPoint handles request for a specific ESG and specific capability (complementary, interactive, repair). Therefore, when requesting a DeliveryList and ESG Fragments or Containers, the ESG_URI and targeted capability do not have to be included as request parameters but are implicit for the AccessPoint.

An AccessPoint shall only handle requests for its capability (e.g. repair requests shall be sent to the repair AccessPoint and not the complementary AccessPoint).

BroadcastDescriptor Syntax in the ESGAccessDescriptor

Syntax	No. of bits	Mnemonic	Notes
`Broadcast Descriptor(){`			
`   Descriptor_tag`	8	uimsbf	0
`   Descriptor_length`	8+	vluimsbf8	
`   MultiStreamTransport`	1	bslbf	
`   IPVersion6`	1	bslbf	
`   Reserved`	4	bslbf	
`   if(IPVersion6 == "1") {`			
`      SourceIPAddress`	128	bslbf	
`      destinationIPAddress`	128	bslbf	
`   } else {`			
`      SourceIPAddress`	32	bslbf	
`      DestinationIPAddress`	32	bslbf	
`   }`			
`   Port`	16	uimsbf	
`   TSI`	16	uimsbf	
`}`			

Semantics of BroadcastDescriptor

Field	Semantics
Descriptor_tag	Identifies uniquely the descriptor. (See table 13 for "Descriptor_tag" assigned values).
Descriptor_length	Length of the descriptor in bytes.
MultiStreamTransport	If set to "1" specifies that a FLUTE session is described which transports an Announcement Carousel Session (see clause 8.4). If set to "0" specifies that a FLUTE session is described which contains all ESG Containers of that ESG (see clause 8.3).
IPVersion6	If set to "1" specifies that the SourceIPAddress and the DestinationIPAddress are signalled according to IP version 6. If set to "0" specifies that the SourceIPAddress and the DestinationIPAddress are signalled according to IP version 4.
Reserved	Field reserved for future extensions.
SourceIPAddress	Specifies the source IP address of the FLUTE session transporting the ESG. The IPVersion is signalled by the IPVersion6 field.
DestinationIPAddress	Specifies the destination IP address of the FLUTE session transporting the ESG. The IPVersion is signalled 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.

Interactive Descriptor Syntax in the ESGAccessDescriptor

Syntax	No. of bits	Mnemonic
`Interactive Descriptor(){`		
`   Descriptor_tag`	8	uimsbf
`   Descriptor_length`	8+	vluimsbf8
`        Interactive_capability`	2	uimsbf
`        Reserved`	4	
`        n_o_alternativeAPs`	4	uimsbf
`        for(n=0; i< n_o_alternativeAPs; n++) {`		
`            URLLength`	8+	vluimsbf8
`        for(i=0; i<URLLength; i++) {`		
`            URLByte`	8	uimsbf
`        }`		
`        }`		
`}`		

InteractiveDescriptor Semantics

Field	Semantics
Descriptor_tag	Identifies uniquely the descriptor (see table 14 for the "Descriptor_tag" assigned values).
Descriptor_length	The "Descriptor_length" specifies the length of the actual interactive entry in bytes excluding the "Descriptor_tag" and the "Descriptor_length" fields.
Interactive_capability	Signals the capability of the interactive AccessPoint (see table16 for Interactive_capabilities values semantic).
Reserved	Field reserved for future extensions.
n_o_alternativeAPs	number of alternative AccessPoints (each offering same capability).
URLLength	Number of bytes that form the URI as specified in RFC 3987 [36].
URLByte	URI octet.

Table 15: Descriptor_tag values

Descriptor_tag Value	Descriptor_tag semantic
0x00	Broadcast descriptor.
0x01	Interactive descriptor.
0x02 to 0xFF	Reserved for future use.

In the table below, the semantic of the Interactive AccessPoint capability values is provided.

Table 16: Interactive_capabilies values

Interactive_capability value	Semantics
0	The "repair" capability is signalled.
1	The "complementary" capability is signalled.
2	The "interactive" capability is signalled.
3	For future use.

Semantics of ESG AccessPoint capabilities

AccessPoint Capability	Semantics	Supported Queries
"interactive"	All Fragments of an ESG are delivered over Interactive Channel. An ESG retrieved over this AccessPoint may be different from the ESG received over the broadcast channel.	DeliveryList [ESG Containers] [ESG Fragments] (see note).
"repair"	broadcasted containers can be delivered also over the Interactive Channel.	ESG Containers.
"complementary"	fragments of an ESG are partly delivered over Broadcast and Interactive Channel.	DeliveryList [ESG Containers] [ESG Fragments] (see note).
NOTE: If a DeliverList contains AccessPoints those shall be used to query for Containers and Fragments. If no AccessPoints are given in the DeliveryList the AccessPoint signalled here in the Bootstrap shall support queries for Containers and Fragments.		

See also annex G for the scenarios.

9.2 Transport of ESG Bootstrap Descriptors

This clause specifies the transport and retrieval of the ESG Bootstrap descriptors over broadcast and interactive channel.

Over the broadcast channel Version 1 and Version 2 bootstrap information as specified in clauses 9.3.1 and 9.3.2 can be delivered as specified in clause 9.2.1. I.e. to address legacy terminals Version 1 BootstrapDescriptors are delivered along with Version 2 BootstrapDescriptors.

9.2.1 Transport of ESG Bootstrap Descriptors over broadcast channel

The ESG Bootstrap Descriptors are transported on ALC/LCT as specified in TS 102 472 [5] for FLUTE sessions.

According to the file delivery specification in TS 102 472 [5] the following information is required to launch a FLUTE agent in the terminal. As there is no explicit signalling 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 TS 102 472 [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 Version 1 ProviderDiscovery Descriptor (see clause 9.1.1) file is signalled in the FDT by setting the attribute Content-Type="text/xml". The ProviderDiscovery Descriptor shall be represented in UTF-8 character encoding (RFC 3629 [14]). The Version 1 ESGAccessDescriptor file (see clause H.2) is signalled in the FDT by setting the attribute Content-Type="application/vnd.dvb.ipdcesgaccess".

The Version 2 ProviderDiscovery Descriptor file (see clause 9.1.1.1) is signalled in the FDT by setting the attribute Content-Type=" application/vnd.dvb.ipdcesgpdd". The ProviderDiscovery Descriptor shall be represented in UTF-8 character encoding (RFC 3629 [14]). ESGProviderDiscovery Version 2 can be compressed with GZip which is signalled in FDT with Content-Encoding="gzip". The Version 2 ESGAccessDescriptor file is signalled in the FDT by setting the attribute Content-Type="application/vnd.dvb.ipdcesgaccess2".

NOTE: The mechanism of Split TOI described in clause 8.1.3 may also be applied in the case of ESG Bootstrap Descriptor transport. If split TOI is not supported, the Content-MD5 field can be used to identify updates of ESGBootstrp Descriptors as specified in clause 8.4. Therefore the FLUTE FDT should signal the Content-MD5 for transport objects in the ESG Bootstrap session.

9.2.2 Retrieval for ESG Bootstrap Descriptors over interactive channel

This clause specifies how a terminal can retrieve ESG Bootstrap information for a specific IP Platform over an Interactive Channel.

HTTP/1.1 protocol shall be used to retrieve the ESG Bootstarp descriptors by the terminal.

9.2.2.1 Request for ESG Bootstrap Descriptors

The terminal requests for the ESG Bootstrap descriptors using the entry point discovered as specified in clause 9.1.3, with a request command with type="bootstrap" to the ESG Bootstrap Entry Point. The bootstrap entry point URL shall be unique, i.e. not collide with URLs to query for ESG information.

The entries in the ESG Bootstrap Descriptors requested over interactive network may not be the same as in the bootstrap descriptors received over broadcast.

The "Request-URI" of HTTP POST request SHALL be set to the interactive ESG Bootstrap Entry Point URL. When the terminal requests for Bootstrap information, the "message-body" of HTTP/1.1 request message MAY contain key-value pairs, with possible values specified in table 17.

The HTTP/1.1 request message MAY contain a Key-value pair using "platformID" as the key and the IP Platform identifier (PLATFORM_ID) as the value. If the PlatformID is not globally unique, the "message-body" of HTTP/1.1 request message SHALL contain a key-value pair, using "networkID" as the key and the DVB Network identifier (Network_ID) as registered by DVB [45].

The response shall be as defined in clause 9.2.2.2 with the Bootstrap Descriptors of the requested Platform.

- If the request does not contain the "platformID" parameter the response shall be as defined in in clause 9.2.2.2 with the Bootstrap Descriptors of all Platforms available from this Bootstrap Entry Point.

The HTTP/1.1 request message MAY contain key-value pair using "IPDCKMSID" and "IPDCOperatorID" as the key and the IPDC KMS identifier (IPDCKMSId) and IPDC operator identifier (IPDCOperatorId) as the value of the supported IPDCOperator of the terminal.

- Support for this query by the server is optional:
  - If a Bootstrap Entry Point does not support queries for IPDCOperator specific Bootstrap Descriptors, it shall ignore these parameters and the response shall be as defined in clause 9.2.2.2 with the Bootstrap Descriptors available from this Bootstrap Entry Point.
    - If a Bootstrap Entry Point does support this query, the response shall be as defined in clause 9.2.2.2 with Bootstrap Descriptors tailored for the requested IPDCOperator.

NOTE: The latter query parameters are useful when a roaming terminal does not know the IP PlatformID to use or which ESG is associated to its roaming partners (e.g. when RoamingInformationDescriptor is not available in the Bootstrap information). In this case, based on the key-value of IPDCOperator in the request message, the bootstrap entry point could check its roaming partner and find associated IP Platforms and ESGs and provide tailored Bootstrap Descriptors to the terminal.

The HTTP/1.1 request message MAY contain key-values for both IPDCOperator and IP Platform. The response message contains ESG Bootstrap information for the requested IPDCOperator in the requested IP platform.

Key-value for querying the ESG Bootstrap Descriptor:

Table 17: Key-Value description for Bootstrap query

Key	**Value**	**Semantics**
IPDCKMSID	IPDCKMSId	The ID of the key management system(KMS) as defined and registered by DVB. This is the IPDCKMSId of the terminals Home IPDC Operator.
IPDCOperatorID	IPDCOperatorId	The ID of the entity who operates this key stream. This is the IPDCOperatorId of the terminals Home IPDCOperator.
Platform ID	Platform_id	The ID of the IP platform.
Network ID	Network_id	The ID of the network.

9.2.2.2 Response for ESG Bootstrap Descriptors

The response to the request SHALL be HTTP/1.1 response.

The response message shall be an instance of the following schema with the BootstrapDescriptors embedded:

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns:bs="urn:dvb:ipdc:esgbs:2005"
            xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
            xmlns:bs2="urn:dvb:ipdc:esgbs:2008" xmlns:bsResponse="urn:dvb:ipdc:esgbs:response:2008"
            xmlns="http://www.w3.org/2001/XMLSchema"
            targetNamespace="urn:dvb:ipdc:esgbs:response:2008" elementFormDefault="qualified"
            attributeFormDefault="unqualified">
   <import namespace="urn:dvb:ipdc:esgbs:2008" schemaLocation="dvd_ipdc_esg_bs_2008.xsd"/>
   <import namespace="urn:dvb:ipdc:esgbs:2005" schemaLocation="dvd_ipdc_esg_bs_2005.xsd"/>
   <import namespace="urn:mpeg:mpeg7:schema:2001" schemaLocation="mpeg7.xsd"/>
   <element name="BootstrapDescriptorResponse">
      <complexType>
         <sequence>
```

```
            <element name="BootstrapDescriptors" type="bsResponse:BootstrapDescriptorType"
            maxOccurs="unbounded"/>
         </sequence>
      </complexType>
   </element>
   <complexType name="BootstrapDescriptorType">
      <sequence>
         <element name="ESGProviderDescriptors">
            <complexType>
               <sequence>
                  <element ref="bs:ESGProviderDiscovery"/>
               </sequence>
               <attribute name="version" type="unsignedByte" use="optional"/>
               <attribute name="mime-type" type="mpeg7:mimeType" use="optional"/>
            </complexType>
         </element>
         <element name="ESGAccessDescriptor" type="bsResponse:BinaryDescriptor"/>
         <element name="RoamingInfoDescriptor" type="bsResponse:BinaryDescriptor" minOccurs="0"/>
      </sequence>
    <attribute name="IPPlatformID" use="required">
      <simpleType>
        <restriction base="hexBinary">
          <maxLength value="3" />
        </restriction>
      </simpleType>
    </attribute>
    <attribute name="DVBNetworkID" use="optional">
      <simpleType>
        <restriction base="hexBinary">
          <maxLength value="2" />
        </restriction>
      </simpleType>
    </attribute>
   </complexType>
   <complexType name="BinaryDescriptor">
      <simpleContent>
         <extension base="base64Binary">
            <attribute name="version" type="unsignedByte" use="optional"/>
            <attribute name="mime-type" type="mpeg7:mimeType" use="required"/>
         </extension>
      </simpleContent>
   </complexType>
</schema>
```

9.3 Discovery of ESG Bootstrap Entry Points

In this clause, the mechanisms to discover an BootstrapEntryPoint from which the terminal can receive the ESG Bootstrap information is defined.

9.3.1 Bootstrap Entry Point Discovery over Broadcast Channel

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 TR 102 469 [i.1]). 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 (see EN 300 468 [25]).

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.

The ESG Bootstrap Entry Point of the broadcast network is a well-known destination IP address (224.0.23.14 defined for IP Version 4 and FF0X:0:0:0:0:0:0:12D for IP Version 6 format) and the port 9214, for this ESG bootstrap session, are registered at IANA; see [13], [22], [23] for respective multicast addresses and port assignments.

9.3.2 Bootstrap Entry Point Discovery over Interactive Channel

The ESG Bootstrap Entry Point of the interactive network can be discovered in the following ways:

- A preconfigured address stored on the terminal.
- Manually entering an ESG Bootstrap Entry Point Information which address to enter is provided out of band.
- Using Terminal Provisioning as specified by OMA DM with an IPDC MO as outlined in clause I.1.
- Using Service Discovery Entry Points with DNS SRV as outlined in clause I.2.

10 ESG Retrieval over Interactive Channel

ESG transport over interactive channel is carried out by querying the corresponding server which is signalled in the ESG Bootstrap Descriptors (see clause 9) or in a DeliveryList (see clause 10.2). It can be used

- to repair the containers received over broadcast;
- to complement the broadcast channel;
- as a standalone means for transporting a consistent ESG.

For the first case, the query is based only on the information from BC, for example, containerID of the container broadcasted over BC to repair. For the second and third case, the query is based on the information given in DeliveryList, which gives a list of fragments or containers from BC or IA that make a consistent ESG.

10.1 Transport Protocol

ESG Query Requests and ESG Query Responses SHALL be transmitted by HTTP [19] over TCP/IP.

10.2 ESG DeliveryList

This clause defines the ESG Delivery List that has the following purposes:

- Signal the List of Fragments or containers the terminal needs to request/receive a consistent ESG instance.
- Signal from wich source fragments and containers SHALL be received.

The ESG DeliveryList shall signal the Fragments or Containers the terminal must receive in order to construct a consistent ESG instance.

A DeliveryList is split into one or more SubLists each of which is identified by its ID and version. A DeliveryList instance should contain all SubLists with at least one SubList signalling a fragment or container set. If there are more than one SubList, the other SubLists are signalled but may be collapsed and requested later.

When requesting a SubList, the response is a DeliveryList instance with the requested SubList expanded.

Purpose of the SubLists:

- Split the DeliveryList into smaller parts for delivery to reduce load on the network.
- To reduce time to process the DeliveryList.
- To organize and prioritize delivery.

For more information about handling the updates please see the clause 10.8.

```
<?xml version="1.0" encoding="utf-8" ?>
<schema xmlns:dl="urn:dvb:ipdc:esgdl:2008" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
            attributeFormDefault="unqualified" elementFormDefault="qualified"
            targetNamespace="urn:dvb:ipdc:esgdl:2008" xmlns="http://www.w3.org/2001/XMLSchema">
  <import schemaLocation="mpeg7_2001.xsd" namespace="urn:mpeg:mpeg7:schema:2001" />
   <element name="DeliveryList" type="dl:DeliveryListType" >
<complexType name="DeliveryListType">
    <sequence>
      <element name="SubList" type="dl:SubListType" maxOccurs="unbounded" />
    </sequence>
    <attribute name="lastupdated" type="unsignedInt" use="required" />
    <attribute name="expirationDate" type="unsignedInt" use="optional" />
    <attribute name="expirationWindow" type="unsignedInt" use="optional" />
    <attribute name="listID" type="anyURI" use="required" />
  </complexType>
  <complexType name="SubListType">
    <sequence>
      <element minOccurs="0" name="IADeliveryChannel" type="dl:IADeliveryChannelType" />
      <element minOccurs="0" name="BCDeliveryChannel" type="dl:BCDeliveryChannelType" />
    </sequence>
    <attribute name="subListID" type="unsignedInt" use="required" />
    <attribute name="priority" type="unsignedInt" use="optional" />
    <attribute name="lastupdatedSL" type="unsignedInt" use="required" />
  </complexType>
  <complexType name="IADeliveryChannelType">
    <sequence>
      <element minOccurs="0" name="IAInitContainer" type="dl:IAContainerType" />
      <element maxOccurs="unbounded" name="IAContainer" type="dl:IAContainerType" />
      <element minOccurs="0" maxOccurs="unbounded" name="Fragment" type="dl:FragmentType" />
    <element name="Source" type="anyURI" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>
  <complexType name="BCContainerType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="BCFragment" type="dl:FragmentType" />
    </sequence>
    <attribute name="containerID" type="anyURI" use="required" />
    <attribute name="version" type="unsignedInt" use="optional" />
    <attribute name="MD5" type="base64Binary" use="required" />
  </complexType>
<complexType name="FragmentType">
      <attribute name="mimeType" type="mpeg7:mimeType" use="optional"/>
      <attribute name="ID" type="anyURI" use="required" />
      <attribute use="optional" name="size" type="unsignedLong" />
      <attribute use="optional" name="transportID" type="unsignedInt" />
      <attribute use="optional" name="version" type="unsignedInt" />
      <attribute use="required" name="MD5" type="base64Binary" />
  </complexType>
  <complexType name="BCDeliveryChannelType">
    <sequence>
      <element minOccurs="0" name="BCCInitContainer" type="dl:BCContainerType" />
      <element maxOccurs="unbounded" name="BCContainer" type="dl:BCContainerType" />
      <element minOccurs="0" maxOccurs="unbounded" name="Fragment" type="dl:FragmentType" />
    </sequence>
  </complexType>
  <complexType name="IAContainerType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="IAFragment" type="dl:FragmentType" />
    </sequence>
    <attribute name="containerID" type="anyURI" use="optional" />
    <attribute name="version" type="unsignedShort" use="optional" />
    <attribute name="size" type="unsignedLong" use="required" />
    <attribute name="MD5" type="base64Binary" use="required" />
  </complexType>
</schema>
```

DeliveryListType semantics

Field	Semantics
lastupdated	the version in NTP timestamp format of the consistent ESG instance represented by this DeliveryList .Updates are tracked by comparing the fragment set in the DeliveryList with the local fragment data base or with a previous DeliveryList.
expirationDate	NTP timestamp when this DeliveryList expires. The terminal should request for a new DeliveryList to check for updates within the expirationWindow.
expirationWindow	Time in seconds that gives the time window for requests of a new DeliveryList.
listID	A unique identifier of the DeliveryList.
SubList	A DeliveryList can be split into one or more SubLists.

SubListType Semantics

Field	Semantics
lastupdatedSL	Version of the subList in NTP timestamp format.
subListID	The ID of a specific SubList, unique within the DeliveryList (listID). The ID value can be zero. The ID may be used to request for a SubList as specified in clause 10.5.1 and is unique within a given DeliveryList.
priority	Priority of the SubList and the Fragments/Containers it describes. The priority tells the recommended order in which a terminal should query for SubLists. The integer value starts from 1 (highest priority) and increments for lower priorities.
IADeliveryChannel	The Interactive Channel according to the IA Access point in the ESG Bootstrap descriptor.
BCDeliveryChannel	The broadcast delivery channel according to the BC Access point in the ESG Bootstrap descriptor.

IADeliveryChannelType Semantics

Field	Semantics
IAInitContainer	Signals the InitContainer available over interactive channel that shall be used to initialize reception of ESG fragment over that channel.
IAContainer	Signals an ESG Container that should be fetched over this Interactive Channel.
Fragment	Signals an ESG Fragment that should be fetched over this Interactive Channel.
AccessPoint	An URL from which Containers or Fragments listed in the IADeliveryChannel shall be requested if present. If this element is not present requests shall use the AccessPoint as signalled in the Bootstrap AccessDescriptor. AccessPoints signalled by this element are in the same context as the AcessPoint in the Bootstrap Access Descriptor from which the DeliveryList has been requested. NOTE: To enable load balancing and recovery from unavailable AccessPoints more than one AccessPoint element may be present.

BCDeliveryChannelType Semantics

Field	Semantics
BCContainer	Signals an ESG Container that should be fetched over this Broadcast Channel.
BCInitContainer	Signals the InitContainer available over Broadcast Channel that shall be used to initialize reception of ESG fragments over that channel.
Fragment	Signals an ESG Fragment that should be fetched over this Broadcast Channel.

BCContainerType Semantics

Field	Semantics
containerID	ESG container ID of an ESG Container that can be fetched over broadcast channel. The ID shall be as specified in clause 7.2.1.
version	Version of an ESG container available over broadcast channel.
md5	MD5 checksum of an ESG container available over broadcast channel.
size	Size of a specific ESG container available over interactive channel. If a Container is compressed it shall tell the uncompressed size of the container.
BCFragment	Specific Fragment over broadcast channel that must be used to build the consistent ESG.

IAContainerType Semantics

Field	Semantics
containerID	ESG container ID of an ESG Container that can be requested over interactive channel. The ID shall be as specified in clause 10.3.
version	Version of an ESG container available over interactive channel.
md5	MD5 checksum of an ESG container available over interactive channel.
size	Size of a specific ESG container available over interactive channel. If a Container is compressed it shall tell the uncompressed size of the container.
IAFragment	Specific Fragment over interactive channel that must be used to build the consistent ESG.

FragmentType Semantics

Field	Semantics
ID	Uniquely identifies an ESG Fragment (e.g. SDPURI for SDP files, or ServiceID for a Service Fragment). NOTE 1: In case of ESG XML Fragments, this maps to the Datamodel ID (e.g. ServiceID).
mimeType	The mimeType of the ESG Fragment.
size	The size of the fragment. NOTE 2: If the fragment is compressed, it is the size before compression.
transportID	The ID assigned for transport of the fragment. For Fragments signalled in IAContainers or BCContainers the transportID is required and maps to the fragment_id in the Fragment Management Information. Fragments signalled in the IAContainer shall be requested with type=container. Fragments signelled outside IAContainer shall be requested with type=fragment from the given AccessPoint.
version	Incrementing integer value signalling the version of the ESG Fragment.
MD5	MD5 checksum of the ESG fragment.

10.3 ESG Container Identification and Version Information using the DeliveryList

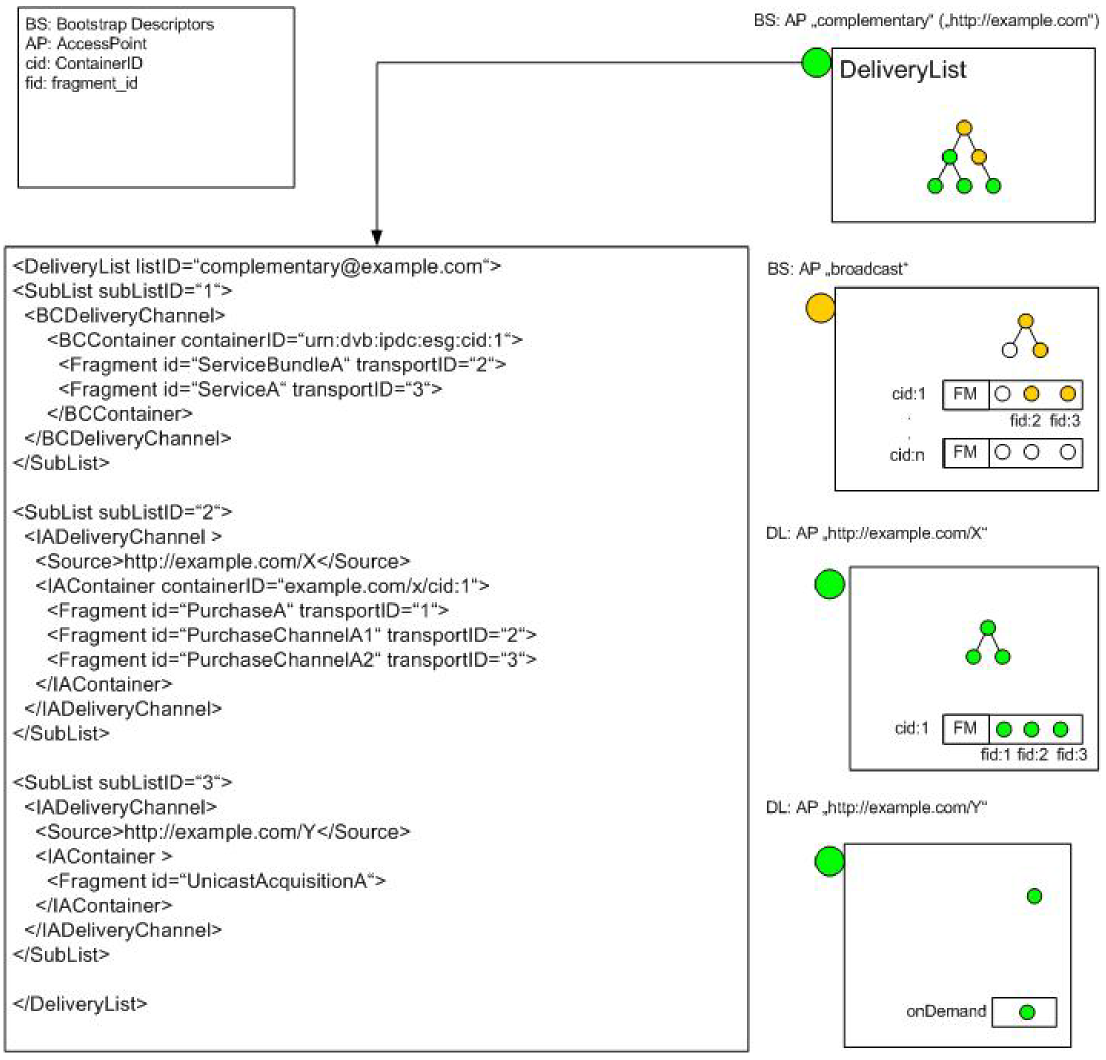
ESG Containers available over interactive channel are signalled in the DeliveryList as specified in clause 10.2.

For delivery over the interactive channel ESG Containers can be either preconfigured or created on demand.

Preconfigured ESG containers hold a specific set of fragments and their updates as specified in clause 7.2.

On Demand Containers are created on the fly when dedicated fragments have been requested. They only serve as encapsulation structure and shall not be used to track fragment updates.

The following figure gives an example for a complementary ESG:



```
<DeliveryList listID="complementary@example.com">
<SubList subListID="1">
 <BCDeliveryChannel>
    <BCContainer containerID="urn:dvb:ipdc:esg:cid:1">
     <Fragment id="ServiceBundleA" transportID="2">
     <Fragment id="ServiceA" transportID="3">
    </BCContainer>
 </BCDeliveryChannel>
</SubList>

<SubList subListID="2">
 <IADeliveryChannel >
   <Source>http://example.com/X</Source>
   <IAContainer containerID="example.com/x/cid:1">
     <Fragment id="PurchaseA" transportID="1">
     <Fragment id="PurchaseChannelA1" transportID="2">
     <Fragment id="PurchaseChannelA2" transportID="3">
   </IAContainer>
 </IADeliveryChannel>
</SubList>

<SubList subListID="3">
 <IADeliveryChannel>
   <Source>http://example.com/Y</Source>
   <IAContainer >
     <Fragment id="UnicastAcquisitionA">
   </IAContainer>
 </IADeliveryChannel>
</SubList>

</DeliveryList>
```

Figure 14: Example of ID signalling in the DeliveryList of a complementing ESG

The rest of this clause describes the versioning of ESG data:

If the FDT Split TOI mechanism as specified in clause 8.1.1.3 is supported, the version attribute of a container, if signalled in the DeliveryList, should be used to identify the right version of a broadcasted container.

If the FDT Split-TOI mechanism is not supported, MD5 checksum SHALL be used to identify the correct container. In the DeliveryList the MD5 checksum SHALL be signalled for containers.

If the MD5 value is signalled in the FDT by the Content-MD5 attribute a terminal can instantly match it with the MD5 value signalled in the DeliveryList to identify the right version of a container. If no Content-MD5 is signalled in the FDT the terminal SHOULD compile the MD5 checksum of received containers or may only use the container identification as signalled by the FDT Content-Location. The latter may result in possible inconsistency if a new version has been announced but the old version is still in the FLUTE carousel and used by the terminal.

After a terminal has identified updates of ESG Fragments from the DeliveryList, it may request for a container carrying only these fragments. The server shall respond with a container carrying the requested fragments or with HTTP 404 if it does not support the request. For OnDemand Containers no ContainerID is signalled in the DeliveryList. Preconfigured Containers shall be signalled with ContainerIDs in the IADeliveryChannel of the DeliveryList and the rules of clause 10.3 apply.

Container identification is signalled in the DeliveryList by containerID attributes for BCDeliveryChannel and IADeliveryChannel. The ContainerIDs may be managed independently for each DeliveryChannel, i.e. ContainerIDs may be unique only in the context of that DeliveryChannel. Therefore the terminal shall be aware of the ContainerID context. The ContainerID as signalled in the DeliveryList is constructed by <context>:<cid> with the context being a URI unique within a DeliveryList and the cid being the integer id number of the container.

NOTE: Containers signalled with the same <context> can be expected to be managed consistently, i.e. cid ranges are not overlapping.

When a Container is requested by its ID signalled in the DeliveryList, the latest Version shall be provided. If no containerID attribute is present in the DeliveryList the server shall support requests for Fragments based on fragmentIDs as specified in clause 10.5.3.

In addition containers can be identified using MD5 signalling as described above. In the response message the Content-MD5 header field shall be used to identify the requested Container.

10.4 ESG Reception Initialization

The DeliveryList contains a reference to an IA Init Container that contains the ESG Init Message specific for the IA delivery. The IA Init Container contains only the IA ESG Init message.

In case the Init Container over broadcast is broken, the Init Container MAY be received over interactive channel by sending a request to the "repair" AccessPoint. In this case the content of the Init Container SHALL be the same and shall be used for initialization of ESG reception over broadcast.

The IA ESG Init Message structure is the same as the BC ESG Init Message structure as specified in clause 6.2 with Indexing Flag always set to 0.

NOTE: The IA Iinit Message for initialization of ESG reception over interactive channel can differ in the complementary case from the BC ESG Init Message and both messages may be needed to correctly initialize the ESG reception and processing over each channel.

In case of ESG requests to a "repair" Access Point, exactly the same ESG Data as over broadcast shall be received and therefore the BC ESG Init Message shall be used. If not previously received by the terminal, it shall be requested from the AccessPoint with "repair" capability (see clause 9.1.2).

In case of complementary delivery, the terminal has to receive one or both InitContainers for each channel as signalled in the ESG DeliveryList.

In case of interactive delivery, the terminal requests the IA ESG InitContainer and shall only use the IA ESG InitMessage.

The encoding signalled in the IA Init Message shall be used in the requests of type=containers (both Container and Fragment-Encoding) or type=fragment (only Fragment-Encoding) posted to the signalled AccessPoint.

10.5 ESG Query Request

This clause defines the ESG Query Request structure (ESG Query Request Message and ESG Query Request Payload).

The terminal shall originate requests. The network shall respond to requests.

The request shall be made using "POST" method of HTTP/1.1.

The "Request-URI" of HTTP POST request shall be set to the interactive AccessPoint URL signalled in the ESG Bootstrap AccessDescriptor.

The parameters associated with the HTTP POST request shall be communicated as key-value pairs following the conventions defined in clause 17.13 of HTML 4.01 [46] for submitting HTML form data by the "POST" method using the "application/x-www-form-urlencoded" encoding type. More specifically, once encoded as "application/x-wwwform-urlencoded", the parameters to be passed from terminal to system shall be communicated in the "message-body" of HTTP/1.1 "Request" message as defined in clause 5 of RFC2616 [19].

Within a single request, the terminal may include multiple key-value pairs. As defined by HTML 4.01 these key-value pairs SHALL be delimited by an "&".

The terminal may assign several values for a certain key. In this case the different values are separated by comma ",".

More information about supported scenarios to request for ESG can be found in the annex G.

The ESG Query Response message is an HTTP Response message as specified in HTTP 1.1 [19].

The response payload will be delivered in the message body of the HTTP response message.

Optionally the response may be compressed with GZip. In that case the content-encoding entity header field is set to Content-Encoding: gzip.

10.5.1 Query for DeliveryLists and SubLists

When the terminal requests DeliveryLists the "message-body" of HTTP/1.1 request SHALL be prefixed with "type=DeliveryList".

Optionally the request may be prefixed with "LASTUPDATED=" and the NTP time of the last version of the DeliveryList.

When terminal requests a SubList by its identifier the "message-body" of HTTP/1.1 request message SHALL contain key-value pair using "subListID" as the key and the subListID of the requested SubList as the value.

The request is sent to an "interactive" or "complementary " AccessPoint associated to the given ESG_URI.

The response to the request SHALL be HTTP/1.1 response with the DeliveryList or SubList in the response body.

The response SHALL have HTTP header field "Content-Type" set to "text/xml".

10.5.1.1 Query for DeliveryList with Capability Signalling

When the terminal requests DeliveryLists, request parameters "accept-container-encoding" and "accept-xmlfragment-encoding" can be used to tell which are the accepted encoding schemes for containers and fragments.

Values to be used in accept-xmlfragment-encoding are "GZip", "BiM" and "Unencoded" as specified in table 2.

Values to be used in accept-container-encoding are "GZip" and "Unencoded".

If none of these parameters is included in the request the server can decide which encoding is used.

The exchanged encodings with this query only affect the IA delivery.

If none of the content-encodings signalled in the requests match the server capabilities the response shall be an HTTP 404 response.

If the server does support 2 or more encodings that the client supports it shall respond with a DeliverList that contains the ID of an IA Init Container with an Init message that matches one of the encodings in the terminal request.

If the server supports exactly one of the encodings it shall respond with a DeliveryList that contains the ID of an IA Init Container with an Init message that matches this encoding.

See clause 10.4. on reception initialization based on the Init Message inside the returned Init Container.

10.5.2 Query for Containers

The terminal SHALL request ESG Container over the Interaction Channel as follows:

When the terminal requests ESG Containers the "message-body" of HTTP/1.1 request SHALL be prefixed with "type=ESGContainer".

When terminal requests ESG Container by its identifier the "message-body" of HTTP/1.1 request message SHALL contain key-value pair using "containerID" as the key and the ContainerID of the requested Container as the value.

Additionally to the containerID the request message shall contain the key-value pair using "accept-container-encoding" as the key and the container encoding as signalled in the Init Message as the value.

If AccessPoint elements are signalled in the DeliveryList one of those URLs shall be used by the terminal to download the container from the dedicated IA Channel.

The response to the request SHALL be HTTP/1.1 response.

The response must contain the following HTTP header parameters:

Content-Type set to "application/vnd.dvb.esgcontainer" if only one container is delivered in the response

Content-Encoding: shall match the accept-container-encoding in the request.

Content-MD5: shall signal the MD5 checksum of the container.

Content-Type: shall be set to Multipart/Mixed when multiple containers are delivered in the response with all of the above fields signalled in each part of the response (after the boundary).

10.5.3 Query for Fragments

The terminal SHALL request ESG Fragments over the Interaction Channel as follows:

When the terminal requests ESG Fragments the "message-body" of HTTP/1.1 request SHALL be prefixed with "type=ESGContainer" or "type=ESGFragment".

When terminal requests ESG Fragments by its identifier, the "message-body" of HTTP/1.1 request message SHALL contain key-value pair using "fragmentID" as the key and the Fragment identifier of the requested Fragment as the value.

When the terminal requests ESG Fragments prefixed with type="ESGContainer" the request message shall additionally to the fragmentID contain:

- the key-value pair using "accept-container-encoding" as the key and the container encoding as signalled in the Init Message as the value;
- the key-value pair using "accept-xmlfragment-encoding" as the key and the fragment encoding as signalled in the Init Message as the value.

When the terminal requests ESG Fragments prefixed with type="ESGFragment" the request message shall additionally to the fragmentID contain:

- the key-value pair using "accept-xmlfragment-encoding" as the key and the fragment encoding as signalled in the Init Message as the value.

The response to the request SHALL be HTTP/1.1 response.

The response must contain the following HTTP header parameters for request type=ESGContainer:

- Content-Type set to "application/vnd.dvb.esgcontainer" if only one container is delivered in the response;
- Content-Encoding: shall match the accept-container-encoding in the request;
- Content-MD5: shall signal the MD5 checksum of the container;
- Content-Type: shall be set to Multipart/Mixed when multiple containers are delivered in the response with all of the above fields signalled in each part of the response (after the boundary).

The response must contain the following HTTP header parameters for type=ESGFragment:

- Content-Type set to mime-type of the ESG Fragment if only one container is delivered in the response.
- Content-Encoding: the encoding of the Fragment. For ESG XML Fragments it shall match the accept-xmlfragment-encoding in the request.
- Content-MD5: shall signal the MD5 checksum of the Fragment.
- Content-Type: shall be set to Multipart/Mixed when multiple Fragments are delivered in the response with all of the above fields signalled in each part of the response (after the boundary).

10.6 Update Handling and Versioning

An ESG server creates a snapshot of an ESG instance scheduled for delivery which is represented by a version of a DeliveryList.

A DeliveryList is structured in SubLists. For optimization, only SubLists that contain updates may be expanded in the DeliveryList, i.e. the DeliveryList always signals all SubLists and their version but only some SubLists signalling updates contain an expanded list of Fragments to reduce delivery size and terminal processing. Therefore updates between two delivery snapshots are signalled in the corresponding SubLists by lastupdatedSL version. Every time the version of a SubList is set to a new timestamp, the version of the DeliveryList is also set to the same timestamp. In case a terminal requests for a specific SubList by SubListID, the server shall send a DeliveryList with the requested SubList expanded.

A Terminal should use the lastupdated attribute when requesting for a DeliveryList to optimize overall response traffic: the ESG server will respond with the latest version of the DeliveryList or with an HTTP 304 Not Modified response with HTTP Expires entity-header set to the time the terminal should again request for a DeliveryList. If the terminal does not send the lastupdated version in the request, it shall always receive the latest version of the DeliveryList.

A terminal shall track updates to the ESG Fragments by comparing the fragment versions or MD5 in a DeliveryList with a previous DeliveryList or the local data set (e.g. in the terminal database).

If the terminal wants to recognize updates as soon as possible it should check more frequently if the DeliveryList has been updated. In general, the DeliveryList provides an Expiration date that provides the point of time recommended to check for updates. Furthermore, if the broadcast part of an ESG has been updated, the terminal can also request the DeliveryList to check if IA components have been updated as well.

Tracking of IA updates:

1) Check DeliveryList lastupdated. If lastupdated is a new version go to 2) else no need for action.

2) Check wich SubLists were updated.

3) Compare versions or MD5 of elements within the SubList:

   - If the MD5 has changed or a version is higher than in a previous DeliveryList or the local database the terminal should request the container or fragments over interactive channel.
     In case of Containers requested by ContainerID (preconfigured containers) only those fragments that have been updated may be processed.

4) If the Content-MD5 in the response does not match the MD5 in the DeliveryList a new DeliveryList shall be requested.

Tracking of BC updates:

1) Check DeliveryList lastupdated. If lastupdated is a new version go to 2) else no need for action.

2) Check wich SubLists were updated.

3) Compare BCContainer versions or MD5 within the SubList.

4) Compare versions or MD5 of elements within the SubList:

   - If the MD5 has changed or a version is higher than in a previous DeliveryList or the local database, the terminal should request the container or auxiliary data over broadcast channel and use only those fragments that are listed in the DeliverList for that container. Only those fragments that have been updated may be processed.

5) After a terminal has identified updates of ESG Fragments from the DeliveryList it may request for a container only encapsulating these fragments.

NOTE: Tracking of deleted fragments: The DL does not explicitly signal what fragments or containers have been deleted. Rather it tells what are the containers and fragments that have to be received. All other containers and fragments not signalled in the DL should be ignored from the transport and be no longer valid if previously received.

Tracking of moved fragments: The DL signals unique FragmentIDs. Therefore a terminal can detect when a fragment has moved from one delivery channel to another and in which container it may be found. ContainerIDs may be managed independently for IA and BC channels. In this case, if a container has moved from one channel to another, it may disappear in one DeliveryChannel and appear in another DeliveryChannel with a different ID and different version (starting from 1). However the IDs of the listed fragments will be the same and the MD5 will be the same if the Fragment has been moved without update.

In case of SDP files (or other data) not encapsulated in Containers: Those files are referenced by DatamodelID (e.g. SDPURI in the Acquisition fragment). No transportID is needed to find the file, because the DatamodelID is identical to the transportID (same URI). If those files are listed in a DeliveryList with version information the terminal can rely on the tracking of version/MD5 change. If version is not given, the terminal will have to track version, itself e.g. by tracking the FDT.

11 Consistency signalling in the DeliveryList

For the cases where an ESG is delivered over interactive channel (complementing or interactive delivery) the DeliveryList as specified in clause 10.2 shall be used to manage consistency.

The DeliveryList provides the consistent set of Fragments for a given ESG at a given time.

An ESG is complete when all Containers/Fragments in the DeliveryList are received.

NOTE: A DeliveryList received from an AccessPoint in the context of a specific ESG manages only consistency of that ESG.

Annex A (informative): TV-Anytime Datatypes

In this annex the datatypes of the TV_Anytime (TS 102 822-3-1 [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="optional"/>
      </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 (informative): MPEG-7 Datatypes

In this annex the datatypes of the ISO/IEC 15938-5 [4] MPEG-7 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*)(\[\d+\])?))|\.)|((attribute::|@)((\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 (normative): Default Classification Schemes

In this annex 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 TS 126 346 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>
```

C.3 TargetAudience

```
<ClassificationScheme uri="urn:dvb:ipdc:esg:cs:targetAudienceTypeCS">
   <Term termID="1">
      <Name xml:lang="en">User Origin</Name>
      <Definition xml:lang="en">Origin of the user with respect to the access network </Definition>
      <Term termID="1.1">
            <Name xml:lang="en">Roaming</Name>
            <Definition xml:lang="en">Roaming user </Definition>
      </Term>
      <Term termID="1.2">
            <Name xml:lang="en">Local</Name>
            <Definition xml:lang="en">Local user</Definition>
      </Term>
   </Term>
</ClassificationScheme>
```

C.4 Protocols

```
<?xml version="1.0" encoding="utf-8"?>
<ClassificationScheme uri="urn:dvb:ipdc:ProtocolCS:2008"
            xmlns="urn:dvb:metadata:schema:dvbCSschema:2007"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xml="http://www.w3.org/XML/1998/namespace"
            xsi:schemaLocation="urn:dvb:metadata:schema:dvbCSschema:2007 ./dvbCSschema.xsd">
   <Term termID="1">
      <Name>HTTP</Name>
      <Definition>Hypertext Transfer Protocol</Definition>
      <Term termID="1.1">
         <Name>HTTP/1.1</Name>
         <Definition>IETF RFC 2616: "Hypertext Transfer Protocol -- HTTP/1.1"</Definition>
      </Term>
   </Term>
   <Term termID="2">
      <Name>PSS</Name>
      <Definition>ETSI TS 126 234: "Universal Mobile Telecommunications System (UMTS); Transparent
            end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs
            (3GPP TS 26.234)"</Definition>
      <Term termID="2.1">
         <Name>PSS/7.5</Name>
         <Definition>ETSI TS 126 234 V7.5: "Universal Mobile Telecommunications System (UMTS);
            Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs
            (3GPP TS 26.234)"</Definition>
      </Term>
   </Term>
   <Term termID="3">
      <Name>RTSP</Name>
      <Definition>IETF RFC 2326: "Real Time Streaming Protocol (RTSP)"</Definition>
   </Term>
   <Term termID="4">
      <Name>FLUTE</Name>
      <Definition>IETF RFC 3926: "FLUTE - File Delivery over Unidirectional Transport"</Definition>
      <Term termID="4.1">
         <Name>IPDC FLUTE Profile</Name>
         <Definition>Extensions defined in DVB IPDC for ESG (ETSI TS 102 471 v1.3.1) and
            Notification (ETSI TS 102 832 V1.1.1)</Definition>
      </Term>
   </Term>
   <Term termID="5">
      <Name>OMA Push</Name>
      <Definition>Push Access Protocol</Definition>
      <Term termID="5.1">
         <Name>OMA Push/2.2</Name>
         <Definition>OMA Push V2.2: "Push Access Protocol"</Definition>
      </Term>
   </Term>
</ClassificationScheme>
```

Annex D (normative): 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.
- If an element is declared as a derived type for which definition is unknown and cannot be resolved, the application should process the portion of the element (i.e. sub-elements and attributes) matching the element's base type definition known by the application, and should ignore any other sub-elements or attributes not matching this base type definition.

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 (informative): ESG Init Message

This clause contains examples of the ESG init Message.

E.1 Default ESG Init Message

Table E.1 is an example ESG Init Message that conforms to the BiM profile specified in the 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 Initialization message.
`}`			

Table E.2 is an example ESG Init Message when textual XML representation is used with gzip encoding specified in the present 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 Initialization message.
`}`			

E.2 Example of a DecoderInit message

Table E.3 is an example DecoderInit message that conforms to the BiM profile specified in the present 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	0x15	21 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

Table E.4 is an example DecoderInit message that conforms to the BiM profile specified in the present 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	1111	
`AdvancedFeaturesFlag_Length`	8	0x01	Eight Flags signalled.
`ReservedBitsZero`	7	0000000	
`ContextPathTableFlag`	1	1	A ContextPathTable is signalled.
`NumberOfSchemas`	8	0x01	Only one schema used.
`{`			
`SchemaURI_Length[0]`	8	0x15	21 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 Signalled.
`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.
`}`			

Field	No. of bits	Value	Semantic
InitialDescription_Length	8	0x00	The initial root description is conveyed in the TVAMain fragment.
}			

Annex F (informative): Bitmap Image Formats

In this annex the image formats are referenced which are supported as ESG auxiliary data.

F.1 JPEG

JPEG as defined in ISO/IEC 10918-1 [28] using the JFIF [47] file exchange format.

F.2 PNG

PNG is defined as in PNG [30].

F.3 GIF

GIF is defined as in GIF 89a [31].

Annex G (informative): Scenarios for interactive delivery of the ESG

This annex details the three possible scenarios for requesting ESG Fragments and Containers over the IA Channel. Each scenario provides a table overview of its associated use case, along with the request steps and arguments required by the terminal, and the responses returned by the server. Query arguments in square brackets are optional. The Delivery List is defined in clause 10.2.

G.1 Retrieve an ESG over IA Channel

The terminal can request a complete ESG over IA Channel from an "interactive" AccessPoint. This ESG can be "branded" (i.e. extended or modified compared to an ESG that is received over BC) or it can be the same as delivered over broadcast.

Request Use Case	Step	Request for	Query Arguments	Response message
Request a complete ESG over interactive channel	a	DeliveryList	[LASTUPDATED]	Terminal may request a DeliveryList using the LASTUPDATED value from a previous obtained DeliveryList. As a response, terminals will receive either a new version of the DeliveryList or an HTTP 304 Not Modified response with Retry-After header field. Terminals may request a DeliveryList without using a LASTUPDATED value. In response, a terminal will receive the DeliveryList of the latest version.
	b1 (may be used only if containerID is signalled in the DeliveryList)	Container	ContainerID [Container Version] [MD5]	The container retrieved from a query using containerID and optionally versioning information (either version or MD5). When the versioning information is provided as an argument, the server either sends back a container that is more recent than the current version on the terminal or HTTP 304 Not Modified Response with Retry-After header field.
	b2 (may be used only if sourceURL is signalled in the DeliveryList)	Container via source URL in the DeliveryList	ContainerID [Container Version] [MD5]	Same as step b1. NOTE: The URL is not an argument. This step may be repeated numerous times (with various URLs) in order to get all needed containers.
	b3	Fragments	FragmentID [fragment Version] [MD5]	Similar to step b1, the server either sends back one or more normal ESG containers encapsulating the requested fragments that are more recent than the current version on the terminal or HTTP 304 Not modified response with Retry-After header field.

A terminal requests for a complete ESG of a certain ESG provider over the Interactive Channel with the following two steps:

- Request for a DeliveryList;
- Request for necessary ESG Containers or Fragments by IDs given in the DeliveryList.

Terminals wishing to update an existing ESG can request a DeliveryList using the last obtained LASTUPDATED value. As a response, terminals will receive an exhaustive DeliveryList or an HTTP 304 Not Modified response with Retry-After header filed. Updates are tracked by comparing container and fragment versions or MD5 as specified in clause 9.8 of the present document.

The following rules apply for matching versions of the DeliveryList by the lastupdated filed between terminal and server:

- server version = terminal version: HTTP 304 Not Modified response with Retry-After header field;
- server version > terminal version: The server provides the updated DeliveryList with the new version in the lastupdated field;
- server version < terminal version: obviously the terminal has received a wrong DeliveryList before or some other error occurred. The server replies with the latest version of the DeliveryList.

Terminals without an existing ESG or those wishing to completely refresh their ESG can request a DeliveryList without using a LASTUPDATED value as a parameter. In response, a terminal will receive the DeliveryList of the latest version. Terminals can then ask for all containers listed in the DeliveryList.

G.2 Repair the Broadcast ESG

A terminal can query for the same Containers that are delivered over broadcast if the containerIDs are known from the FDT. Therefore it can repair or even request all broadcasted containers from the "repair" AccessPoint.

Request Use Case	Step	Request for	Query Arguments	Response message
Request an ESG Container that was lost over broadcast	a	Container	Container ID	The most recent version of the container that was distributed on the BC.

To repair a Broadcast ESG, a terminal requests for containers using the necessary Container IDs found in the FDT of the broadcast delivery.

G.3 Retrieving a Complementary ESG

Some servers may provide only complementary parts to an ESG delivered over broadcast. For example additionally to the broadcast acquisition a unicast acquisition may be offered to receive a service also over 3G network.

The AccessPoint to receive these parts is the one for which "complementary" capability is signalled.

Request Use Case	Step	Request for	Query Arguments	Response message
Request the complementing parts over IA channel	a	DeliveryList	[LASTUPDATED]	Response as G.1 step a.
	b1	Container	ContainerID [Container Version] [MD5]	Response as G.1 step b1.
	b2	Container via source URL in the DeliveryList	ContainerID [Container Version] [MD5]	Response as G.1 step b2.
	b3	Fragments	FragmentID [Fragment Version] [MD5]	Response as G.1 step b3.

Complementing ESG information may be provided over the Interactive Channel. A terminal requests this information in two steps:

- Request a DeliveryList of ESG Containers as specified in clause G.1.
- Requests for ESG Containers or Fragments available over interactive channel as signalled in the DeliveryList.

Annex H (normative): Version 1 ESG Bootstrap Descriptors

This clause contains Version 1 ESG Bootstrap Descriptors.

H.1 Version 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>
      <attribute name="format" type="anyURI" use="optional" default="urn:dvb:ipdc:esg:2005"/>
   </complexType>

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

ESGProviderDiscovery Descriptor Semantics

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.
format	Specifies the format of the ESG provided by this ESG provider. This value shall be set to urn:dvb:ipdc:esg:2005 for ESG compliant to the present document.

H.2 Version 1 ESGAccessDescriptor

Version 1 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
`}`		

Version 1 ESGAccessDescriptor Semantics

Field	Semantics
n_o_ESGEntries	Specifies the number of ESGEntries in which access information to ESGs is signalled.
ESGEntryVersion	Specifies the version of the ESG Entry Specification. The value shall be set to "1". NOTE 1: This version is incremented if the specification of ESG Entry is changed in a not forward compatible way. NOTE 2: 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 3: 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 clause 8.4). If set to "0" specifies that a FLUTE session is described which contains all ESG Containers of that ESG (see clause 8.3).
IPVersion6	If set to "1" specifies that the SourceIPAddress and the DestinationIPAddress are signalled according to IP version 6. If set to "0" specifies that the SourceIPAddress and the DestinationIPAddress are signalled 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 signalled by the IPVersion6 field.
DestinationIPAddress	Specifies the destination IP address of the FLUTE session transporting the ESG. The IPVersion is signalled 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 TS 102 472 [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".

H.3 Bootstrap version handling

This clause defines how different versions of ESG Bootstrap Descriptors shall be handled.

Server behaviour: Implementations according to the present document shall always provide Version 2 Bootstrap information. If legacy terminals should be supported additionally Version 1 ESGBootstrapDescriptors should be delivered.

Terminal behaviour: A terminal can distinguish between Version 1 and Version 2 bootstrap information from FDT parameters in the ESG Bootstrap session. Depending on the terminal capabilities it should select either Version 1 or Version 2: terminals implementing version 1.3.x of ESG specification shall select Version 2 Bootstrap information. It is assumed that legacy terminals always use Version 1 bootstrap descriptors. Terminals aware of Version 2 bootstrap descriptors but not implementing those shall always select Version 1.

Annex I (normative): Entry Point Discovery

I.1 Entry point Discover using OMA DM (informative)

DM is an OMA enabler used to provision terminals with information needed for them to access the system. DM MO can be used to (re)provision terminals after events invalidating previously provisioned information. The IPDC MO enables a DM Server to:

- pre-provision entry points for home and foreign networks with roaming partners:
  - A Terminal can select one of the entry points provisioned by a DM server by leveraging location information available from mobile network or by descriptive information that can be found in the MO.
- provision a configuration dynamically:
  - When the terminal is e.g. in a different country and does not find a pre-provisioned entry point or does not know which entry point to choose, a terminal initiates a DM request providing its current location. The server can then add an entry point in the MO structure or point the active entry point the terminal should use.

The MO tree is structured in 2 main parts:

- TerminalInformation:
  - The TerminalInformation can be read by a DM Server to get the IPDCOperator and current Location of a terminal. The location may change when the terminal is moving and should be used only when no matching EntryPoint configuration can be found and the terminal initiates the DM session. In this case the terminal will set location parameters before the management session starts. The DM server can either add or change parameters for an EntryPoint in the current terminal location and will afterwards reset the location parameters to null and mark the relevant EntryPoint as "active" that the terminal knows which one was added/changed and should be used.
- EntryPoints:
  - An EntryPoint contains the parameters to find an IPDC Bootstrap Entry Point, access the Network and ESG of roaming partners for the various regions. The Home EntryPoint is marked. A terminal is able to select the right EntryPoint by mapping it's mobile network information with the corresponding region information in the Entry Point structure or by selecting it manually (e.g. via User Interface) based on an Country/City description in the structure.
    The information in the entry points is rather static (it can however be updated by the server) and should reflect the DVB-H networks in the regions where roaming partners of the terminals home operator are available. Some information like the ESG may not be provisioned because it is unknown or expected to change often. In some cases it may however be provided to give the terminal a preference when it has different options that are well known, for example when the EntryPoint is of the terminals home operator.

I.1.1 The IPDC Management Object Tree (normative)

OMA DM Version 1.2 [48] shall be used with MO Registration Identifier: urn:dvb:mo:ipdc_esg:1.3.1.

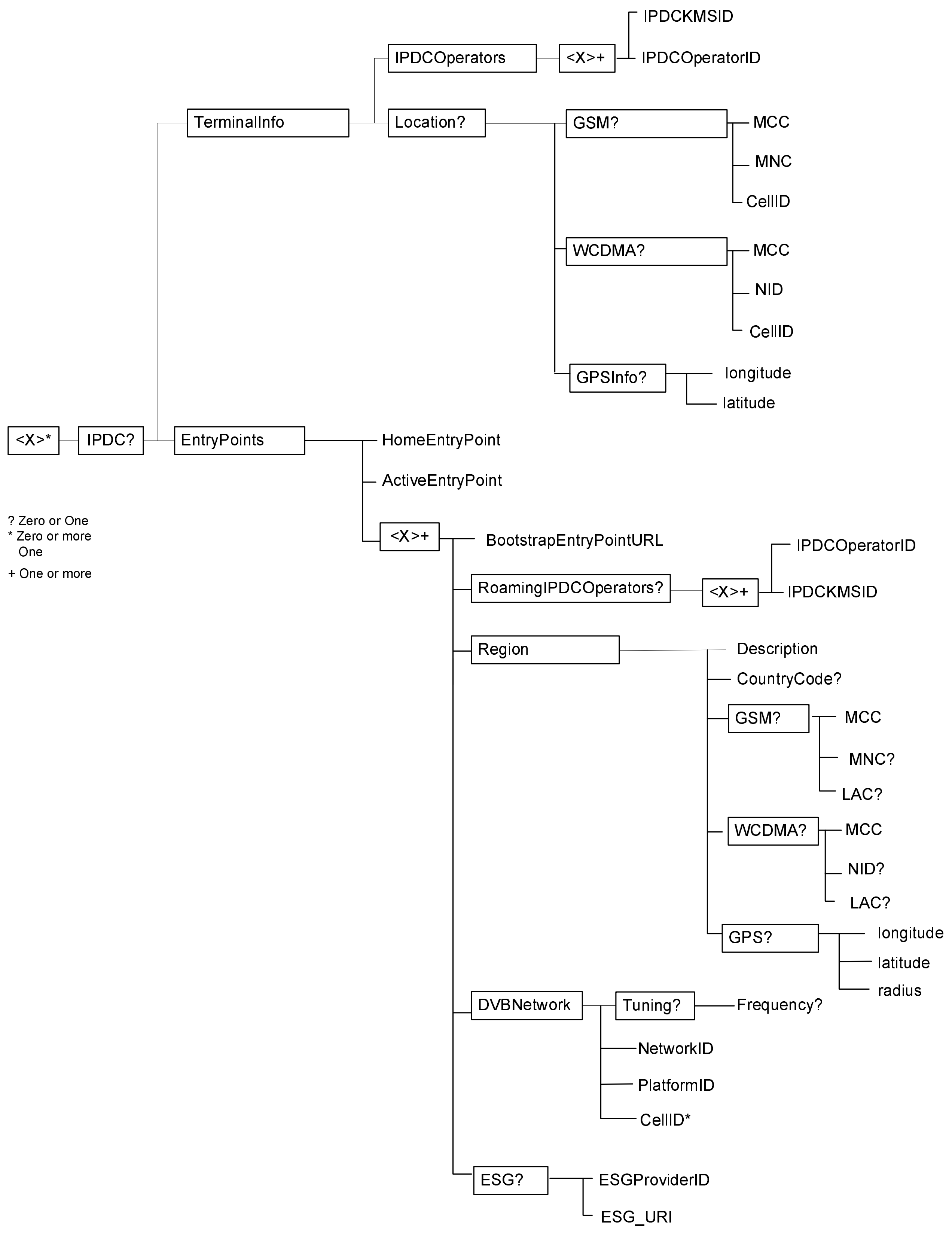


Figure I.1

<X>

This interior node acts as a placeholder for the IPDC Management Object root node.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrMore	node	Get

<X>/IPDC

This interior node is the IPDC Management Object root node

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/TerminalInfo

This interior node contains information about the terminal

Status	Occurrence	Format	Min. Access Types
Required	One	node	Get

<X>/IPDC/TerminalInfo/IPDCOperators

This interior node contains information about the IPDCOperator configurations of the terminal.

Status	Occurrence	Format	Min. Access Types
Required	One	node	Get

<X>/IPDC/TerminalInfo/IPDCOperators/<X>

This interior node acts as a placeholder for different IPDCOperator profiles of the terminal. At least one home shall be present.

Status	Occurrence	Format	Min. Access Types
Required	OneOrMore	node	Get, Add, Replace, Delete

<X>/IPDC/TerminalInfo/IPDCOperators/<X>/IPDCOperatorID

This leaf node contains the IPDCOperatorID. An IPDC Operator is an entity managing key streams. The value is a 16bit integer as present in the RoamingInformationDescriptor. See TS 102 611 [i.5] for more information.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Add, Replace

<X>/IPDC/TerminalInfo/IPDCOperators/<X>/IPDCKMSId

This leaf node contains the IPDC KMSId. The value is a 16bit integer of the Key Management System.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Add, Replace

<X>/IPDC/TerminalInfo/Location

This interior node contains information about the location of the terminal. The location information is provided by the terminal and should be updated before the terminal initiates the management session with the DM server. If present the DM server can read this information to provision the terminal with an Entry Point suitable for the location given here.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/TerminalInfo/Location/GSM

This interior node contains location information accessible from a GSM mobile network.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/TerminalInfo/Location/GSM/MCC

This leaf node contains the mobile country code. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Add, Replace

<X>/IPDC/TerminalInfo/Location/GSM/MNC

This leaf node contains the mobile network code. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Add, Replace

<X>/IPDC/TerminalInfo/Location/GSM/CellID

This leaf node contains the mobile cell id. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get Replace

<X>/IPDC/TerminalInfo/Location/WCDMA

This interior node contains location information accessible from W-CDMA UMTS mobile network.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get, Add, Replace

<X>/IPDC/TerminalInfo/Location/WCDMA/MCC

This leaf node contains the mobile country code. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/TerminalInfo/Location/WCDMA/NID

This leaf node contains the network ID. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/TerminalInfo/Location/WCDMA/CellID

This leaf node contains the mobile cell id. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/TerminalInfo/Location/GPSInfo

This interior node contains information acquired from global positioning system.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/TerminalInfo/Location/GPSInfo/longitude

This leaf node contains the GPS longitude of the terminal position. The integer value is the decimal longitude degrees. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/TerminalInfo/Location/GPSInfo/latitude

This leaf node contains the GPS latitude of the terminal position. The integer value is the decimal latitude degrees. The value shall be set to null after it was read.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints

This interior node acts a container for IPDC Entry Points. An Entry Point contains the information a terminal needs to access the system in a specific location.

Status	Occurrence	Format	Min. Access Types
Required	One	node	Get

<X>/IPDC/EntryPoints/HomeEntryPoint

This leaf node carries a link to the home EntryPoint. An EntryPoint marked as HomeEntryPoint shall not be changed by an other DM Server than the Home DM Server.

Status	Occurrence	Format	Min. Access Types
Required	One	chr	Get, Replace

<X>/IPDC/EntryPoints/ActiveEntryPoint

This leaf node carries a link to the active entry point. An EntryPoint marked as ActiveEntryPoint contains the configuration a terminal should use at it's current location.

Status	Occurrence	Format	Min. Access Types
Required	One	chr	Get, Replace

<X>/IPDC/EntryPoints/<X>

This interior node acts as a placeholder for IPDC EntryPoint information. At least the HomeEntryPoint should be present.

Status	Occurrence	Format	Min. Access Types
Required	OneOrMore	node	Get, Replace, Delete, Add

<X>/IPDC/EntryPoints/<X>/BootstrapEntryPointURL

This leaf node contains the URL of the Bootstrap Entry Point. The Bootstrap Entry Point URL is needed to request Bootstrap Information as specified in clause 9.2.2 over interactive channel.

Status	Occurrence	Format	Min. Access Types
Required	One	chr	Get, Replace

<X>/IPDC/EntryPoints/<X>/RoamingIPDCOperators

This interior node contains parameters of an IPDCOperator that has a roaming agreement with the terminals Home IPDCOperator. The information is needed to select the right IPDCOperator configuration to access service described in the ESG when roaming. This node is not applicable for the Home EntryPoint.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/EntryPoints/<X>/RoamingIPDCOperators/<x>

This interior node acts as a placeholder for different roaming partner IPDCOperator parameters.

Status	Occurrence	Format	Min. Access Types
Required	OneOrMore	node	Get, Add, Replace, Delete

<X>/IPDC/EntryPoints/<X>/RoamingIPDCOperators/<x>/IPDCOperatorID

This leaf node contains the IPDCOperatorID.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Add, Replace

<X>/IPDC/EntryPoints/<X> RoamingIPDCOperators/<x>/IPDCKMSId

This leaf node contains the IPDC KMSId.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Add, Replace

<X>/IPDC/EntryPoints/<X>/Region

This interior node contains information on the region of the EntryPoint. The Region information enables the terminal to select the right EntryPoint for it's location. The Region information is not expected to provide a 100% accurate overlay description of the access area of the DVB Network of an EntryPoint. Rather it is used to differentiate multiple EntrryPoints available e.g. in a country in a best effort way.

Status	Occurrence	Format	Min. Access Types
Required	One	node	Get, Add, Replace

<X>/IPDC/EntryPoints/<X>/Region/Description

This leaf node contains a description of the region that can be presented to the user. The description should be meaningful to differentiate between multiple Entry Points and to enable a user to select the right configuration even if no other location/region information is available or can be accessed. For example it should describe the country and city of the entry point.

Status	Occurrence	Format	Min. Access Types
Required	One	chr	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/CountryCode

This leaf node carries an ISO 3166 country code (see [i.6] and [i.7]). The value shall contain ISO 3166-1 [i.6] country code and optionally ISO 3166-2 [i.8] subdivision codes to refine the region.

This CountryCode is useful for devices that does not have access to a mobile network information and therefore is a more generic region description.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrMore	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/GSM

This interior node contains GSM mobile network parameters to describe the region of the EntryPoint. A terminal can acquire location information from the mobile network and search for the according EntryPoint.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/EntryPoints/<X>/Region/GSM/MCC

This leaf node contains the mobile country code.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/GSM/MNC

This leaf node contains the mobile network code.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/GSM/LAC

This leaf node contains the GSM Location Area Code. This node may be present when it describes the region of an IPDCEntryPoint in a suitable way, i.e. if it enables to differentiate multiple EntryPoints in one country.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/WCDMA

This interior node contains W-CDMA mobile network parameters to describe the region of the entry point.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/EntryPoints/<X>/Region/WCDMA/MCC

This leaf node contains the mobile country code.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region//WCDMA/NID

This leaf node contains the network ID.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/WCDMA/LAC

This leaf node contains the mobile cell id. This node may be present when it describes the region of an IPDCEntryPoint in a suitable way, i.e. if it enables to differentiate multiple EntryPoints in one country.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/GPS

This interior node describes the region of an EntryPoint with parameters of Global Positioning System.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get

<X>/IPDC/EntryPoints/<X>/Region/GPS/longitude

This leaf node contains the GPS longitude of the EntryPoint. The integer value is the decimal longitude degrees.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/GPS/latitude

This leaf node contains the GPS latitude of the EntryPoint. The integer value is the decimal latitude degrees.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/Region/GPS/radius

This leaf node contains the radius around the GPS position of the EntryPoint. The value contains the radius in decimal km.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>DVBNetwork

This interior node contains access parameters of the DVB Network.

Status	Occurrence	Format	Min. Access Types
Required	One	node	Get

<X>/IPDC/EntryPoints/<X>/DVBNetwork/Tuning

This interior node describes tuning parameters to access the DVB Network.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrMore	node	Get

<X>/IPDC/EntryPoints/<X>/DVBNetwork/Tuning/Frequency

This leaf node carries the centre frequency of the DVB-H channel a terminal shall tune to.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/DVBNetwork/DVBNetworkID

This leaf node provides the DVB network ID. It shall be used in combination with Platform ID if the latter is not globally unique. A terminal may further know the networkID it is currently connected to and use it to select this entry point.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/DVBNetwork/DVBCellID

In some rare cases an entry point may differ between DVB Cells. In order to provision this to a terminal this leaf node provides the DVB cell IDs.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrMore	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/DVBNetwork/PlatformID

This leaf node provides the PlatformID. If the PlatformID is not globally unique it shall be used in combination with DVBNetworkID.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/ESG

This interior node contains information about the ESG a terminal should select.

Status	Occurrence	Format	Min. Access Types
Required	ZeroOrOne	node	Get, Add, Replace, Delete

<X>/IPDC/EntryPoints/<X>/ESG/ProviderID

This leaf node carries the ProviderID of an ESG Provider the terminal should select. If an ESG_URI is provisioned this node is not needed.

Status	Occurrence	Format	Min. Access Types
Required	One	int	Get, Replace

<X>/IPDC/EntryPoints/<X>/ESG/ESG_URI

This leaf node carries the ESG_URI of an ESG the terminal should select. A ESG_URI may be provisioned if it is well known and expected to be valid for a longer time. E.g. the Home provider may provision this to terminals for quick access to an ESG in the home location.

Status	Occurrence	Format	Min. Access Types
Required	One	chr	Get, Replace

I.2 Entry point Discover using DNS SRV

In this clause, it is described the method of discovering entry point using DNS SRV based on DNS SRV [37] (see TS 102 034 [i.8]: Transport of MPEG-2 Based DVB Services over IP Based Networks clause 5.2.4 for a similar practise). To find the entry point, terminal looks up service of name:

_dvbipdcservdisc._tcp.mnc<MNC>.mcc<MCC>.gprs

using DNS SRV, where MNC represents mobile network code and MCC represents mobile country code (see 3GPP TS 23.003 [i.9] "Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 7)). The mnc and mcc of the home network of the user can be obtained from the IMSI stored on the SIM card. The service name will be registered by DVB (see the list of service names and the procedure for registration of new names is given in http://www.dns-sd.org/ServiceTypes.html).

In case of roaming, the terminal can choose to access the bootstrap entry point of:

- its home network operator using the previous mechanism (i.e. constructing the operator identifier from information within the SIM card);
- or the visited network by constructing the operator identifier of the visited network from the broadcasted information within that network.

The choice of .gprs as the domain name requires the hosting of the bootstrap server in the PLMN (Public Land Mobile Network), since the DNS query is done in a private DNS network internal to PLMNs.

Annex J (normative): ESG Schema Definitions

J.1 2008 ESG Schema Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:esg="urn:dvb:ipdc:esg:2005"
            xmlns:esg2="urn:dvb:ipdc:esg:2008" xmlns:tva="urn:tva:metadata:2005"
            xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:dvb:ipdc:esg:2008"
            elementFormDefault="qualified" attributeFormDefault="unqualified">
   <import namespace="urn:dvb:ipdc:esg:2005" schemaLocation="dvb_ipdc_esg_2005.xsd"/>
   <import namespace="urn:mpeg:mpeg7:schema:2001" schemaLocation="mpeg7_2001.xsd"/>
   <import namespace="urn:tva:metadata:2005" schemaLocation="tva_2005.xsd"/>
   <import namespace="http://www.w3.org/XML/1998/namespace" schemaLocation="xml.xsd"/>
   <complexType name="AcquisitionExtensionType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg:AcquisitionType">
            <attribute name="DeliveryChannel" type="esg2:DeliveryChannelType"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="NotificationPrivateDataType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg:PrivateDataType">
            <sequence>
               <element name="NotificationBinding">
                  <complexType>
                     <sequence>
                        <element name="NotificationType" type="unsignedShort"/>
                        <element name="ComponentID" type="esg2:ComponentIDType"/>
                     </sequence>
                  </complexType>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="UnicastAccessType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg:SessionDescriptionBaseType">
            <sequence>
               <element name="Profile" type="tva:ControlledTermType" minOccurs="0"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="PollDownloadType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg2:UnicastAccessType">
            <choice>
               <element name="RegistrationAccess" type="esg2:HTTPAccessType"/>
               <element name="PollAccess">
                  <complexType mixed="false">
                     <complexContent mixed="false">
                        <extension base="esg2:PollAccessBaseType"/>
                     </complexContent>
                  </complexType>
               </element>
            </choice>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="PollAccessBaseType" mixed="false">
      <complexContent mixed="false">
         <extension base="esg2:HTTPAccessType">
            <attribute name="pollInterval" type="unsignedInt" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="PushDownloadType" mixed="false">
      <complexContent mixed="false">
```

```
            <extension base="esg2:UnicastAccessType">
               <sequence>
                  <element name="RegistrationAccess" type="esg2:HTTPAccessType"/>
               </sequence>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="HTTPAccessType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg2:UnicastAccessType">
               <sequence>
                  <element name="HTTPAccessServerURL" type="anyURI"/>
               </sequence>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="RTSPAccessType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg2:UnicastAccessType">
               <choice>
                  <element name="RTSPAccessServerURL" type="anyURI" minOccurs="0"/>
                  <sequence>
                     <element name="InlineSDP" type="esg:SDPType"/>
                     <element name="SDPURI" type="anyURI" minOccurs="0"/>
                  </sequence>
                  <element name="SDPURL" type="anyURI"/>
               </choice>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="PSSAccessType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg2:RTSPAccessType"/>
         </complexContent>
      </complexType>
      <complexType name="DefaultNotificationSessionType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg:SessionDescriptionBaseType">
               <attribute name="isEDN" type="boolean" use="required"/>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="ComponentCharacteristicExtType" abstract="true" mixed="false">
         <complexContent mixed="false">
            <extension base="esg:ComponentCharacteristicType">
               <attribute name="ComponentID" type="esg2:ComponentIDType" use="required"/>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="VideoComponentExtType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg:VideoComponentType">
               <attribute name="componentID" type="esg2:ComponentIDType"/>
            </extension>
         </complexContent>
      </complexType>

      <complexType name="AudioComponentExtType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg:AudioComponentType">
               <attribute name="componentID" type="esg2:ComponentIDType"/>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="FileDownloadComponentExtType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg:FileDownloadComponentType">
               <attribute name="componentID" type="esg2:ComponentIDType"/>
            </extension>
         </complexContent>
      </complexType>
      <complexType name="NotificationComponentType" mixed="false">
         <complexContent mixed="false">
            <extension base="esg2:ComponentCharacteristicExtType">
               <sequence>
                  <element name="NotificationApplicationInit"
               type="esg2:NotificationApplicationInitType" minOccurs="0" maxOccurs="unbounded"/>
               <element name="NotificationComponentIDRef" type="esg2:ComponentIDType"
            minOccurs="0"/>
            </sequence>
            <attribute name="ReplaceSDP" type="boolean" use="required"/>
         </extension>
      </complexContent>
   </complexType>
   <complexType name="NotificationApplicationInitType">
      <sequence>
         <element name="NotificationType" type="unsignedShort"/>
         <element name="NotificationMIMEType" type="mpeg7:mimeType"/>
         <element name="ComponentIDRef" type="esg2:ComponentIDType" minOccurs="0"/>
         <element name="ContentLocation" type="anyURI" minOccurs="0"/>
      </sequence>
   </complexType>
   <simpleType name="ComponentIDType">
      <restriction base="string">
         <pattern value="[!#\$%&amp;‘\*\+-\.0-9A-Z\^_`a-z\{\|\}~]+"/>
      </restriction>
   </simpleType>
   <simpleType name="DeliveryChannelType">
      <restriction base="string">
         <enumeration value="Broadcast"/>
         <enumeration value="Interactive"/>
         <enumeration value="Both"/>
      </restriction>
   </simpleType>
</schema>
```

J.2 2005 ESG Schema Definition

```
<?xml version="1.0" encoding="utf-8"?>
<schema xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:esg="urn:dvb:ipdc:esg:2005"
            xmlns:tva="urn:tva:metadata:2005" attributeFormDefault="unqualified"
            elementFormDefault="qualified" targetNamespace="urn:dvb:ipdc:esg:2005"
            xmlns="http://www.w3.org/2001/XMLSchema">
  <import schemaLocation="xml.xsd" namespace="http://www.w3.org/XML/1998/namespace" />
  <import schemaLocation="mpeg7_2001.xsd" namespace="urn:mpeg:mpeg7:schema:2001" />
  <import schemaLocation="tva_2005.xsd" namespace="urn:tva:metadata:2005" />
  <element name="ESGMain" type="esg:ESGMainType" />
  <complexType name="ESGMainType">
    <sequence>
      <element minOccurs="0" name="CopyrightNotice" type="string" />
      <element minOccurs="0" name="ClassificationSchemeTable"
            type="tva:ClassificationSchemeTableType" />
      <element minOccurs="0" name="ESG" type="esg:ESGType" />
    </sequence>
    <attribute default="en" ref="xml:lang" use="optional" />
    <attribute name="publisher" type="string" use="optional" />
    <attribute name="publicationTime" type="dateTime" use="optional" />
    <attribute name="rightsOwner" type="string" use="optional" />
  </complexType>
  <complexType name="ESGType">
    <sequence>
      <element minOccurs="0" name="ContentTable" type="esg:ContentTableType" />
      <element minOccurs="0" name="ScheduleEventTable" type="esg:ScheduleEventTableType" />
      <element minOccurs="0" name="ServiceTable" type="esg:ServiceTableType" />
      <element minOccurs="0" name="ServiceBundleTable" type="esg:ServiceBundleTableType" />
      <element minOccurs="0" name="PurchaseTable" type="esg:PurchaseTableType" />
      <element minOccurs="0" name="PurchaseChannelTable" type="esg:PurchaseChannelTableType" />
      <element minOccurs="0" name="AcquisitionTable" type="esg:AcquisitionTableType" />
    </sequence>
  </complexType>
  <complexType name="ContentTableType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="Content" type="esg:ContentType" />
    </sequence>
  </complexType>
  <complexType name="ScheduleEventTableType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="ScheduleEvent" type="esg:ScheduleEventType"
            />
    </sequence>
  </complexType>
  <complexType name="ServiceTableType">
    <sequence>
```

```
      <element minOccurs="0" maxOccurs="unbounded" name="Service" type="esg:ServiceType" />
    </sequence>
  </complexType>
  <complexType name="ServiceBundleTableType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="ServiceBundle" type="esg:ServiceBundleType"
            />
    </sequence>
  </complexType>
  <complexType name="PurchaseTableType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="Purchase" type="esg:PurchaseType" />
    </sequence>
  </complexType>
  <complexType name="PurchaseChannelTableType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="PurchaseChannel"
            type="esg:PurchaseChannelType" />
    </sequence>
  </complexType>
  <complexType name="AcquisitionTableType">
    <sequence>
      <element minOccurs="0" maxOccurs="unbounded" name="Acquisition" type="esg:AcquisitionType" />
    </sequence>
  </complexType>
  <complexType name="ESGIDRefType">
    <attribute name="IDRef" type="anyURI" />
  </complexType>
  <complexType name="AcquisitionRefType">
    <complexContent mixed="false">
      <extension base="esg:ESGIDRefType">
        <sequence>
          <element minOccurs="0" maxOccurs="unbounded" name="Label" type="mpeg7:TextualType" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="ServiceRefType">
    <complexContent mixed="false">
      <extension base="esg:ESGIDRefType">
        <attribute name="serviceNumber" type="unsignedShort" />
      </extension>
    </complexContent>
  </complexType>
  <complexType name="RelatedMaterialType">
    <sequence>
      <element minOccurs="0" name="HowRelated" type="tva:ControlledTermType" />
      <element name="MediaLocator" type="mpeg7:MediaLocatorType" />
      <element minOccurs="0" maxOccurs="unbounded" name="PromotionalText" type="mpeg7:TextualType"
            />
      <element minOccurs="0" maxOccurs="unbounded" name="PromotionalMedia"
            type="mpeg7:TitleMediaType" />
    </sequence>
  </complexType>
  <complexType name="ProviderType">
    <sequence>
      <element minOccurs="0" name="ProviderURI" type="anyURI" />
      <element maxOccurs="unbounded" name="ProviderName" type="mpeg7:TextualType" />
      <element minOccurs="0" name="ProviderLogo" type="mpeg7:TitleMediaType" />
      <element minOccurs="0" name="ProviderInformationURL" type="anyURI" />
    </sequence>
  </complexType>
  <complexType name="PrivateDataType" abstract="true" />
  <complexType name="ServiceType">
    <sequence>
      <element maxOccurs="unbounded" name="ServiceName" type="tva:ServiceInformationNameType" />
      <element minOccurs="0" name="ServiceNumber" type="unsignedShort" />
      <element minOccurs="0" maxOccurs="unbounded" name="ServiceLogo" type="mpeg7:TitleMediaType" />
      <element minOccurs="0" maxOccurs="unbounded" name="ServiceDescription" type="tva:SynopsisType"
            />
      <element minOccurs="0" maxOccurs="unbounded" name="ServiceGenre" type="tva:GenreType" />
      <element minOccurs="0" maxOccurs="unbounded" name="ServiceType" type="tva:ControlledTermType"
            />
      <element minOccurs="0" maxOccurs="unbounded" name="ParentalGuidance"
            type="mpeg7:ParentalGuidanceType" />
      <element minOccurs="0" name="ServiceLanguage" type="language" />
      <element minOccurs="0" name="ServiceProvider" type="esg:ProviderType" />
    <element minOccurs="0" maxOccurs="unbounded" name="AcquisitionRef"
          type="esg:AcquisitionRefType" />
    <element minOccurs="0" maxOccurs="unbounded" name="RelatedMaterial"
          type="esg:RelatedMaterialType" />
    <element minOccurs="0" maxOccurs="unbounded" name="PrivateData" type="esg:PrivateDataType" />
  </sequence>
  <attribute name="serviceID" type="anyURI" use="required" />
  <attribute name="freeToAir" type="boolean" use="optional" />
  <attribute name="clearToAir" type="boolean" use="optional" />
</complexType>
<complexType name="ServiceBundleType">
  <sequence>
    <element maxOccurs="unbounded" name="ServiceBundleName" type="mpeg7:TextualType" />
    <element minOccurs="0" name="ServiceBundleProvider" type="esg:ProviderType" />
    <element minOccurs="0" name="ServiceBundleMediaTitle" type="mpeg7:TitleMediaType" />
    <element minOccurs="0" maxOccurs="unbounded" name="ServiceBundleDescription"
          type="mpeg7:TextualType" />
    <element minOccurs="0" maxOccurs="unbounded" name="ServiceBundleGenre" type="tva:GenreType" />
    <element minOccurs="0" maxOccurs="unbounded" name="ServiceRef" type="esg:ServiceRefType" />
    <element minOccurs="0" maxOccurs="unbounded" name="ParentalGuidance"
          type="mpeg7:ParentalGuidanceType" />
    <element minOccurs="0" maxOccurs="unbounded" name="RelatedMaterial"
          type="esg:RelatedMaterialType" />
  </sequence>
  <attribute name="serviceBundleID" type="anyURI" use="required" />
</complexType>
<complexType name="ContentType">
  <sequence>
    <element minOccurs="0" maxOccurs="unbounded" name="Title" type="mpeg7:TitleType" />
    <element minOccurs="0" maxOccurs="unbounded" name="MediaTitle" type="mpeg7:TitleMediaType" />
    <element minOccurs="0" maxOccurs="unbounded" name="ServiceRef" type="esg:ESGIDRefType" />
    <element minOccurs="0" maxOccurs="unbounded" name="Synopsis" type="tva:SynopsisType" />
    <element minOccurs="0" maxOccurs="unbounded" name="Keyword" type="tva:KeywordType" />
    <element minOccurs="0" maxOccurs="unbounded" name="Genre" type="tva:GenreType" />
    <element minOccurs="0" maxOccurs="unbounded" name="ContentType" type="tva:ControlledTermType"
          />
    <element minOccurs="0" maxOccurs="unbounded" name="ParentalGuidance"
          type="mpeg7:ParentalGuidanceType" />
    <element minOccurs="0" maxOccurs="unbounded" name="Language" type="mpeg7:ExtendedLanguageType"
          />
    <element minOccurs="0" maxOccurs="unbounded" name="CaptionLanguage"
          type="tva:CaptionLanguageType" />
    <element minOccurs="0" maxOccurs="unbounded" name="SignLanguage" type="tva:SignLanguageType"
          />
    <element minOccurs="0" name="CreditsList" type="tva:CreditsListType" />
    <element minOccurs="0" maxOccurs="unbounded" name="RelatedMaterial"
          type="esg:RelatedMaterialType" />
    <element minOccurs="0" name="Duration" type="duration" />
    <element minOccurs="0" maxOccurs="unbounded" name="PrivateData" type="esg:PrivateDataType" />
  </sequence>
  <attribute name="contentID" type="anyURI" use="required" />
</complexType>
<complexType name="ScheduleEventType">
  <sequence>
    <element minOccurs="0" name="PublishedStartTime" type="dateTime" />
    <element minOccurs="0" name="PublishedEndTime" type="dateTime" />
    <element name="ServiceRef" type="esg:ESGIDRefType" />
    <sequence maxOccurs="unbounded">
      <element minOccurs="0" name="ContentFragmentRef" type="esg:ESGIDRefType" />
      <sequence maxOccurs="unbounded">
        <element minOccurs="0" name="AcquisitionRef" type="esg:AcquisitionRefType" />
        <element minOccurs="0" name="ContentLocation" type="anyURI" />
      </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>
<complexType name="PurchaseType">
  <sequence>
    <element minOccurs="0" name="ServiceBundleRef" type="esg:ESGIDRefType" />
    <element maxOccurs="unbounded" name="Price">
      <complexType>
        <simpleContent>
          <extension base="float">
```

```
            <attribute name="currency" type="esg:currencyCodeType" use="required" />
          </extension>
        </simpleContent>
      </complexType>
    </element>
    <element minOccurs="0" maxOccurs="unbounded" name="UsageConstraints"
          type="esg:UsageConstraintsType" />
    <element minOccurs="0" maxOccurs="unbounded" name="Description" type="mpeg7:TextualType" />
    <element minOccurs="0" maxOccurs="unbounded" name="PurchaseRequest"
          type="esg:PurchaseRequestType" />
    <element minOccurs="0" maxOccurs="unbounded" name="MediaTitle" type="mpeg7:TitleMediaType" />
  </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 minOccurs="0" name="PurchaseType" type="tva:ControlledTermType" />
    <element minOccurs="0" name="QuantityUnit" type="tva:ControlledTermType" />
    <element minOccurs="0" name="QuantityRange">
      <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 minOccurs="0" name="PurchaseData" type="esg:PurchaseDataBaseType" />
    <element minOccurs="0" name="PurchaseChannelIDRef" type="esg:ESGIDRefType" />
  </sequence>
</complexType>
<complexType name="PurchaseDataBaseType" abstract="true" />
<complexType name="PurchaseDataType">
  <complexContent mixed="false">
    <extension base="esg:PurchaseDataBaseType">
      <sequence>
        <element name="Data" type="string" />
      </sequence>
    </extension>
  </complexContent>
</complexType>
<complexType name="PurchaseChannelType">
  <sequence>
    <element maxOccurs="unbounded" name="Name" type="mpeg7:TextualType" />
    <element minOccurs="0" maxOccurs="unbounded" name="Description" type="mpeg7:TextualType" />
    <element minOccurs="0" name="PortalURL" type="anyURI" />
    <element minOccurs="0" maxOccurs="unbounded" name="ContactInfo" type="anyURI" />
    <element minOccurs="0" maxOccurs="unbounded" name="MediaTitle" type="mpeg7:TitleMediaType" />
    <element minOccurs="0" maxOccurs="unbounded" name="PrivateData" type="esg:PrivateDataType" />
  </sequence>
  <attribute name="purchaseChannelID" type="anyURI" use="required" />
</complexType>
<complexType name="AcquisitionType">
  <sequence>
    <element maxOccurs="unbounded" name="ComponentDescription" type="esg:ComponentDescriptionType"
          />
    <element minOccurs="0" name="ZappingSupport" type="esg:ZappingSupportType" />
    <element minOccurs="0" maxOccurs="unbounded" name="KeyStream" type="esg:KeyStreamBaseType" />
  </sequence>
  <attribute name="contentMimeType" type="mpeg7:mimeType" use="required" />
  <attribute name="acquisitionID" type="anyURI" use="required" />
</complexType>
<complexType name="ComponentDescriptionType">
  <sequence>
    <element minOccurs="0" maxOccurs="unbounded" name="ComponentCharacteristic"
          type="esg:ComponentCharacteristicType" />
    <element name="SessionDescription" type="esg:SessionDescriptionBaseType" />
  </sequence>
</complexType>
```

```
 <complexType name="SessionDescriptionBaseType" abstract="true" />
 <complexType name="InlinedSDPType">
   <complexContent mixed="false">
     <extension base="esg:SessionDescriptionBaseType">
       <sequence>
         <element name="SDP" type="esg:SDPType" />
         <element minOccurs="0" name="AssociatedDeliveryProcedure" type="anyURI" />
       </sequence>
     </extension>
   </complexContent>
 </complexType>
 <complexType name="SDPRefType">
   <complexContent mixed="false">
     <extension base="esg:SessionDescriptionBaseType">
       <sequence>
         <element name="SDPStream" type="esg:SDPType" />
         <element name="SDPURI" type="anyURI" />
         <element minOccurs="0" name="AssociatedDeliveryProcedure" type="anyURI" />
       </sequence>
     </extension>
   </complexContent>
 </complexType>
 <simpleType name="SDPType">
   <restriction base="string" />
 </simpleType>
 <complexType name="ZappingSupportType">
   <sequence maxOccurs="unbounded">
     <element minOccurs="0" maxOccurs="unbounded" name="Type" type="tva:ControlledTermType" />
     <choice>
       <element name="MediaLocator" type="mpeg7:MediaLocatorType" />
       <element minOccurs="0" name="ZappingSDP" type="esg:SessionDescriptionBaseType" />
     </choice>
   </sequence>
 </complexType>
 <complexType name="KeyStreamBaseType" abstract="true" />
 <complexType name="KeyStreamType">
   <complexContent mixed="false">
     <extension base="esg:KeyStreamBaseType">
       <attribute name="CBMSKMSId" type="unsignedShort" use="required" />
       <attribute name="IPDCOperatorID" type="string" use="required" />
     </extension>
   </complexContent>
 </complexType>
 <complexType name="ComponentCharacteristicType" abstract="true">
   <sequence>
     <element minOccurs="0" name="Bandwidth" type="tva:BitRateType" />
   </sequence>
   <attribute name="purpose" type="string" use="optional" />
 </complexType>
 <complexType name="VideoComponentType">
   <complexContent mixed="false">
     <extension base="esg:ComponentCharacteristicType">
       <sequence>
         <element minOccurs="0" name="CodecCharacteristic" type="esg:VideoCodecCharacteristicType"
           />
         <element minOccurs="0" name="FrameRate" type="tva:FrameRateType" />
         <element minOccurs="0" name="OpenCaptionLanguage" type="language" />
         <element minOccurs="0" name="SignLanguage" type="tva:SignLanguageType" />
       </sequence>
     </extension>
   </complexContent>
 </complexType>
 <complexType name="VideoCodecCharacteristicType">
   <sequence>
     <element minOccurs="0" name="Codec" type="tva:ControlledTermType" />
     <element minOccurs="0" name="ProfileLevelIndication" type="tva:ControlledTermType" />
   </sequence>
 </complexType>
 <complexType name="AudioComponentType">
   <complexContent mixed="false">
     <extension base="esg:ComponentCharacteristicType">
       <sequence>
         <element minOccurs="0" name="Codec" type="tva:ControlledTermType" />
         <element minOccurs="0" name="Mode" type="tva:ControlledTermType" />
         <element minOccurs="0" maxOccurs="unbounded" name="Language"
           type="mpeg7:ExtendedLanguageType" />
       </sequence>
     </extension>
```

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

History

Document history		
V1.1.1	April 2006	Publication
V1.2.1	November 2006	Publication
V1.3.1	April 2009	Publication