ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection

Doc. A/331:2017
6 December 2017

Advanced Television Systems Committee
1776 K Street, N.W.
Washington, D.C. 20006
202-872-9160

The Advanced Television Systems Committee, Inc., is an international, non-profit organization developing voluntary standards and recommended practices for digital television. ATSC member organizations represent the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops digital television implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on InterSociety Coordination (JCIC): the Electronic Industries Association (EIA), the Institute of Electrical and Electronic Engineers (IEEE), the National Association of Broadcasters (NAB), the National Cable Telecommunications Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). For more information visit www.atsc.org.

Note: The user's attention is called to the possibility that compliance with this standard may require use of an invention covered by patent rights. By publication of this standard, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. One or more patent holders have, however, filed a statement regarding the terms on which such patent holder(s) may be willing to grant a license under these rights to individuals or entities desiring to obtain such a license. Details may be obtained from the ATSC Secretary and the patent holder.

Implementers with feedback, comments, or potential bug reports relating to this document may contact ATSC at https://www.atsc.org/feedback/.

Revision History

Version	Date
Candidate Standard approved	5 January 2016
Updated CS approved	21 September 2016
Standard approved	6 December 2017
Reference to A/344 [45] editorally updated to point to the current published version	19 December 2017
Reference to A/337 [7] editorally updated to point to the current published version	2 January 2018
Reference to A/360 [10] editorally updated to point to the current published version	9 January 2018

Table of Contents

1. SCOPE 1
   1.1 Introduction and Background 1
   1.2 Organization 1
2. REFERENCES 1
   2.1 Normative References 2
   2.2 Informative References 4
3. DEFINITION OF TERMS 5
   3.1 Compliance Notation 5
   3.2 Treatment of Syntactic Elements 6
      3.2.1 Reserved Elements 6
   3.3 Acronyms and Abbreviation 6
   3.4 Terms 8
   3.5 Extensibility 11
   3.6 XML Schema and Namespace 11
4. SYSTEM OVERVIEW 12
   4.1 System Conceptual Model 12
   4.2 Features 13
5. SERVICE SIGNALING OVERVIEW 14
   5.1 Receiver Protocol Stack 14
   5.2 Entity Relationships and Addressing Architecture 15
   5.3 Service Types 15
6. LOW LEVEL SIGNALING 19
   6.1 IP Address Assignment 19
   6.2 LLS Table Format 20
   6.3 Service List Table (SLT) 22
      6.3.1 SLT Syntax Description 22
      6.3.2 SLT Semantics 24
   6.4 System Time Fragment 28
   6.5 Advanced Emergency Alert Table 30
      6.5.1 AEAT and AEA Syntax Description 31
      6.5.2 AEAT and AEA Semantics 32
   6.6 OnscreenMessageNotification Fragment 37
      6.6.1 OnscreenMessageNotification Syntax 37
      6.6.2 OnscreenMessageNotification Semantics 38
   6.7 Signaling Server 39
7. SERVICE LAYER SIGNALING 41
   7.1 ROUTE/DASH Service Layer Signaling 44
      7.1.1 Streaming Content Signaling 45
      7.1.2 App-based Feature Signaling 45
      7.1.3 User Service Description 45
      7.1.3.1 User Service Description for ROUTE – Semantics 46
      7.1.4 Service-based Transport Session Instance Description (S-TSID) 48
      7.1.5 DASH Media Presentation Description (MPD) 50

7.1.5.1 Signaling for Staggercast Audio Representation 51
7.1.5.2 Content ID for Content in ROUTE/DASH Services 51
7.1.6 Service Signaling Delivery 51
7.1.6.1 Signaling Description Encapsulation 51
7.1.6.2 Signaling Description Filtering 51
7.1.7 Associated Procedure Description (APD) 52
7.1.7.1 APD Semantics 53
7.1.7.2 End of File Transmission and Start of File Repair Procedure 53
7.1.8 Distribution Window Description (DWD) 53
7.2 MMTP-Specific Service Layer Signaling 54
7.2.1 User Service Description for MMTP 55
7.2.1.1 User Service Description for MMTP – Semantics 58
7.2.2 DASH Media Presentation Description (MPD) 61
7.2.3 MMTP-Specific Signaling Message 61
7.2.3.1 mmt_atsc3_message() MMTP-Specific Signaling Message 62
7.2.3.2 Video Stream Properties Descriptor 65
7.2.3.2.1 Syntax 65
7.2.3.2.1.1 Scalability Information 68
7.2.3.2.1.2 MultiView Information 68
7.2.3.2.1.3 Resolution, Chroma Format, Bit-depth and Video Properties Information: 69
7.2.3.2.1.4 Picture Rate Information 70
7.2.3.2.1.5 Bit Rate Information 70
7.2.3.2.1.6 Color Information 71
7.2.3.2.1.7 Transfer Function Information 72
7.2.3.3 ATSC Staggercast Descriptor 72
7.2.3.4 Audio Stream Properties Descriptor 73
7.2.3.4.1 Syntax and Semantics 73
7.2.3.4.2 Emergency Information Time Information 80
7.2.3.4.3 Multi-Stream Information 81
7.2.3.4.4 Presentation Aux-Stream Information 82
7.2.3.5 Caption Asset Descriptor 82
7.2.3.5.1 Syntax and Semantics 82
7.3 Content Advisory Ratings in Service Signaling 84
7.3.1 DASH Signaling of RRT-based Content Advisories 84
7.3.2 MMTP-Specific Signaling of RRT-Based Content Advisories 86
7.3.3 MMTP-Specific Signaling of Non-RRT Content Advisories 86
8. DELIVERY AND AL-FEC 86
8.1 Broadcast Delivery 86
8.1.1 ROUTE/DASH 86
8.1.1.1 Streaming Services Delivery 87
8.1.1.2 Locally-Cached Content Delivery 88
8.1.1.3 Synchronization and Time 88
8.1.1.4 ROUTE/DASH System Model 89
8.1.1.5 ROUTE System Buffer Model 91
8.1.1.5.1 Data Delivery Event (DDE) 91
8.1.1.5.2 Media Delivery Event (MDE) 91
8.1.1.5.3 ROUTE Transport Buffer Model 92
8.1.1.6 Application of AL-FEC (Informative) 93

8.1.2 MMTP/MPU 94
8.1.2.1 Broadcast Streaming Delivery of MPUs 94
8.1.2.1.1 Mapping Between an ATSC 3.0 Service and MMT Packages 94
8.1.2.1.2 Constraints on MPU 95
8.1.2.1.3 Constraints on MMTP 95
8.1.2.2 Packetization of MPU 97
8.1.2.3 Synchronization 98
8.1.2.4 Delivery of Locally-Cached Service Content 98
8.1.2.5 AL-FEC 99
8.2 Hybrid (Broadcast/Broadband) Delivery 99
8.2.1 ROUTE/DASH Modes of Hybrid Service Access 99
8.2.1.1 Synchronization 99
8.2.1.2 Broadband DASH-only Service Access 100
8.2.1.2.1 Exclusive Delivery of Service Components via Broadband 100
8.2.1.2.2 Hand-off from Broadcast to Broadband Service Access 100
8.2.2 MMTP/MPU Modes of Hybrid Service Access 100
8.2.2.1 Introduction 100
8.2.2.2 Constraints on DASH 101
8.2.2.3 Synchronization 101
8.2.2.4 Acquisition 102
8.2.2.5 Broadband-Only Service (Signaled by Broadcast) 102
8.3 File Repair Procedure 102
8.3.1 File Repair for Receivers that are not AL-FEC Capable 102
8.3.2 File Repair for Receivers that are AL-FEC capable 103
8.3.3 Repair Server Location and Response Deadline 103
ANNEX A : ROUTE 105
A.1 OVERVIEW of ROUTE 105
A.1.1 General 105
A.1.2 Source Protocol 105
A.1.3 Repair Protocol 105
A.1.4 Features 106
A.1.5 System Architecture 106
A.2 Data Model and Session Initiation 108
A.2.1 Data Model 108
A.2.2 ROUTE Session 108
A.2.3 Transport Sessions 109
A.3 Source Protocol Specification 109
A.3.1 Overview 109
A.3.2 Description 109
A.3.2.1 Source Flow Syntax Description 109
A.3.2.2 Source Flow Semantics 111
A.3.3 Delivery Objects 114
A.3.3.1 Overview 114
A.3.3.2 File Mode 114
A.3.3.3 Entity Mode 121
A.3.3.4 Unsigned Package Mode 121
A.3.3.5 Signed Package Mode 121
A.3.4 Usage of ALC and LCT 121

A.3.5 Packet Format 122
A.3.5.1 FEC Payload ID for Source Flows 123
A.3.5.2 FEC Payload ID for Repair Flows 123
A.3.6 LCT Building Block 123
A.3.7 Extension Headers for MDE Mode Operation 126
A.3.7.1 EXT_ROUTE_PRESENTATION_TIME Header 126
A.3.7.2 `EXT_TIME` Header 126
A.3.8 Basic ROUTE Sender Operation 126
A.3.9 Basic ROUTE Receiver Operation 127
A.3.9.1 Overview 127
A.3.9.2 Basic Delivery Object Recovery 128
A.3.9.3 General Metadata Recovery 129
A.3.9.4 Packaged Mode Reception 129
A.4 Repair Protocol Specification 129
A.4.1 Introduction 129
A.4.2 FEC Protocol 130
A.4.2.1 Introduction 130
A.4.2.2 FEC Transport Object Construction 131
A.4.2.3 Super-Object and FEC repair for delivery objects 132
A.4.2.4 Repair Packet Structure 133
A.4.2.5 Summary FEC Information 133
A.4.2.6 Extension Headers 134
A.4.3 Repair Flow Declaration 134
A.4.3.1 General 134
A.4.3.2 Semantics 134
A.4.3.3 TOI Mapping 137
A.4.3.4 Examples for `RepairFlow` Declarations 137
A.4.4 Receiver Operation 139
A.4.4.1 Introduction 139
A.4.5 Example Operation 140
A.4.6 Repair Protocol 141
A.5 Security Considerations 141
ANNEX B : SIGNALING INSTANCE EXAMPLES 142
B.1 Hierarchy Signaling Structure 142
B.2 Fast Service Scan 143
B.3 Full Service Scan 144
B.4 Service Acquisition in the Pure Broadcast (One ROUTE/MMTP Session) 147
B.5 Service Acquisition in the Pure Broadcast (Multiple ROUTE/MMTP Sessions) 148
B.6 ESG Bootstrapping via Broadband 150
B.7 Hybrid Delivery (Multiple Audio Language) 151
B.8 Handoff (Broadcast to Broadband, and Back) 153
B.9 Scalable Coding (Capability in the SLT) 154
B.10 SLT Delivery Path 156
B.11 Multiple SLTs in a Broadcast Stream 157
B.12 Portions of a Service delivered in Multiple RF Channels without Channel Bonding 158
B.13 Duplicates of a Service delivered in Multiple RF Channels without Channel Bonding 160

B.14 Portions of a Service delivered in Multiple RF Channels with Channel Bonding – case#1: essential Portion is delivered in non-bonded PLP(s) 162
B.15 Portions of a Service delivered in Multiple RF Channels with Channel Bonding – case #2: essential Portion is delivered in bonded PLP(s) 164
B.16 Duplicates of a Service delivered in Multiple RF Channels with Channel Bonding 167
ANNEX C : FILTERING FOR SIGNALING FRAGMENTS 170
ANNEX D : TEMPLATE-BASED COMPRESSION 171
D.1 Diff and Patch Operation 171
D.2 Diff Representation 172
D.3 Template pre-sharing 172
ANNEX E : ACQUISITION AND PLAYBACK OF SERVICE USING MMTP 173
E.1 Introduction 173
E.2 Mapping Between the Physical Layer and MMTP Sessions 173
E.3 Service Acquisition 174
ANNEX F : RATING REGION TABLE REQUIREMENTS 177
F.1 Introduction 177
F.2 RRT Requirements 177
F.2.1 Rating Region Tables Semantics 178
ANNEX G : EMERGENCY MESSAGE SIGNALING 180
G.1 Emergency Alert System Structure 180
G.1.1 Overview of the System 180
G.1.2 Flow of Emergency Alert Signaling and Rich Media Contents 184
G.1.2.1 Emergency Alert Message from External Sources 185
G.2 Meaning of Wake-Up bits 186
G.3 Emergency Alert System Operation 187
G.4 Advanced Emergency Alerting (AEA) 188
G.4.1 AEA Delivery 188
G.4.2 Rich Media Content 188
ANNEX H : MEDIA TYPE REGISTRATIONS 189
H.1 S-TSID 189
H.2 USD for ROUTE 190
H.3 USD for MMTP 191
H.4 APD 193

Index of Figures and Tables

Figure 4.1 Conceptual protocol stack. 13
Figure 5.1 ATSC 3.0 receiver protocol stack. 14
Figure 5.2 Service List Table references to Services. 15
Figure 5.3 UML Diagram showing relationships among Service Management, Delivery, and Physical Layer entities. 15
Figure 7.1 Example use of service signaling for bootstrapping and service discovery. 43
Figure 7.2 Service Layer Signaling data model for ROUTE/DASH Linear Services. 44
Figure 7.3 Service Layer Signaling data model for MMTP/MPU Linear Services. 54
Figure 8.1 ROUTE/DASH system model. 89
Figure 8.2 Concept of an MDE data block starting with a RAP. 91
Figure 8.3 Illustration of an MDE data block at the video level. 92
Figure 8.4 ROUTE transport buffer model. 92
Figure 8.5 An MMT Package delivered over a single MMTP packet flow. 95
Figure 8.6 An MMT Package delivered over two MMTP packet flows. 95
Figure 8.7 Receiver buffer model consuming MMTP sessions. 97
Figure 8.8 Example of media aware packetization of MPU. 98
Figure 8.9 ROUTE/DASH hybrid synchronization. 99
Figure 8.10 Conceptual hybrid service architecture. 101
Figure A.1.1 Reference receiver architecture model in ROUTE. 107
Figure A.1.2 Sender operation of ROUTE protocol. 108
Figure A.3.1 ROUTE distribution in file mode compared to FLUTE distribution. 115
Figure A.3.2 Overall ROUTE packet format. 123
Figure A.3.3 FEC Payload ID for Source Flows 123
Figure A.3.4 FEC Payload ID for Repair Flows 123
Figure A.3.5 12-byte `EXT_ROUTE_PRESENTATION_TIME` header. 126
Figure A.3.6 Receiver operation. 128
Figure A.4.1 FEC packet generation. 131
Figure A.4.2 FEC transport object construction (example with S = 3). 132
Figure A.4.3 `EXT_TOL` Header (24-bit version shown on top and 48-bit version shown on bottom). 134
Figure A.4.4 `RepairFlow` example #1. 137
Figure A.4.5 `RepairFlow` example #2. 138
Figure A.4.6 `RepairFlow` example #3. 138
Figure A.4.7 `RepairFlow` example #4. 139
Figure A.4.8 `RepairFlow` example #5. 139
Figure A.4.9 ROUTE receiver with FEC. 140
Figure B.1.1 ATSC 3.0 ROUTE hierarchical signaling architecture. 142

Figure B.1.2 ATSC 3.0 MMTP hierarchical signaling architecture. 143
Figure B.2.1 Fast service scan signaling flow. 143
Figure B.3.1 ROUTE full-service scan signaling flow. 144
Figure B.3.2 MMTP full-service scan signaling flow. 146
Figure B.4.1 Service acquisition in the pure broadcast (one ROUTE session). 147
Figure B.4.2 Service acquisition in the pure broadcast (one MMTP session). 148
Figure B.5.1 Service acquisition in the pure broadcast (multiple ROUTE sessions). 149
Figure B.5.2 Service acquisition in the pure broadcast (multiple MMTP sessions). 150
Figure B.5.3 Service acquisition in the pure broadcast (one MMTP session and one ROUTE session). 150
Figure B.6.1 ROUTE ESG bootstrapping via broadband. 151
Figure B.7.1 ROUTE hybrid (multiple audio languages). 152
Figure B.7.2 MMTP hybrid (multiple audio languages). 153
Figure B.8.1 Handoff (broadcast to broadband, and back). 154
Figure B.9.1 Scalable coding for a linear TV Service delivered by ROUTE (capability in SLT). 155
Figure B.9.2 Scalable coding for a linear TV Service delivered by MMTP (capability in SLT). 156
Figure B.10.1 ROUTE SLT delivery path. 157
Figure B.11.1 Multiple SLTs. 158
Figure B.12.1 Portions of a Service delivered in multiple RF channels without channel bonding (first RF Channel). 159
Figure B.12.2 Portions of a Service delivered in multiple RF channels without channel bonding (second RF Channel). 160
Figure B.13.1 Duplicates of a Service delivered in multiple RF channels without channel bonding (first channel). 161
Figure B.13.2 Duplicates of a Service delivered in multiple RF channels without channel bonding (second channel). 162
Figure B.14.1 Portions of a Service delivered in multiple RF channels with channel bonding – Case #1: essential portion is delivered in non-bonded PLP(s) (first channel). 163
Figure B.14.2 Portions of a Service delivered in multiple RF channels with channel bonding – Case #1: essential portion is delivered in non-bonded PLP(s) (second channel). 164
Figure B.15.1 Portions of a Service delivered in multiple RF channels with channel bonding – Case #2: essential portion is delivered in bonded PLP(s) (first channel). 165
Figure B.15.2 Portions of a Service delivered in multiple RF channels with channel bonding – Case #2: essential portion is delivered in bonded PLP(s) (second channel). 166
Figure B.16.1 Duplicates of a Service delivered in multiple RF channels with channel bonding (first channel). 168
Figure B.16.2 Duplicates of a Service delivered in multiple RF channels with channel bonding (second channel). 169
Figure C.1 TOI bit assignment for SLS packages. 170

Figure D.1.1 Template based signaling fragment compression. 172
Figure E.2.1 A PHY channel consisting of two PLPs. 174
Figure E.3.1 MPU broadcast streaming channel change. 175
Figure G.1.1 A receiver behavior in two different states. 181
Figure G.1.2 High level architecture of the U.S. Emergency Alert system. 182
Figure G.1.3 Information Flows in the U.S. Emergency Alert System. 182
Figure G.1.4 Information Flows in the Canadian National Alert Aggregation and Distribution System. 183
Figure G.1.5 Emergency Alert data flows. 184
Figure G.1.6 Emergency Alert data flows. 186

Table 6.1 Common Bit Stream Syntax for LLS Tables 21
Table 6.2 SLT XML Format 23
Table 6.3 Code Values for `urlType` 24
Table 6.4 Code Values for `SLT.Service@serviceCategory` 25
Table 6.5 Code Values for SLT.Service.BroadcastSvcSignaling@slsProtocol 27
Table 6.6 Code Values for `SLT.Service.OtherBsid@type` 28
Table 6.7 `SystemTime` Element Structure 28
Table 6.8 Daylight Saving Signaling Throughout the Year 30
Table 6.9 `AEAT` Element Structure 31
Table 6.10 Code Values for `AEAT.AEA@audience` 33
Table 6.11 Code Values for `AEAT.AEA@aeaType` 33
Table 6.12 Code Values for `AEAT.AEA@priority` 34
Table 6.13 Code Values for `AEAT.AEA.Media@mediaType` 37
Table 6.14 `OnscreenMessageNotification` Element Structure 38
Table 6.15 Path Terms, in Order of Appearance in Path 39
Table 6.16 Metadata Object Types 40
Table 6.17 ATSC-Defined Extension to MBMS Metadata Envelope `item` Element 41
Table 7.1 Semantics of the User Service Bundle Description Fragment for ROUTE 46
Table 7.2 Semantics of the Service-based Transport Session Instance Description Fragment 49
Table 7.3 Semantics of the Associated Procedure Description Fragment 53
Table 7.4 XML Format of the User Service Bundle Description Fragment for MMTP 56
Table 7.5 Bit Stream Syntax for mmt_atsc3_message() 63
Table 7.6 Code Values for atsc3_message_content_type 64
Table 7.7 Code Values for atsc3_message_content_compression 64
Table 7.8 Bit Stream Syntax for Video Stream Properties Descriptor 65
Table 7.9 Values of Multiple Frame Rate Temporal Filtering Parameters 68
Table 7.10 Bit Stream Syntax for Scalability Information 68

Table 7.11 Bit Stream Syntax for Multi-View Information 68
Table 7.12 Bit Stream Syntax for Resolution, Chroma Format, Bit-Depth 69
Table 7.13 Bit Stream Syntax for Picture Rate Information 70
Table 7.14 Bit Stream Syntax for Bit Rate Information 70
Table 7.15 Bit Stream Syntax for Color Information 71
Table 7.16 Bit Stream Syntax for Transfer Function Information 72
Table 7.17 Bit Stream Syntax for ATSC Staggercast Descriptor 72
Table 7.18 Bit Stream Syntax for the Audio Stream Properties Descriptor 73
Table 7.19 Syntax for AC-4 profile_level_indication() 76
Table 7.20 Syntax for MPEG-H audio_channel_config() 77
Table 7.21 Meaning of audio_rendering_indication Values 78
Table 7.22 accessibility Bits 79
Table 7.23 Meaning of role Values 80
Table 7.24 Bit Stream Syntax for Emergency Information Time Information 80
Table 7.25 Bit Stream Syntax for Multi-Stream Information 81
Table 7.26 Bit Stream Syntax for presentation_aux_stream Information 82
Table 7.27 Bit Stream Syntax for caption_asset_descriptor() 83
Table 7.28 Code Values for role 84
Table 7.29 Code Values for aspect_ratio 84
Table 7.30 Example Content Advisory Rating Strings 86
Table A.3.1 XML Format of `SrcFlow` Element 110
Table A.3.2 Meaning of the Delivery Object Format ID Values 114
Table A.3.3 ATSC-Defined Extensions to `FDT-Instance` Element 116
Table A.3.4 ATSC-Defined Extensions to `FDT-Instance.File` Element 118
Table A.3.5 Identifiers for File Templates 120
Table A.3.6 Defined Values of `Codepoint` Field of LCT Header 125
Table A.4.1 Semantics of `RepairFlow` Element 135
Table F.2.1 `RatingRegionTables` Element Structure 178
Table F.2.2 `TextType` Element Structure 179
Table G.2.1 Meaning of EA Wakeup 187

ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection

1. SCOPE

This document specifies protocols used for delivery and synchronization of media and non-timed data in the ATSC 3.0 system.

1.1 Introduction and Background

This document specifies the technical mechanisms and procedures pertaining to service signaling and IP-based delivery of a variety of ATSC 3.0 services and contents to ATSC 3.0-capable receivers over broadcast, broadband and hybrid broadcast/broadband networks. The service signaling functionality defines the data formats and information components necessary to discover and acquire user services. The IP-based delivery functionality specifies two application transport protocols for the carriage of media content and service signaling data over broadcast and/or broadband networks to receivers. The delivery functionality also includes mechanisms for the synchronization of media components delivered on the same or different transport networks, and application-layer forward error correction methods that enable error-free reception and consumption of media streams or discrete file objects.

1.2 Organization

This document is organized as follows:

- Section 1 – Scope and organization
- Section 2 – Lists references and applicable documents.
- Section 3 – Provides a definition of terms, acronyms, and abbreviations for this document.
- Section 4 – System overview
- Section 5 – Service signaling overview
- Section 6 – Specification of low-layer signaling
- Section 7 – Specification of service-layer signaling
- Section 8 – Specification of delivery, synchronization, and AL-FEC
- Annex A – Real-time Object delivery over Unidirectional Transport (ROUTE)
- Annex B – Signaling instance examples
- Annex C – Filtering for signaling fragments
- Annex D – Template-based compression
- Annex E – Acquisition and Playback of Service Using MMTP
- Annex F – Rating Region Table Requirements
- Annex G – Emergency Message Signaling
- Annex H – Media Type Registrations

2. REFERENCES

All referenced documents are subject to revision. Users of this Standard are cautioned that newer editions might or might not be compatible.

2.1 Normative References

The following documents, in whole or in part, as referenced in this document, contain specific provisions that are to be followed strictly in order to implement a provision of this Standard.

[1] ATSC: "ATSC Standard: Program and System Information Protocol for Terrestrial Broadcast and Cable," Doc. A/65:2013, Advanced Television Systems Committee, Washington, D.C., 7 August 2013.

[2] ATSC: "ATSC- Mobile DTV Standard, Part 4: Announcement", Doc. A/153 Part 4:2009, Advanced Television Systems Committee, Washington, D.C., October 2009.

[3] ATSC: "ATSC Standard: System Discovery and Signaling," Doc. A/321:2016, Advanced Television Systems Committee, Washington, D.C., 23 March 2016.

[4] ATSC: "ATSC Standard: Physical Layer Protocol," Doc. A/322:2017, Advanced Television Systems Committee, Washington, D.C., 6 June 2017.

[5] ATSC: "ATSC Standard: Service Announcement," Doc. A/332:2017, Advanced Television Systems Committee, Washington, D.C., 6 December 2017.

[6] ATSC: "ATSC Standard: Service Usage Reporting," Doc. A/333:2017, Advanced Television Systems Committee, Washington, D.C., 4 January 2017.

[7] ATSC: "ATSC Standard: Application Signaling," Doc. A/337:2018, Advanced Television Systems Committee, Washington, D.C., 2 January 2018.

[8] ATSC: "ATSC Standard: AC-4 System," Doc. A/342-2:2017, Advanced Television Systems Committee, Washington, D.C., 23 February 2017.

[9] ATSC: "ATSC Standard: MPEG-H System," Doc. A/342-3:2017, Advanced Television Systems Committee, Washington, D.C., 3 March 2017.

[10] ATSC: "ATSC Standard: ATSC 3.0 Security and Service Protection," Doc. A/360:2018, Advanced Television Systems Committee, Washington, D.C., 9 January 2018.

[11] CTA: "U.S. and Canadian Region Rating Tables (RRT) and Content Advisory Descriptors for Transport of Content Advisory Information Using ATSC Program and System Information Protocol (PSIP)," Doc. CTA-766-D (ANSI), Consumer Technology Association, Arlington, VA, December 11, 2013.

[12] DASH IF: "Guidelines for Implementation: DASH-IF Interoperability Points for ATSC 3.0, Version 1.0," DASH Interoperability Forum, January 31, 2017. http://dashif.org/wp-content/uploads/2017/02/DASH-IF-IOP-for-ATSC3-0-v1.0.pdf

[13] "Digital Audio Compression (AC-4) Standard, Part 2: Immersive and personalized audio," Doc. ETSI TS 103 190-2 V1.1.1 (2015-09), European Telecommunications Standards Institute, September 2015.

[14] "Universal Mobile Telecommunications Systems (UMTS); LTE; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (3GPP TS 26.346 version 13.3.0 Release 13)," Doc. ETSI TS 126 346 v13.3.0 (2016-01), European Telecommunications Standards Institute, 2014.

[15] IEEE: "Use of the International Systems of Units (SI): The Modern Metric System," Doc. SI 10-2010, Institute of Electrical and Electronics Engineers, New York, N.Y.

[16] IETF: RFC 1952, "GZIP file format specification version 4.3," Internet Engineering Task Force, Reston, VA, May, 1996. http://tools.ietf.org/html/rfc1952

[17] IETF: RFC 2387, "The MIME Multipart/Related Content-type", Internet Engineering Task Force, Reston, VA, August 1998. http://tools.ietf.org/html/rfc2387

[18] IETF: RFC 2557, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", Internet Engineering Task Force, Reston, VA, March 1999. http://tools.ietf.org/html/rfc2557

[19] IETF: RFC 3986, "Uniform Resource Identifier (URI): Generic Syntax," Internet Engineering Task Force, Reston, VA, January, 2005. http://tools.ietf.org/html/rfc3986

[20] IETF: RFC 4566, "SDP: Session Description Protocol," Internet Engineering Task Force, Reston, VA, July 2006. http://tools.ietf.org/html/rfc4566

[21] IETF: RFC 5052, "Forward Error Correction (FEC) Building Block," Internet Engineering Task Force, Reston, VA, August 2007. http://tools.ietf.org/html/rfc5052

[22] IETF: RFC 5751, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification," Internet Engineering Task Force, Fremont, CA, January 2010. https://tools.ietf.org/html/rfc5751

[23] IETF: RFC 5905, "Network Time Protocol Version 4: Protocol and Algorithms Specification," Internet Engineering Task Force, Reston, VA, June, 2010. http://tools.ietf.org/html/rfc5905

[24] IETF: RFC 5261, "An Extensible Markup Language (XML) Patch Operations Framework Utilizing XML Path Language (XPath) Selectors," Internet Engineering Task Force, Reston, VA, Sep, 2008. http://tools.ietf.org/html/rfc5261

[25] IETF: RFC 5445, "Basic Forward Error Correction (FEC) Schemes" Internet Engineering Task Force, Reston, VA, March, 2009. http://tools.ietf.org/html/rfc5445

[26] IETF: RFC 5651, "Layered Coding Transport (LCT) Building Block," Internet Engineering Task Force, Reston, VA, October, 2009. http://tools.ietf.org/html/rfc5651

[27] IETF: RFC 5775, "Asynchronous Layered Coding (ALC) Protocol Instantiation," Internet Engineering Task Force, Reston, VA, April, 2010. http://tools.ietf.org/html/rfc5775

[28] IETF: RFC 6330, "RaptorQ Forward Error Correction Scheme for Object Delivery," Internet Engineering Task Force, Reston, VA, August, 2011. http://tools.ietf.org/html/rfc6330

[29] IETF: RFC 6381, "The 'Codecs' and 'Profiles' Parameters for 'Bucket' Media Types," Internet Engineering Task Force, Reston, VA, August 2011. https://tools.ietf.org/html/rfc6381

[30] IETF: RFC 6726, "FLUTE - File Delivery over Unidirectional Transport," Internet Engineering Task Force, Reston, VA, November, 2012. http://tools.ietf.org/html/rfc6726

[31] IETF: RFC 7231, "Hypertext Transfer Protocol -- HTTP/1.1," Internet Engineering Task Force, Reston, VA, June 2014. http://tools.ietf.org/html/rfc7231

[32] IETF: BCP 47, "Tags for Identifying Languages," Internet Engineering Task Force, Reston, VA, September 2009. https://tools.ietf.org/html/bcp47

[33] ISO/IEC: "Information Technology – Generic coding of moving pictures and associated audio – Part 1: Systems," Doc. ISO/IEC 13818-1:2015, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[34] ISO/IEC: "Information technology – Coding of audio-visual objects – Part 12: ISO base media file format," Doc. ISO/IEC 14496-12:2015, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[35] ISO/IEC: "Information technology – Coding of audio-visual objects – Part 15: Carriage of network abstraction layer (NAL) unit structured video in ISO base media file format," Doc.

ISO/IEC 14496-15:2014 with Cor. 1:2015, International Organization for Standardization/ International Electrotechnical Commission, Geneva Switzerland.

[36] ISO/IEC: "Information technology – MPEG systems technologies – Part 8: Coding-independent code points," Doc. ISO/IEC 23001-8:2013, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[37] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 1: MPEG media transport (MMT)," Doc. ISO/IEC 23008-1:2017(E), International Organization for Standardization/ International Electrotechnical Commission, Geneva Switzerland.

[38] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 2: High Efficiency Video Coding," Doc. ISO/IEC 23008-2:2015, International Organization for Standardization/ International Electrotechnical Commission, Geneva Switzerland.

[39] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 3: 3D audio," Doc. 23008-3:2015, with Amendment 2:2016 and Amendment 3:2017. International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[40] ITU: "Parameter values for the HDTV standards for production and international programme exchange," ITU-R Recommendation BT.709-5 (2002), International Telecommunications Union, Geneva.

[41] SMPTE: "Advertising Digital Identifier (Ad-ID®) Representations," RP 2092-1:2015, Society of Motion Picture and Television Engineers.

[42] SMPTE: "Digital Object Identifier (DOI) Name and Entertainment ID Registry (EIDR) Identifier Representations," RP 2079:2013, Society of Motion Picture and Television Engineers.

[43] W3C: "XML Schema Part 2: Datatypes Second Edition" W3C Recommendation, Worldwide Web Consortium, 28 October 2004.
https://www.w3.org/TR/xmlschema-2/

2.2 Informative References

The following documents contain information that may be helpful in applying this Standard.

[44] ATSC: "ATSC-Mobile DTV Standard, Part 3 – Service Multiplex and Transport Subsystem," Doc. A/153:2013, Advanced Television Systems Committee, Washington, D.C.

[45] ATSC: "ATSC Standard: ATSC 3.0 Interactive Content," Doc. A/344:2017, Advanced Television Systems Committee, Washington, D.C., 18 December 2017.

[46] ETSI: "3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service (MBMS) User service guidelines (Release 13)," Doc. TR 126 946 V13.0.0 (2016-03), European Telecommunications Standards Institute, March 2016.

[47] IEEE: "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," Doc. IEEE 1588-2008, Institute for Electrical and Electronics Engineers.

[48] IERS: Bulletin C, "Contains announcements of the leap seconds in UTC," International Earth Rotation and Reference Systems Service (IERS).
https://www.iers.org/IERS/EN/Publications/Bulletins/bulletins.html

[49] IETF: RFC 2365, "Administratively Scoped IP Multicast," Internet Engineering Task Force, Reston, VA, July 1998. http://tools.ietf.org/html/rfc2365

[50] IETF: RFC 6363, "Forward Error Correction (FEC) Framework," Internet Engineering Task Force, Reston, VA, October, 2011. http://tools.ietf.org/html/rfc6363

[51] IETF: RFC 6838 (BCP 13), "Media Type Specifications and Registration Procedures," Internet Engineering Task Force, Reston, VA, January 2013. https://tools.ietf.org/html/rfc6838

[52] IETF: RFC 6968, "FCAST: Object Delivery for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols," Internet Engineering Task Force, Reston, VA, July, 2013. http://tools.ietf.org/html/rfc6968

[53] IETF: RFC 7303, "XML Media Types," Internet Engineering Task Force, Reston, VA, July 2014. https://tools.ietf.org/html/rfc7303

[54] ISO/IEC: "Information Technology – High efficiency coding and media delivery in heterogeneous environments – Part 13: MMT implementation guidelines," Doc. TR 23008-13:2015(E), International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[55] ISO/IEC: "Information technology – Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," Doc. ISO/IEC 23009-1:2014, 2nd Edition, International Organization for Standardization/International Electrotechnical Commission, Geneva Switzerland.

[56] ITU-R: "The ESR5 Criterion for the Assessment of DVB-T Transmission Quality," Doc. 6E/64-E, International Telecommunication Union, April 2004

[57] NIST: "Federal Information Processing Standard Geographic Codes," 47 C.F.R. 11.31(f), National Institute of Standards and Technology, Gaithersburg, MD, 22 October 2015. http://www2.census.gov/geo/docs/reference/codes/files/national_county.txt

[58] OASIS: "Common Alerting Protocol" Version 1.2, Organization for the Advancement of Structured Information Standards, 1 July 2010. http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf

[59] SGC: "Standard Geographic Classification Code," Version 2006, updated May 2010, Statistics Canada. http://geodepot.statcan.gc.ca/2006/040120011618150421032019/031904_05-eng.jsp#archived

[60] ATSC: "ATSC Standard: Captions and Subtitles," Doc. A/343:2016, Advanced Television Systems Committee, Washington, D.C., 21 December 2016.

3. DEFINITION OF TERMS

With respect to definition of terms, abbreviations, and units, the practice of the Institute of Electrical and Electronics Engineers (IEEE) as outlined in the Institute's published standards [15] shall be used. Where an abbreviation is not covered by IEEE practice or industry practice differs from IEEE practice, the abbreviation in question is described in Section 3.3 of this document.

3.1 Compliance Notation

This section defines compliance terms for use by this document:

shall – This word indicates specific provisions that are to be followed strictly (no deviation is permitted).

shall not – This phrase indicates specific provisions that are absolutely prohibited.

should – This word indicates that a certain course of action is preferred but not necessarily required.

should not – This phrase means a certain possibility or course of action is undesirable but not prohibited.

3.2 Treatment of Syntactic Elements

This document contains symbolic references to syntactic elements used in the audio, video, and transport coding subsystems. These references are typographically distinguished by the use of a different font (e.g., restricted), may contain the underscore character (e.g., sequence_end_code) and may consist of character strings that are not English words (e.g., dynrng).

3.2.1 Reserved Elements

One or more reserved bits, symbols, fields, or ranges of values (i.e., elements) may be present in this document. These are used primarily to enable adding new values to a syntactical structure without altering its syntax or causing a problem with backwards compatibility, but they also can be used for other reasons.

The ATSC default value for reserved bits is '1.' There is no default value for other reserved elements. Use of reserved elements except as defined in ATSC Standards or by an industry standards-setting body is not permitted. See individual element semantics for mandatory settings and any additional use constraints. As currently-reserved elements may be assigned values and meanings in future versions of this Standard, receiving devices built to this version are expected to ignore all values appearing in currently-reserved elements to avoid possible future failure to function as intended.

3.3 Acronyms and Abbreviation

The following acronyms and abbreviations are used within this document.

3GPP – 3rd Generation Partnership Program

ALC – Asynchronous Layered Coding

APD – Associated Procedure Description

AL-FEC – Application Layer Forward Error Correction

ALP – ATSC 3.0 Link layer Protocol

ATSC – Advanced Television Systems Committee

BMFF – Base Media File Format

BSID – Broadcast Stream ID

bslbf – bit string, left bit first

CENC – Common ENCryption

CP – Codepoint

CTA – Consumer Technology Association

CVS – Coded Video Sequence

CRI – Clock Relation Information

CAP – Common Alerting Protocol

DASH – Dynamic Adaptive Streaming over HTTP

DASH-IF – DASH Industry Forum

DDE – Data Delivery Event

DMD – Dynamic MetaData
DWD – Distribution Window Description
EA – Emergency Alert
EAA – Emergency Alert Application
EAS – Emergency Alert System
EB_n – Elementary Stream Buffer (n^{th} instance)
EFDT – Extended File Delivery Table
eMBMS – enhanced Multimedia Broadcast/Multicast Service
EME – Encrypted Media Extensions
ESG – Electronic Service Guide
FDD – File Delivery Description
FDT – File Delivery Table
FEC – Forward Error Correction
FLUTE – File Delivery over Unidirectional Transport
HELD – HTML Entry pages Location Description
HRBM – Hypothetical Receiver Buffer Model
HTML – Hyper Text Markup Language
HTML5 – Hyper Text Markup Language, rev 5
HTTP – Hypertext Transfer Protocol
HTTPS – Secure Hyper Text Transfer Protocol
IANA – Internet Assigned Numbers Authority
IDR – Instantaneous Decode Refresh
IETF – Internet Engineering Task Force
IP – Internet Protocol
ISO BMFF – ISO Base Media File Format
LCT – Layered Coding Transport
MA3 – MMT ATSC 3.0 Signaling Message
MBMS – Multimedia Broadcast/Multicast Service
LAN – Local Area Network
LLS – Low Level Signaling
MDE – Media Delivery Event
MIME – Multipurpose Internet Mail Extensions
MMT – MPEG Media Transport
MMTP – MPEG Media Transport Protocol
MPD – Media Presentation Description
MPI – Media Presentation Information
MPEG – Moving Pictures Experts Group
MPT – MMT Package Table
MPU – Media Processing Unit
MSB – Most Significant Bit
MSE – Media Source Extensions
NRT – Non-Real Time

OTI – Object Transmission Information
PAT – MPEG-2 Program Association Table
PLP – Physical Layer Pipe
RAP – Random Access Point
RFC – Request for Comments
ROUTE – Real-Time Object Delivery over Unidirectional Transport
RRT – Rating Region Table
SCT – Sender Current Time
SLS – Service Layer Signaling
SLT – Service List Table
S-TSID – Service-based Transport Session Instance Description
TAI – International Atomic Time
TB_n – Transport Buffer (n^{th} instance)
T-MDE – Transport Media Delivery Event
TOI – Transport Object Identifier
TOL – Transport Object Length
T-RAP – Transport Random Access Point
TSI – Transport Session Identifier
UDP – User Datagram Protocol
uimsbf – unsigned integer, most significant bit first
URI – Uniform Resource Identifier
UML – Unified Modeling Language
URL – Uniform Resource Locator
USBD/USD – User Service Bundle Description / User Service Description
UTC – Universal Coordinated Time
W3C – Worldwide Web Consortium
WAN – Wide Area Network
XLink – XML Linking Language
XML – eXtensible Markup Language
XSL – eXtensible Stylesheet Language

3.4 Terms

The following terms are used within this document.

App-Based Feature – Service component consisting of an application, optional files to be used by the application, and optional notifications directing the application to take particular actions at particular times.

App-Based Service – Service consisting entirely of app-based features, which provide the user interface for the service.

Asset – Any multimedia data entity that is associated with a unique identifier and that is used for building a multimedia presentation.

Bootstrap Signaling (Information) – Synonymous with Low Level Signaling (LLS) information implemented by the Service List Table (SLT) in support of rapid RF channel scanning and

service acquisition by the receiver through discovery of Service Layer Signaling (SLS) information.

Broadcast Stream – The abstraction for an RF Channel which is defined in terms of a carrier frequency centered within a specified bandwidth.

Byte-Range-based File Repair – The method of file repair whereby in the event of incomplete file reception over broadcast delivery, the receiver, as the file repair client, calculates the byte range(s) corresponding to the missing source symbols, and uses HTTP partial GET (or nominal HTTP GET) to retrieve that data from the repair server. The repair server, as a conventional HTTP server, is AL-FEC unaware, and stores the file object as the resource to be returned in part or in whole according to the client request.

Coded Video Sequence – Sequence parameter sets which apply to a series of consecutive coded video pictures.

Data Delivery Event (DDE) – A Data Delivery Event (DDE) is the result of a block based MAC/PHY delivering relevant contents of a specific physical layer block to a specific ROUTE session at specific time.

DASH Client – A DASH Client per the DASH-IF [12] profile.

DASH Media Presentation – A DASH Media Presentation per the DASH-IF [12] profile.

DASH Media Presentation Description (MPD) – A DASH MPD per the DASH-IF [12] profile.

DASH Media Segment – A DASH Media Segment per the DASH-IF [12] profile

DASH Period – A DASH Period per the DASH-IF [12] profile.

DASH Representation – A DASH Representation per the DASH-IF [12] profile.

DASH Segment – Refers to a DASH Initialization Segment or Media Segment per the DASH-IF [12] profile).

Default Video – For a Service that includes multiple video assets, indicates the video preferred by the service provider for selection in the absence of an action, instruction or configuration to make an alternate selection.

EA Service – Service that delivers rich media resources that are referenced in an emergency alert CAP message.

ESG Service – Service that delivers Electronic Service Guide (ESG) information.

Extended FDT Instance – An instance of the FLUTE [30] FDT that includes extensions. The Extended FDT may be transported as the `EFDT` element within the `Srcflow` element (see Section A.3.2.2), or as an FLUTE `FDT-Instance` element in TOI=0 in accordance with [30].

(HTTP) File Repair – HTTP transactions between the receiver and a network repair server, conducted over the broadband channel, which enables the receiver to recover partially delivered object(s).

Filter Code – An integer which represents a personalization description. Filter Codes are created by broadcasters according to broadcasters' particular personalization categories, such as truck owner, sustaining member, or a zip code. Filter Codes are unique within the scope of an AppContextID described in A/344 [45]. Filter Codes may be associated with files (see `File@filterCodes` in Section A.3.3.2.3.1).

Linear Audio/Video Service – Service consisting of one or more continuous video components, one or more continuous audio components, each associated with one or more of the video components, and one or more closed caption components, each associated with one or more of the audio components, all streamed in real time. May also contain app-based features.

Linear Audio-only Service – Service consisting of one or more continuous audio components and one or more closed caption components, each associated with one or more of the audio components, all streamed in real time. May also contain app-based features.

LLS (Low Level Signaling) – Signaling information which supports rapid channel scans and bootstrapping of service acquisition by the receiver.

Media Delivery Event (MDE) – A Media Delivery Event (MDE) is the arrival of a collection of bytes that is meaningful to the upper layers of the stack for example the media player and decoder(s). MDE data blocks have delivery deadlines. The grouping of bytes that is a RAP is a "Delivery" in ROUTE and the arrival of these bytes is an "Event" at an upper layer. See Section 8.1.1.5.2 for further details.

MPI message – MMT signaling message containing an MPI Table.

MP Table – MMT Package Table containing information on MMT assets/content components.

MPI Table – MMT table containing presentation information.

Media Processing Unit – Generic container for independently decodable timed or non-timed data that is media codec agnostic.

MMT Package – Logical collection of media data, delivered using the MMT protocol.

MMT Protocol – Application layer transport protocol for delivering MMTP payload over IP networks.

MMTP Packet – Formatted unit of the media data to be delivered using the MMT protocol.

MPT message – MMT signaling message containing an MP Table.

PLP (Physical Layer Pipe) – A portion of the RF channel which has certain modulation and coding parameters.

Random Access Point (RAP) – A Random Access Point is the starting byte of a sequence of data that allows an applicable media client and decoder to start.

Repair Symbol – A symbol containing information generated by the AL-FEC code which can be used to recover lost source symbols of the transport object.

reserved – Set aside for future use by a Standard.

Service – A collection of media components presented to the user in aggregate; components can be of multiple media types; a Service can be either continuous or intermittent; a Service can be Real Time or Non-Real Time; Real Time Service can consist of a sequence of TV programs.

Source Symbol – A symbol containing information from the transport object.

Staggercast – A robustness feature that can be optionally added to audio components consisting of delivery of a redundant version of a main audio component, possibly coded with lower quality (lower bitrate, number of channels, etc.), and with a significant delay ahead of the audio with which it is associated. Receivers that support the Staggercast feature can switch to the Staggercast stream should main audio become unavailable. The delivery delay between Staggercast audio and main audio is chosen to be high enough to provide robustness thanks to sufficient time diversity between the two.

SLS (Service Layer Signaling) – Signaling which provides information for discovery and acquisition of ATSC 3.0 services and their content components.

SLT (Service List Table) – Table of signaling information which is used to build a basic service listing and provide bootstrap discovery of SLS.

S-TSID (Service-based Transport Session Instance Description) – An SLS XML fragment which provides the overall session description information for transport session(s) which carry the content components of an ATSC 3.0 service.

Symbol – A unit of data processed by an FEC code; e.g., N bytes of data. A symbol is always considered as a unit—i.e., it is either completely received or completely lost.

Transport Media Delivery Event (T-MDE) – A Transport Media Delivery Event is a Media Delivery Event wrapped in IP/UDP/ROUTE.

Transport Random Access Point (T-RAP) – A Transport Random Access Point (T-RAP) is the first byte of a Random Access Point as expressed in IP/UDP/ROUTE transport.

USBD/USD (User Service Bundle Description / User Service Description) – The XML-based SLS fragment which provides entry point information for the description and discovery of the technical details of an ATSC 3.0 service.

3.5 Extensibility

The protocols specified in the present standard are designed with features and mechanisms to support extensibility. In general, the mechanisms include:

- Use of "protocol version" fields
- Definition of fields and values reserved for future use
- Use of XML, which is inherently extensible by means of future addition of new attributes and elements, potentially associated with different namespaces

Receiving devices are expected to disregard reserved values, and unrecognized or unsupported descriptors, XML attributes and elements.

3.6 XML Schema and Namespace

A number of new XML elements are defined and used in this Standard. These elements provide various service signaling elements and attributes defined in this standard (see for example Sections 6, Low Level Signaling, Section 7 Service Layer Signaling and Annex A ROUTE protocol). These new XML elements are defined with separate namespaces in schema documents that accompany this standard. The namespaces used by various schemas are described in individual sections of the present document. The sub-string part of namespaces between the right-most two '/' delimiters indicate major and minor version of the schemas. The schemas defined in this present document shall have version '1.0', which indicates major version is 1 and minor version is 0.

The namespace designator, "`xs:`", and many terms in the "Data Type" column of tables is a shorthand for datatypes defined in W3C XML Schema [43] and shall be as defined there.

In order to provide flexibility for future changes in the schema, decoders of XML documents with the namespaces defined in the present document should ignore any elements or attributes they do not recognize, instead of treating them as errors.

All element groups and attribute groups are explicitly extensible with elements and attributes respectively. Elements can only be extended from namespaces other than the target namespace. Attributes can be extended from both the target namespace and other namespaces. If the XML schema does not permit this for some element, that is an error in the schema.

XML schemas shall use `processContents="strict"` in order to reduce inadvertant typos in instance documents. Further, users are encouraged to modify the IETF FDT schema found in RFC 6726 [30] to change `processContents` to `"strict"`. Similarly for the MBMS [14] 3GPP schemas.

XML instance documents shall use UTF-8 encoding.

In the event of any discrepancy between the XML schema definitions implied by the tables that appear in this document and those that appear in the XML schema definition files, those in the XML schema definition files are authoritative and take precedence.

The XML schema document for the schemas defined in this document can be found at the ATSC website.

4. SYSTEM OVERVIEW

An overview of the signaling, delivery, synchronization, and Application-Layer FEC (AL-FEC) protocols specified in the present document is provided below, starting with a conceptual model of the system.

4.1 System Conceptual Model

A conceptual model of the system may be found in Figure 4.1.

Two methods of broadcast service delivery are specified in this Standard. The method depicted on the left side of Figure 4.1 is based on MPEG Media Transport (MMT), ISO/IEC 23008-1 [37] and uses MMT protocol (MMTP) to deliver Media Processing Units (MPU). The method shown in the center is based on the DASH-IF [12] profile, which is based on MPEG DASH [55]. It uses Real-time Object delivery over Unidirectional Transport (ROUTE) protocol to deliver DASH Segments. The ROUTE protocol is specified in Annex A. Content not intended for rendering in real time as it is received, for example, a) a downloaded application, b) a file comprising continuous or discrete media and belonging to an app-based enhancement, or c) a file containing ESG or EA information, is also delivered by ROUTE. Signaling may be delivered over MMTP and/or ROUTE, while Bootstrap Signaling information is provided by the means of the Service List Table (SLT).

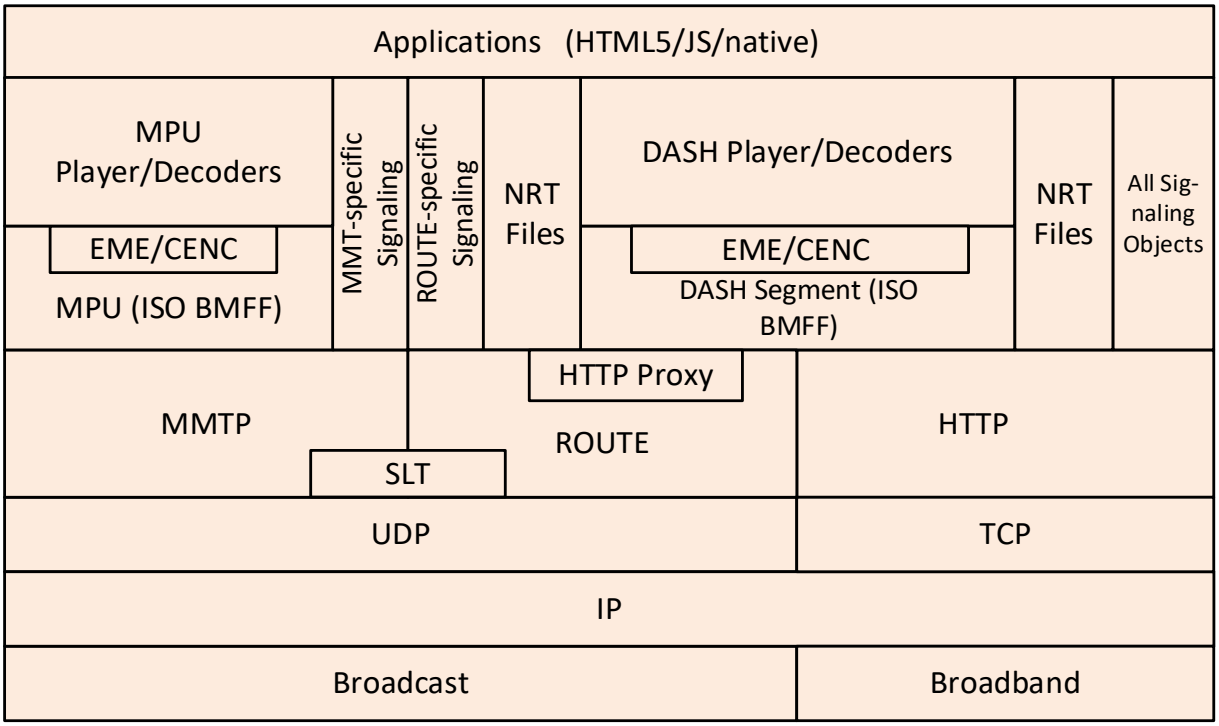


Figure 4.1 Conceptual protocol stack.

To support hybrid service delivery, in which one or more program elements are delivered via the broadband path, the DASH-IF [12] profile over HTTP/TCP/IP is used on the broadband side. Media files in the DASH-IF [12] profile based on the ISO Base Media File Format (ISO BMFF) [34] are used as the delivery, media encapsulation and synchronization format for both broadcast and broadband delivery.

4.2 Features

The protocols specified herein provide support for system features including:

- Real-time streaming of broadcast media.
- Efficient and robust delivery of file-based objects.
- Support for fast Service acquisition by receivers (fast channel change).
- Support for hybrid (broadcast/broadband) Services.
- Highly efficient Forward Error Correction (FEC)
- Compatibility within the broadcast infrastructure. with formats and delivery methods developed for (and in common use within) the Internet.
- Support for DRM, content encryption, and security.
- Support for Service definitions in which all components of the Service are delivered via the broadband path (note that acquisition of such Services still requires access to the signaling delivered in the broadcast).
- Signaling to support state-of-the-art audio and video codecs.
- Non-real-time delivery of media content.

- Non-multiplexed delivery of Service components (e.g., video and audio in separate streams).
- Support for adaptive streaming on broadband-delivered streaming content.
- Appropriate linkage to application-layer features such as ESG and Interactive Content.

5. SERVICE SIGNALING OVERVIEW

5.1 Receiver Protocol Stack

ATSC 3.0 services are delivered using three functional layers. These are the Physical layer, the Delivery layer and the Service Management layer. The Physical layer provides the mechanism by which signaling, service announcement and IP packet streams are transported over the Broadcast Physical layer and/or Broadband Physical layer. The Delivery layer provides object and object flow transport functionality. It is enabled by the MPEG Media Transport Protocol (MMTP) as defined in [37] or the Real-Time Object Delivery over Unidirectional Transport (ROUTE) protocol as defined in the present document, operating on a UDP/IP multicast over the Broadcast Physical layer, and enabled by the HTTP protocol [31] on a TCP/IP unicast over the Broadband Physical layer. The Service Management layer primarily supports the means for service discovery and acquisition to enable different types of services, such as linear TV and/or HTML5 application service, to be carried by the underlying Delivery and Physical layers. Figure 5.1 shows the ATSC 3.0 receiver protocol stack.

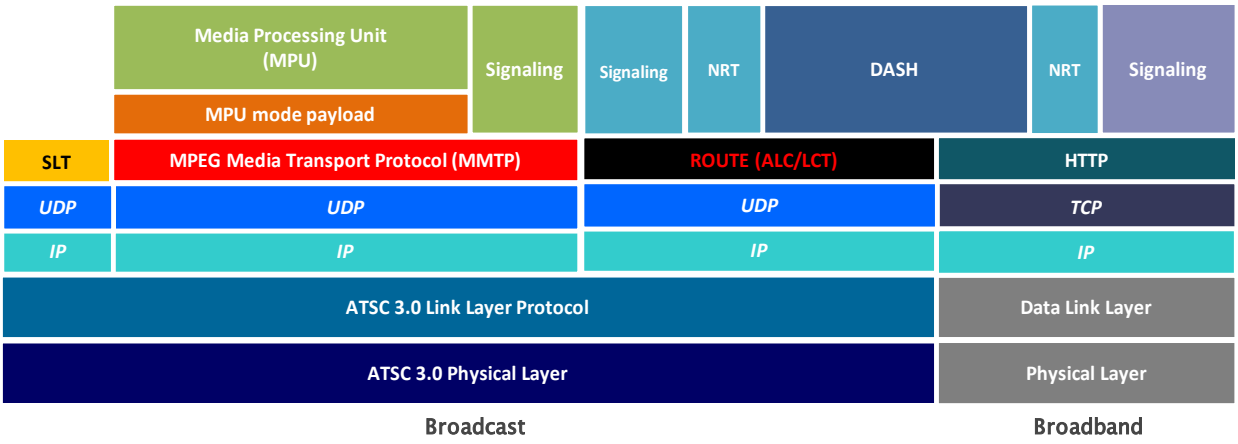


Figure 5.1 ATSC 3.0 receiver protocol stack.

Service Signaling provides service discovery and description information, and comprises two functional components: Bootstrap Signaling via the Service List Table (SLT) and Service Layer Signaling (SLS). These represent the information that is necessary to discover and acquire ATSC 3.0 services. The SLT, as described in Section 6.3, enables the receiver to build a basic service list, and bootstrap the discovery of the SLS for each ATSC 3.0 service.

The SLT can enable very rapid acquisition of basic service information. The SLS, as described in Section 7, enables the receiver to discover and access ATSC 3.0 services and their content components. The relationship between SLT and SLS for ROUTE Signaling (for ROUTE/DASH services) and the relationship between SLT and SLS for MMT Signaling (for services using MMTP/MPU streaming) is shown in Figure 5.2 below.

For ROUTE/DASH services delivered over broadcast, the SLS is carried by ROUTE/UDP/IP in one of the LCT transport channels comprising a ROUTE session, at a suitable carousel rate to support fast channel join and switching. For MMTP/MPU streaming delivered over broadcast, the SLS is carried by MMTP Signaling Messages, at a suitable carousel rate to support fast channel join and switching. In broadband delivery, the SLS is carried over HTTP(S)/TCP/IP.

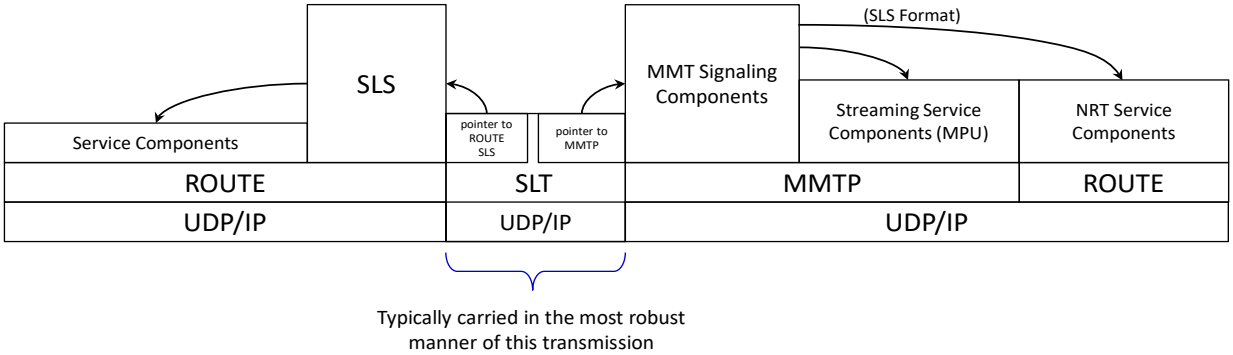


Figure 5.2 Service List Table references to Services.

5.2 Entity Relationships and Addressing Architecture

Figure 5.3 shows the ATSC 3.0 Service Management, Delivery and Physical Layer logical entities and their relationships.

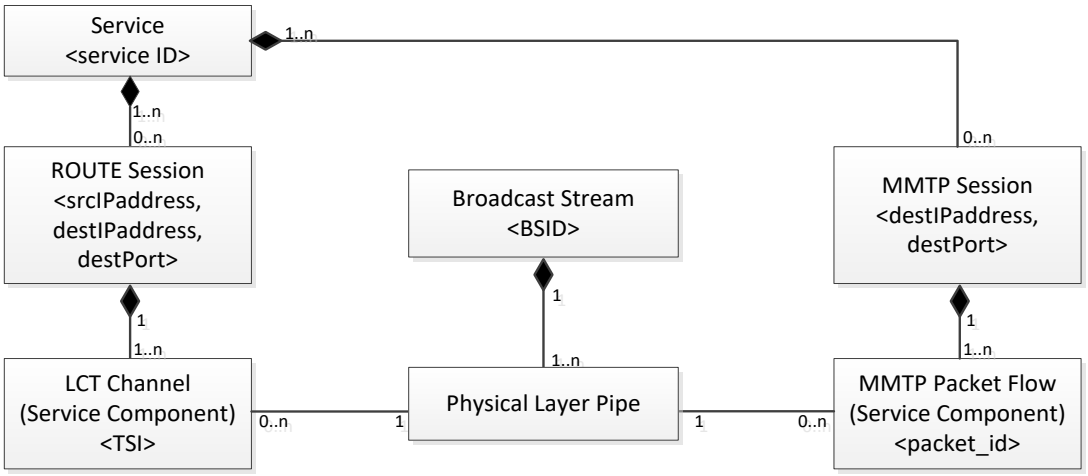


Figure 5.3 UML Diagram showing relationships among Service Management, Delivery, and Physical Layer entities.

5.3 Service Types

Five basic types of ATSC 3.0 services are currently defined:

1) Linear Audio/Video Service
2) Linear Audio-Only Service
3) App-Based Service
4) ESG Service
5) EA Service

These service types correspond to the values of `SLT.Service@serviceCategory`. New types of ATSC 3.0 services may be defined in future versions of this standard

The rules regarding presence of ROUTE sessions and/or MMTP sessions for carrying the content components of an ATSC 3.0 service shall be as follows:

a) For a broadcast delivery of a Linear service without app-based enhancement, the service's content components are carried by either (but not both):
   - One or more ROUTE sessions, or
   - One or more MMTP sessions.

b) For broadcast delivery of a Linear service with app-based enhancement, the service's content components are carried by:
   - One or more ROUTE sessions, and
   - Zero or more MMTP sessions.

   Use of both MMTP and ROUTE for streaming media components in the same service shall be disallowed.

c) For broadcast delivery of an App-based service, an ESG service, or an EA service, the service's content components are carried by:
   - One or more ROUTE sessions.

Each ROUTE session comprises one or more LCT channels which carry as a whole, or in part, the content components that make up the ATSC 3.0 service. In streaming services delivery, an LCT channel may carry an individual component of a user service such as an audio, video or closed caption stream. Streaming media is formatted as DASH Segments.

Each MMTP session comprises one or more MMTP packet flows which carry MMT signaling messages or as a whole, or in part, the content component. An MMTP packet flow may carry MMT signaling messages or components formatted per MMT [37] as MPUs.

For the delivery of App-Based features or system metadata such as service and application signaling information, an LCT channel carries file-based content items. These content files may consist of applications or continuous (time-based) or discrete (non-time-based) media components of an App-Based feature, or metadata such as SLS or ESG [5] fragments. Delivery of SLS may also be achieved through the Signaling Message mode of MMTP as per Section 7.2.3.

A Broadcast Stream is the abstraction for an RF Channel, which is defined in terms of a carrier frequency centered within a specified bandwidth. It is identified by the pair [geographic area, frequency]. A Physical Layer Pipe (PLP) corresponds to a portion of the RF channel. Each PLP has certain modulation and coding parameters.

Each service is identified by two forms of service identifier: a compact form that is used in the SLT and is unique only within the broadcast area, and a globally unique form that is used in the SLS and the ESG. A ROUTE session is identified by a source IP address, destination IP address and destination port number. An LCT channel (associated with the service component(s) it carries) is identified by a Transport Session Identifier (TSI) which is unique within the scope of the parent ROUTE session, and additionally, unique within the scope of all ROUTE sessions defined for a given Service in the S-TSID.

An LCT channel shall not be used to carry both media components of a linear Service and application files associated with that Service.

Properties common to the LCT channels, and certain properties unique to individual LCT channels, are given in a ROUTE signaling structure called a Service-based Transport Session Instance Description (S-TSID), which is part of the Service Layer Signaling. Each LCT channel

is carried over a single Physical Layer Pipe. Each PLP may contain one or more LCT channels. Different LCT channels of a ROUTE session may or may not be contained in different Physical Layer Pipes. The properties described in the S-TSID include the TSI value for each LCT channel, descriptors for the delivery objects/files, and Application Layer FEC parameters.

A MMTP Session is identified by destination IP address and destination port number. An MMTP packet flow (associated with the service component(s) it carries) is identified by a packet_id which is unique within the scope of the parent MMTP session. Properties common to each MMTP packet flow, and certain properties of MMTP packet flows, are given in the SLS. Properties for each MMTP session are given by MMT signaling messages, which may be carried within the MMTP session.

An ATSC 3.0 Service may have components delivered in more than one RF channel. A set of components of such a Service is called a "portion" of the Service when the set does not consist of all components of the Service and more than one such set of components make up the Service. On the other hand, a set of components of a Service is called a "duplicate" of the Service when the set consists of all components of the Service and more than one such set of components are delivered. Each Service represented by portions shall have only one "essential" portion which is sufficient for a meaningful presentation of the Service without the use of the other portions; i.e., "non-essential" portions (although using the other portions also may provide a more appealing presentation).

A portion or a duplicate of a Service may be delivered in a single RF channel without channel bonding. When channel bonding is applied to an ALP (ATSC 3.0 Link Layer Protocol) packet stream, a portion or a duplicate of a Service may be delivered in bonded RF channels as defined in A/322 [4].

When all Service portions or duplicates are delivered without channel bonding, SLTs (Service List Tables, defined in Section 6.3 of this document), S-TSIDs (Service-based Transport Session Instance Descriptions, defined in Section 7.1.4 of this document) and MPT messages (MMT Package Table message, defined in subclause 10.3.4 of ISO/IEC 23008-1 [37]) shall follow the signaling rules as described below:

- Each ATSC 3.0 Service represented by either Service portions or duplicates shall be included in SLTs of the RF channels in which the portions or duplicates appears. Each of these multiple listings of Services as represented by its portions or duplicates shall have the same value of service ID, and the same value of major/minor channel number. This enables the multiple portions or duplicates of a Service carried in multiple RF channels to be consolidated into a single Service in the channel map of receivers when they perform channel scans. The SLT entry for an essential portion or any duplicate of such a Service shall also list the BSIDs (Broadcast Stream Identifiers) of the Broadcast Streams where the other portions or duplicates can be found.
- For ROUTE/DASH, an S-TSID shall be delivered in a PLP of the each RF channel that delivers a Service portion or duplicate. The S-TSID for each portion or duplicate of such a Service shall describe the ROUTE sessions and LCT channels for each component of that Service portion or duplicate.
- For MMTP/MPU, MPT messages shall be delivered in a PLP of each RF channel that delivers the essential portion or a duplicate of a Service.

When Service portions are delivered with channel bonding, i.e., when those Service portions are delivered in bonded PLP(s) of more than one RF channel (i.e., bonded RF channels), the associated SLTs, S-TSIDs and MPT messages shall follow the signaling rules as described below:

1) If the essential portion of a Service is delivered by non-bonded PLP(s) of an RF channel:

- SLTs associated with both non-bonded and bonded PLP(s) that deliver any portions of the Service shall list the Service. Each of these multiple listings of the Service as represented by its portion shall have the same value of service ID, and the same value of major/minor channel number. Only the SLT entry for an essential portion of such a Service shall list the BSIDs of the Broadcast Streams where the other portions can be found.
- For ROUTE/DASH, S-TSIDs shall be delivered as follows: (a) in a single, non-bonded PLP of each RF channel that delivers one or more Service portions via non-bonded PLP(s) and (b) in a single, bonded PLP of the bonded RF channels that deliver another Service portion. In (a), each S-TSID instance shall describe the ROUTE sessions and LCT channels for each component of the Service portion delivered by the corresponding non-bonded PLP(s) in the RF channel to which the PLP carrying that S-TSID belongs. In (b), each S-TSID instance shall describe the ROUTE sessions and LCT channels for each component of the Service portion delivered by the corresponding bonded PLP(s) in the bonded RF channels to which the PLP carrying that S-TSID belongs.
- For MMTP/MPU, MPT messages shall be delivered in a single, non-bonded PLP of the RF channel that delivers the essential portion of the Service.

2) If the essential portion of a Service is delivered by bonded PLP(s) of bonded RF channels:

- Only a single SLT, associated with the bonded PLP(s) that deliver the essential portion of the Service shall list the Service. That SLT instance of such Service shall list the BSIDs of the Broadcast Streams where the other service portions can be found.
- For ROUTE/DASH, only a single S-TSID for such Service shall be delivered in a bonded PLP of the bonded RF channels that deliver the essential portion of the Service. The S-TSID instance shall describe the ROUTE sessions and LCT channels for all the components of the Service.
- For MMTP/MPU, MPT messages for such Service shall be delivered in a bonded PLP of the bonded RF channels that deliver the essential portion of the Service.

When Service duplicates are delivered with channel bonding; i.e., when those Service duplicates are delivered in bonded PLP(s) of bonded RF channels, and SLTs, S-TSIDs and MPT messages shall follow the signaling rules as described below.

- SLTs associated with both non-bonded and bonded PLP(s) that deliver duplicates of the Service shall list the Service. Each of these multiple listings of the Service as represented by its duplicates shall have the same value of service ID, and the same value of major/minor channel number. The SLT entry for a duplicate of such a Service shall list the BSIDs (Broadcast Stream Identifiers) of the Broadcast Streams where the other duplicates can be found.
- For ROUTE/DASH, S-TSIDs shall be delivered as follows: (a) in a single, non-bonded PLP of each RF channel that delivers a Service duplicate and b) in a single, bonded PLP of the bonded RF channels that deliver another Service duplicate. In (a), each S-TSID instance shall describe the ROUTE sessions and LCT channels for each component of the Service duplicate delivered by the corresponding non-bonded PLP(s) in the RF channel to

which the PLP carrying that S-TSID belongs. In (b), each S-TSID instance shall describe the ROUTE sessions and LCT channels for each component of the Service duplicate delivered by the corresponding bonded PLP(s) in the bonded RF channels to which the PLP carrying that S-TSID belongs.

- For MMTP/MPU, MPT messages shall be delivered as follows: (a) in a single, non-bonded PLP of each RF channel that delivers a Service duplicate and (b) in a single, bonded PLP of the bonded RF channels that deliver another Service duplicate.

  Note: Multiple tuners are required to recover an ALP packet stream from the bonded PLPs, i.e., the same number of tuners as the number of the RF channels for the bonded PLPs are required. For single-tuner receivers to be able to acquire a Service when channel bonding is applied, it must be able to acquire the SLT describing that Service. If the S-TSID of Service can be acquired with a single tuner tuned to a particular RF channel, then all components listed in the S-TSID can be acquired with a single tuner tuned to the same RF channel.

6. LOW LEVEL SIGNALING

Signaling information which is carried in the payload of IP packets with a well-known address/port dedicated to this function is referred to as Low Level Signaling (LLS). Five types of LLS information may be carried, each in the form of a LLS Table: Service List Table (SLT), Rating Region Table (RRT), System Time fragment, and Advanced Emergency Alert Table (AEAT), and Onscreen Message Notification fragment.

6.1 IP Address Assignment

LLS shall be transported in IP packets with address `224.0.23.60` and destination port `4937/udp`.[1] All IP packets other than LLS IP packets shall carry a Destination IP address either (a) allocated and reserved by a mechanism guaranteeing that the destination addresses in use are unique in a geographic region[2],or (b) in the range of `239.255.0.0` to `239.255.255.255`[3], where the bits in the third octet shall correspond to a value of **`SLT.Service`**`@majorChannelNo` registered to the broadcaster for use in the Service Area[4] of the broadcast transmission, with the following caveats:

- If a broadcast entity operates transmissions carrying different Services on multiple RF frequencies with all or a part of their service area in common, each IP address/port combination shall be unique across all such broadcast emissions;
- In the case that multiple LLS streams (hence, multiple SLTs) are present in a given broadcast emission, each IP address/port combination in use for non-LLS streams shall be unique across all Services in the aggregate broadcast emission;

[1] IANA has assigned this multicast address to atsc-mh-ssc and this port address as AtscSvcSig, although these packets are not intended for distribution over a LAN or WAN subsequent to reception.

[2] For a destination IP address and port number pair to be "unique in a geographic region," the broadcaster must assure that no receiver in the Service Area of the broadcast can receive content (from a different broadcast emission) using that same IP address/port combination.

[3] This IP address range is the "IPv4 Local Scope," per RFC 2365 [49], Section 6.1.

[4] Assignments for major channel numbers in the range 2–69 follow the rules in A/65 [1] Annex B. Management of assignment of major channel numbers above 69 are not currently defined.

- To minimize the impacts on the complexity of redistribution of multicast IP packets in local networks, the total number of IP multicast addresses/ports in use by a given service should be minimized.

All IP packets shall have Destination IP Port values above 1024.

Some examples:

a) A broadcaster whose Services in the SLT all have `majorChannelNo` equal to 50 may use UDP multicast Destination IP addresses in the range of `239.255.50.0` – `239.255.50.255`.
b) A broadcaster whose SLT signals that some Services are associated with `majorChannelNo` 50 and some with `majorChannelNo` 89 may use 50 for the third octet of the IP address for all Services.
c) Two broadcaster entities share one ATSC 3.0 emission. One specifies an SLT defining a Service on Virtual Channel 52.1; the other specifies an SLT defining a Service on Virtual Channel 48.1. Each may use their respective `majorChannelNo` as the third octet of the IP multicast address.
d) A broadcaster operates an ATSC 3.0 emission on RF channel 58 carrying Services with Virtual Channel Numbers 58.1 and 58.2, and also an ATSC 3.0 emission on RF channel 60 carrying Services with Virtual Channel Numbers 58.3 and 58.4, may use 58 for the third octet of all IP addresses used on the two emissions. In this case, all UDP multicast Destination IP addresses must be unique across the two emissions.

6.2 LLS Table Format

UDP/IP packets delivering LLS data shall be formatted per the bit stream syntax given in Table 6.1 below. The first byte of every UDP/IP packet carrying LLS data shall be the start of an `LLS_table()`. The maximum length of any LLS table is limited by the largest IP packet that can be delivered from the PHY layer, 65,507 bytes[5].

[5] The maximum size of the IP datagram is 65,535 bytes. The maximum UDP data payload is 65,535 minus 20 bytes for the IP header minus 8 bytes for the UDP header.

Table 6.1 Common Bit Stream Syntax for LLS Tables

Syntax	No. of Bits	Format
LLS_table() {		
LLS_table_id	8	uimsbf
LLS_group_id	8	uimsbf
group_count_minus1	8	uimsbf
LLS_table_version	8	uimsbf
switch (LLS_table_id) {		
case 0x01:		
SLT	var	Sec. 6.3
break;		
case 0x02:		
RRT	var	See Annex F
break;		
case 0x03:		
SystemTime	var	Sec. 6.4
break;		
case 0x04:		
AEAT	var	Sec. 6.5
break;		
case 0x05:		
OnscreenMessageNotification	var	Sec. 6.6
break;		
default:		
reserved	var	
}		
}		

LLS_table_id – An 8-bit unsigned integer that shall identify the type of table delivered in the body. Values of LLS_table_id of 0x00 and in the range 0x06 to 0xFE shall be ATSC reserved. LLS_table_id 0xFF shall be available for user-private usage. When set to 0xFF, the fragment shall be XML encoded and shall have at least one namespace not defined by ATSC. Note: All ATSC-defined LLS table fragments can be extended with user-private information using standard XML techniques. See Section 3.6.

LLS_group_id – An 8-bit unsigned integer that shall associate this instance of LLS_table() with a group of Services. The value of LLS_group_id shall be unique within this broadcast stream.

group_count_minus1 – An 8-bit unsigned integer that shall indicate one less than the total number of different LLS table groups in the LLS packet stream carried in this PLP. A value 0 indicates that LLS_table()s carrying only one value of LLS_group_id will be present, a value of 1 indicates that two are present, etc.

LLS_table_version – An 8-bit unsigned integer that shall be incremented by 1 whenever any data in the table identified by a combination of LLS_table_id and LLS_group_id changes. When the value reaches 0xFF, the value shall wrap to 0x00 upon incrementing. Whenever there is more than one provider sharing a broadcast stream, the LLS_table() should be identified by a combination of LLS_table_id and LLS_group_id.

SLT – The XML format Service List Table (Section 6.3), compressed with gzip [16].

RRT – An instance of a Rating Region Table conforming to the **`RatingRegionTables`** structure specified in Annex F, compressed with gzip [16].

SystemTime – The XML format System Time fragment (Section 6.4), compressed with gzip [16].

AEAT – The XML format Advanced Emergency Alerting Table fragment conforming to the Advanced Emergency Alerting Message Format (AEA-MF) structure (Section 6.5) compressed with gzip [16].

OnscreenMessageNotification – The XML format Onscreen Message Notification (Section 6.6) compressed with gzip [16].

6.3 Service List Table (SLT)

The Service List Table (SLT) is one of the instance types of LLS information. The function of the SLT is similar to that of the Program Association Table (PAT) in MPEG-2 Systems [33], and the Fast Information Channel (FIC) found in ATSC A/153, Part 3 [44]. For a receiver first encountering the broadcast emission, this is the place to start. It supports a rapid channel scan which allows a receiver to build a list of all the services it can receive, with their channel name, channel number, etc., and it provides bootstrap information that allows a receiver to discover the SLS for each service. For ROUTE/DASH-delivered services, the bootstrap information includes the source IP address, the destination IP address and the destination port of the LCT channel that carries the ROUTE-specific SLS. For MMTP/MPU-delivered services, the bootstrap information includes the destination IP address and destination port of the MMTP session carrying the MMTP-specific SLS.

The SLT supports rapid channel scans and service acquisition by including the following information about each service in the broadcast stream:

- Information necessary to allow the presentation of a service list that is meaningful to viewers and that can support initial service selection via channel number or up/down selection.
- Information necessary to locate the Service Layer Signaling for each service listed.

Each SLT shall be repeated in the LLS in which it is transported at least once every 5 seconds. To reduce channel scan time in the receiver, each SLT should be repeated more frequently, preferably once every second. However, each SLT need not be repeated more frequently than once every second.

The SLT shall be represented as an XML document containing a **`SLT`** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/SLT/1.0/
```

The definition of this schema is in an XML schema file, SLT-1.0-20170922.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"slt"`.

6.3.1 SLT Syntax Description

While the indicated XML schema specifies the normative syntax of the **`SLT`** element, informative Table 6.2 below describes the structure of the **`SLT`** element in a more illustrative way. The specifications following the table give the semantics of the elements and attributes.

Table 6.2 SLT XML Format

Element or Attribute Name	Use	Data Type	Short Description
SLT			Root element of the SLT
@bsid	1	slt:listOfUnsignedShort	Identifies the one or more Broadcast Streams comprising the Services.
SLTCapabilities	0..1	sa:CapabilitiesType	Required capabilities for decoding and meaningfully presenting the content for all the services in this SLT instance.
SLTInetUrl	0..N	anyURI	Base URL to acquire ESG or service layer signalling files available via broadband for services in this SLT.
@urlType	1	unsignedByte	Type of files available with this URL
Service	1..N		Service information
@serviceId	1	unsignedShort	Integer number that identifies this Service within the scope of this Broadcast area.
@globalServiceID	0..1	anyURI	A globally unique URI that identifies the ATSC 3.0 Service. This attribute is not present for the ESG and EAS services.
@sltSvcSeqNum	1	unsignedByte	Version of SLT service info for this service.
@protected	0..1	boolean	Indicates whether one or more components needed for meaningful presentation of this service are protected (e.g. encrypted).
@majorChannelNo	0..1	unsignedShort 1..999	Major channel number of the service
@minorChannelNo	0..1	unsignedShort 1..999	Minor channel number of the service
@serviceCategory	1	unsignedByte	Service category, coded per Table 6.4
@shortServiceName	0..1	string	Short name of the Service
@hidden	0..1	boolean	Indicates whether the service is intended for testing or proprietary use, and is not to be selected by ordinary TV receivers.
@broadbandAccessRequired	0..1	boolean	Indicates whether broadband access is required for a receiver to make a meaningful presentation of the service.
@essential	0..1	boolean	Indicates if the essential portion of the Service is delivered via this Broadcast Stream.
SimulcastTSID	0..1	unsignedShort	Identifier of an ATSC 1.0 broadcast stream carrying the same programming content.
@simulcastMajorChannelNo	0..1	unsignedShort 1..999	Major channel number of the ATSC 1.0 service carrying the same programming content.
@simulcastMinorChannelNo	0..1	unsignedShort 1..999	Minor channel number of the ATSC 1.0 service carrying the same programming content.
SvcCapabilities	0..1	sa:CapabilitiesType	Required capabilities for decoding and meaningfully presenting content of this service.
BroadcastSvcSignaling	0..1		Location, protocol, address, id information for broadcast signaling
@slsProtocol	1	unsignedByte	Protocol used to deliver the service layer signalling for this service
@slsMajorProtocolVersion	0..1	unsignedByte	Major version number of protocol used to deliver Service Layer Signalling for this service.
@slsMinorProtocolVersion	0..1	unsignedByte	Minor version number of protocol used to

Element or Attribute Name	Use	Data Type	Short Description
			deliver Service Layer Signalling for this service.
`@slsDestinationIpAddress`	1	IPv4address	A string containing the dotted-IPv4 destination address of the packets carrying broadcast SLS data for this service.
`@slsDestinationUdpPort`	1	unsignedShort	Port number of the packets carrying broadcast SLS data for this service.
`@slsSourceIpAddress`	0..1	IPv4address	A string containing the dotted-IPv4 source address of the packets carrying broadcast SLS data for this service.
`SvcInetUrl`	0..N	anyURI	URL to access Internet signalling for this service
`@urlType`	1	unsignedByte	Type of files available with this URL
`OtherBsid`	0..N	slt:listOfUnsignedShort	Identifier(s) of other Broadcast Stream(s) that deliver duplicates or portions of this Service
`@type`	1	unsignedByte	Indicates whether the Broadcast Stream identified by the **`OtherBsid`** delivers a duplicate or a portion of this service.

6.3.2 SLT Semantics

The following text specifies the semantics of the elements and attributes in the SLT.

`SLT` – Root element of the SLT.

`@bsid` – This list of one or more 16-bit unsigned integers shall identify the Broadcast Stream ID(s) of the original emission signal(s). The value of each `@bsid` shall be the same as the value signaled in `L1D_bsid` in L1-Detail Signaling in the physical layer (see A/322 [4]). In the case that the Service is delivered via channel bonding at the physical layer, the list shall include the BSID value of each RF emission involved in the bonding.

`SLTCapabilities` – Required capabilities for decoding and meaningfully presenting the content for all the services in this **`SLT`** instance. The syntax and semantics of the **`SLTCapabilities`** element shall be the same as the `sa:Capabilities` element specified under the Content fragment of the ATSC 3.0 Service Announcement specification A/332 [5].

`SLTInetUrl` – Base URL to acquire ESG or service layer signaling files for all services in this **`SLT`** via broadband, if available.

`@urlType` – Type of files available with the `sltInetUrl` (ESG or service layer signaling). See Table 6.3 for values.

Table 6.3 Code Values for `urlType`

`urlType`	**Meaning**
0	ATSC Reserved
1	URL of Service Layer Signaling Server (providing access to the Service Layer Signaling, as specified in Section 7).
2	URL of ESG server (providing access to the ESG data, as specified in A/332, Section 5.5.2 [5])
3	URL of Service Usage Data Gathering Report server (for use in reporting service usage, as specified in A/333 [6])
4	URL of Dynamic Event WebSocket Server (providing access to the dynamic events via WebSocket protocol, as specified in A/337 [7]).
Other values	ATSC Reserved

`Service` – Service information.

`@serviceId` – 16-bit integer that shall uniquely identify this Service within the scope of this Broadcast area.

`@globalServiceID` – The globally unique URI value that shall represent the identity of this ATSC 3.0 Service. This attribute provides the linkage to, and its value shall be identical to that of the **`Service`**`@globalServiceID` attribute in the A/332 ESG [5] for this Service. This attribute shall be present for the Linear A/V, Linear audio only, and App-based services. Absence of this attribute implies that this ATSC 3.0 Service is the ESG or EAS service.

`@sltSvcSeqNum` – This integer number shall indicate the sequence number of the SLT service information with service ID equal to the `serviceId` attribute above. `sltSvcSeqNum` value shall start at 0 for each service and shall be incremented by 1 every time any attribute or child of this **`Service`** element is changed. If no attribute or child element values are changed compared to the previous **`Service`** element with a particular value of `serviceId` then `sltSvcSeqNum` shall not be incremented. The `sltSvcSeqNum` field shall wrap back to 0 after reaching the maximum value.

`@protected` – When set to `"true"` indicates that one or more components necessary for meaningful presentation is protected. When set to `"false"`, indicates that no components necessary for meaningful presentation of the service are protected. Default value is `"false"`.

`@majorChannelNo` – An integer number in the range 1 to 999 that shall represent the "major" channel number of the service. Assignment of major channel numbers shall follow the guidelines given in A/65 Annex B [1] in order to guarantee that the two-part channel number combinations used by a licensee of an ATSC 3.0 broadcast will be different from those used by any other such licensee with an overlapping DTV Service Area. Note that an ATSC 3.0 broadcast Service may use the same two-part channel number combination in use in an ATSC A/53 broadcast within the DTV Service Area, given equivalent programming between the two. Specification of a `@majorChannelNo` is not required for services that are not intended to be selected directly by viewers, such as an ESG data delivery service or an EAS rich media delivery service.

`@minorChannelNo` – An integer number in the range 1 to 999 that shall represent the "minor" channel number of the service. This number is not required for services that are not intended to be selected directly by viewers, such as an ESG data delivery service or an EAS rich media delivery service.

`@serviceCategory` – 8-bit in integer that indicates the category of this service. The value shall be coded according to Table 6.4.

Table 6.4 Code Values for **`SLT.Service`**`@serviceCategory`

`serviceCategory`	**Meaning**
0	ATSC Reserved
1	Linear A/V service
2	Linear audio only service
3	App-based service
4	ESG service (program guide)
5	EAS service (emergency alert)
Other values	ATSC Reserved

`@shortServiceName` – Short name of the Service (up to 7 characters). This name is not required for services that are not intended to be selected directly by viewers, such as an ESG data delivery service or an EAS rich media delivery service.

`@hidden` – Boolean value that when present and set to `"true"` shall indicate that the service is intended for testing or proprietary use, and is not intended to be selected by ordinary TV receivers. The default value shall be `"false"` when not present.

`@broadbandAccessRequired` – A Boolean indicating that broadband access is required for a receiver to make a meaningful presentation of the service. Default value is false.

`@essential` – When this Boolean attribute is present, it shall indicate that this Service has more than one portion delivered via more than one RF channel. When this attribute is not present, it shall indicate that all portions of this service are delivered via this RF channel. When this attribute is set to `"true"`, it shall indicate that the essential portion of this Service is delivered via this Broadcast Stream. When this attribute is set to `"false"`, it shall indicate that the non-essential portion of this Service is delivered via this Broadcast Stream. When this Boolean attribute is present with value as `"true",` at least one **`OtherBsid`** element with `@type` equal to `"2"` shall be present for this Service. When not present, there is no default value.

`SimulcastTSID` – This 16-bit number, when present, shall reference the TSID value of an ATSC 1.0 broadcast emission carrying the same programming content, on the virtual channel identified with **`SimulcastTSID`**`@simulcastMajorChannelNo` if present and **`Service`**`@majorChannelNo` if not present, and **`SimulcastTSID`**`@simulcastMinorChannelNo` if present and **`Service`**`@minorChannelNo` if not present, as is being broadcast in this ATSC 3.0 Service. The TSID shall be as specified in ISO/IEC 13818-1 (MPEG-2 Systems) [33] and as used in ATSC A/65 [1]. When not present, the programming content on this Service is not associated with any ATSC 1.0 virtual channel in the local broadcast area.

`@simulcastMajorChannelNo` – An integer number in the range 1 to 999 that shall represent the "major" channel number of an ATSC 1.0 broadcast service carrying the same programming content, if present. If not present, **`Service`**`@majorChannelNo` shall represent the "major" channel number of an ATSC 1.0 broadcast service carrying the same programming content.

`@simulcastMinorChannelNo` – An integer number in the range 1 to 999 that shall represent the "minor" channel number of an ATSC 1.0 broadcast service carrying the same programming content, if present. If not present, **`Service`**`@minorChannelNo` shall represent the "minor" channel number of an ATSC 1.0 broadcast service carrying the same programming content.

`SvcCapabilities` – Required capabilities for decoding and meaningfully presenting the content for the service with service ID equal to the `@serviceId` attribute above. The syntax and semantics of the **`SvcCapabilities`** element shall be the same as the `sa:`**`Capabilities`** element specified under the Content fragment of the ATSC 3.0 Service Announcement specification A/332 [5].

`BroadcastSvcSignaling` – This element and its attributes provides broadcast signaling related information. When the **`BroadcastSvcSignaling`** sub-element is not present, then either (a) an element `SvcInetUrl` of the **`Service`** element (i.e. **`Service.SvcInetUrl`** element) shall be present with its `urlType` attribute (i.e. **`Service.SvcInetUrl`**`@urlType`) value equal to 1 (URL to SLS server), or (b) an element **`SLTInetUrl`** shall be present as a child element of the `SLT` root element (i.e. **`SLT.SLTInetUrl`**) with its `urlType` attribute (i.e. **`SLT.SLTInetUrl`**`@urlType`) value equal to 1 (URL to Signaling Server). In the latter case, the **`SLTInetUrl`** shall support the `<service_id>` path term where `service_id` corresponds to the `@serviceId` attribute for the **`Service`** element (i.e. **`Service`**`@serviceId` attribute).

`@slsProtocol` – An attribute indicating the type of delivery protocol of Service Layer Signaling used by this service, coded according to Table 6.5.

Table 6.5 Code Values for **`SLT.Service.BroadcastSvcSignaling`**`@slsProtocol`

`slsProtocol`	**Meaning**
0	ATSC Reserved
1	ROUTE
2	MMTP
other values	ATSC Reserved

`@slsMajorProtocolVersion` – Major version number of the protocol used to deliver the Service Layer Signaling for this service. Default value is 1.

`@slsMinorProtocolVersion` – Minor version number of the protocol used to deliver the Service Layer Signaling for this service. Default value is 0.

`@slsDestinationIpAddress` – A string containing the dotted-IPv4 destination address of the packets carrying SLS data for this service. The syntax shall be as defined in RFC 3986 [19] Section 3.2.2.

`@slsDestinationUdpPort` – Port number of the packets carrying SLS data for this service.

`@slsSourceIpAddress` – A string containing the dotted-IPv4 source address of the packets carrying SLS data for this service. The syntax shall be as defined in RFC 3986 [19] section 3.2.2. This attribute shall be present when the value of `@slsProtocol` is 1.

`SvcInetUrl`– Base URL to access ESG or service layer signaling files for this service via broadband, if available.

`@urlType` – Type of files available with `SvcInetUrl`. See Table 6.3 for values.

`OtherBsid` – Each instance of this list of unsigned short integer values shall indicate an identifier of another Broadcast Stream that delivers a duplicate or a portion of this Service. The format of each instance of **`OtherBsid`** shall be identical to the format of `@bsid`. At least one **`OtherBsid`** element shall be present when the `@essential` attribute is present for the parent **`Service`** element and is set to `"true"`. No **`OtherBsid`** element shall be present when the `@essential` attribute is present for the parent **`Service`** element and is set to `"false"`. One or more **`OtherBsid`** elements with `@type` equal to `"1"` may be present when `@essential` attribute is not present for the parent **`Service`** element. There is no default value when **`OtherBsid`** element is not present.

`@type` – This unsigned byte integer value shall indicate whether the Broadcast Stream identified by the **`OtherBsid`** delivers a duplicate or a portion of this Service according to Table 6.6. When the value of `@type` is set to "2", this indicates that this **`Service`** element represents a portion of a Service which has components in the Broadcast Stream identified by the identifier **`OtherBsid`** and whose Service identifier is given by the value of the `@serviceId` attribute of the parent **`Service`** element. When more than one **`OtherBsid`** element are present under its parent a **`Service`** element, the **`OtherBsid`**`@type` attribute values of all these elements shall be equal.

Table 6.6 Code Values for **`SLT.Service.OtherBsid`**`@type`

`type`	Meaning
0	ATSC Reserved
1	Duplicate
2	Portion
3-255	ATSC Reserved

6.4 System Time Fragment

System time is delivered in the ATSC PHY layer as a 32-bit count of the number of seconds, a 10-bit fraction of a second (in units of milliseconds), and optionally 10-bit microsecond and nanosecond components, since January 1, 1970 00:00:00, International Atomic Time (TAI), which is the Precision Time Protocol (PTP) epoch as defined in IEEE 1588 [47]. Further time-related information is signaled in the XML **`SystemTime`** element delivered in LLS.

System Time shall be represented as an XML document containing a **`SystemTime`** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/SYSTIME/1.0/
```

The definition of this schema is in an XML schema file, SYSTIME-1.0-20170921.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"time"`.

An LLS packet including **`SystemTime`** shall be included no less often than once every five seconds.

While the indicated XML schema specifies the normative syntax of the **`SystemTime`** element, informative Table 6.7 below describes the structure of the **`SystemTime`** element in a more illustrative way. The specifications following the table give the normative semantics of the elements and attributes.

Table 6.7 `SystemTime` Element Structure

Element or Attribute Name	Use	Data Type	Description
`SystemTime`	1		
`@currentUtcOffset`	1	short	The current offset in whole seconds between TAI and UTC.
`@ptpPrepend`	0..1	unsignedShort	Signals the upper 16 bits of the 48-bit count of PTP seconds.
`@leap59`	0..1	boolean	Indicates a pending 59-second leap second event
`@leap61`	0..1	boolean	Indicates a pending 61-second leap second event
`@utcLocalOffset`	1	duration	Indicates the offset between the local time zone of the originating broadcast station, and UTC.
`@dsStatus`	0..1	boolean	Indicates that Daylight Saving Time is in effect
`@dsDayOfMonth`	0..1	unsignedByte (range 1..31)	Indicates the local day of the month on which the transition into or out of daylight saving time is to occur.
`@dsHour`	0..1	unsignedByte (range 0..24)	Indicates the local hour at which the transition into or out of daylight saving time is to occur (0–24).

`SystemTime` – Root element.

`@currentUtcOffset` – This xs:short integer shall contain the current number of leap seconds, expressed as TAI − UTC, such that TAI − `@currentUtcOffset` = UTC. See IERS Bulletin C [48] for announcements of leap seconds. Note that IERS Bulletin C describes leap seconds as UTC-TAI. Required.

`@ptpPrepend` – This unsigned integer shall indicate, when present, the upper 16 bits of the 48-bit count of PTP seconds. When not present, the value shall be understood to be zero.

`@leap59` – When present and set to `"true"`, shall indicate that the last minute of the current UTC day contains 59 seconds (which is the case when the `@currentUtcOffset` will decrease by one at the end of the UTC day). Default value is `"false"`.

`@leap61` – When present and set to `"true"`, shall indicate that the last minute of the current UTC day contains 61 seconds (which is the case when the `@currentUtcOffset` will increase by one at the end of the UTC day). Default value is `"false"`.

`@utcLocalOffset` – This attribute shall indicate the offset between UTC and the time zone of the originating broadcast station. As a receiver may be located in a time zone adjacent to that of the broadcast transmitter, this offset is not directly usable to establish the time zone local to a given receiver. The `@utcLocalOffset` may be used to convert the value given in `@dshour` to the time of day local to the receiver. Example: for a broadcaster whose transmitter is located in the U.S. West Coast, when Daylight Saving is not in effect, `@utcLocalOffset = "-PT8H"`. For a broadcaster in Venezuela, `@utcLocalOffset = "-PT4H30M"`.

`@dsStatus` – When set to `"true"`, shall indicate that Daylight Saving Time is in effect at the transmitter location. When set to `"false"`, shall indicate that Daylight Saving Time is not in effect at the transmitter location. Shall be included whenever Daylight Saving Time is in effect at the transmitter location. Default value when not present shall be `"false"`.

`@dsDayOfMonth` – This unsigned integer value in the range 1 to 31 shall indicate, when present, that a transition into or out of daylight saving time is to occur during the present month, and the local day of the month on which it to occur. `@dsDayOfMonth` shall be included whenever there will be a Daylight Saving Time transition during the current month. `@dsDayOfMonth` shall be omitted whenever there will not be a Daylight Saving Time transition during the current month. When `@dsDayOfMonth` is not present there is no default value.

`@dsHour` – Shall indicate the hour at which the transition into or out of daylight saving time is to occur (0–24), in the time zone of the originating broadcast station. Such transitions usually occur at 2 a.m. in the U.S. `@dsHour` shall be included whenever `@dsDayOfMonth` is present and shall not be included whenever `@dsDayOfMonth` is not present. When `@dsHour` is not present there is no default value.

Table 6.8 below illustrates the progression of states of signaling Daylight Saving Time throughout the year.

Table 6.8 Daylight Saving Signaling Throughout the Year

Conditions	`dsStatus`	`dsDayOfMonth`	`dsHour`
At the beginning of the year (January) daylight saving is off.	not present (“false”)	not present	not present
This is the status of the fields until:			
When the transition into daylight saving time is between one day less than one month away and the actual transition, `dsDayOfMonth` takes the value day_in, and the `dsHour` field takes the value hour_in. The `dsStatus` attribute is not present, indicating it is not yet Daylight Saving Time. (The transition is to occur on the day_in day of the month at hour=hour_in; for example, if the transition were on April 15 at 2 a.m., then day_in=15 and hour_in=2.)	not present ("false")	day_in	hour_in
This is the status of the fields until:			
After all time zone daylight transitions (within the span of the network) have occurred, `dsStatus` is present and set to "true", indicating that daylight saving time is on. Attributes `dsDayOfMonth` and `dsHour` are not present. (In the U.S., this transition occurs no later than 7 p.m. Pacific Time on the day day_in).	"true"	not present	not present
This is the status of the fields until:			
When the transition out of daylight saving time is between one day less than one month away and the actual transition, the `dsDayOfMonth` field takes the value day_out, and `dsHour` takes the value hour_out. The `dsStatus` is present and set to "true", indicating it is still Daylight Saving Time. (The transition is to occur on the day_out day of the month at hour=hour_out; for example, if the transition were on October 27 at 2 a.m., then day_out=27 and hour_out=2)	"true"	day_out	hour_out
This is the status of the fields until:			
After all time zones (within the span of the network) have shifted out of daylight saving time, `dsStatus` takes the value "false" (or is not present), indicating that daylight saving time is off. Attributes `dsDayOfMonth` and `dsHour` are not present. (In the U.S., this transition occurs no later than 7 p.m. Pacific Time on the day day_out).	not present ("false")	not present	not present
This finishes the cycle.			

6.5 Advanced Emergency Alert Table

The AEAT (Advanced Emergency Alerting Table) is one of the instances of LLS information. The AEAT is composed of one or more AEA (Advanced Emergency Alerting) messages. The AEA message is formatted in the Advanced Emergency Alerting-Message Format (AEA-MF) structure.

- The AEA Message Format is intended to provide a digital message format for ATSC 3.0 related emergency message transmission.
- It is a specific format for forwarding a broad range of emergency-related information, including urgent bulletins, advisories, all-hazard warnings and other urgent information over an ATSC 3.0 system.
- It is intended to be extensible, and sufficient to accommodate the content alert message formats currently in the United States, Canada, Mexico, Caribbean and other regions.
- It allows a warning message to be disseminated to receivers in a consistent manner.
- It includes facilities for multimedia content that could be forwarded from the alert originator (the public authority) or the broadcaster itself (such as ancillary content the broadcast may want to forward to accompany the emergency alert).

The **AEAT** shall be represented as an XML document containing an **AEAT** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/AEAT/1.0/
```

The definitions of this schema is in an XML schema file, AEAT-1.0-20170920.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"aeat"`.

6.5.1 AEAT and AEA Syntax Description

The AEAT contains one or more AEA messages. The AEA message is composed of four basic elements: zero or one **Header**, zero or more **AEAtext**, zero or one **LiveMedia**, and zero or more **Media** for alert multimedia resources.

While the indicated XML schema specifies the normative syntax of the AEAT element, including the AEA, informative Table 6.9 below describes the structure of the AEAT element in a more illustrative way. The specifications following the table provide the semantics of the elements and attributes.

Table 6.9 AEAT Element Structure

Element or Attribute Name	Use	Data Type	Short Description
AEAT			Root element of the **AEAT**
AEA	1..N		Advanced Emergency Alert formatted as AEA-MF.
`@aeaId`	1	string	The identifier of AEA message.
`@issuer`	1	string	The identifier of the broadcast station originating or forwarding the message.
`@audience`	1	string	The intended distribution of the AEA message.
`@aeaType`	1	string	The category of the message.
`@refAEAId`	0..1	string	The referenced identifier of AEA message. It shall appear when the `@aeaType` is “update” or “cancel” and shall not appear when the `@aeaType` is ”alert”.
`@priority`	0..1	unsignedByte	The priority of the message. It shall appear when the `@aeaType` is “alert” or “update” and may appear when the `@aeaType` is “cancel”.
`@wakeup`	0..1	boolean	Indication that this AEA is associated with a wake-up event.
`Header`	0..1		The container for the basic alert envelope.
`@effective`	0..1	dateTime	The effective time of the AEA message. It appears when the `@aeaType` is “alert” or “update.” If omitted, the default is immediate.
`@expires`	0..1	dateTime	The expiration time of the AEA message. It appears when the `@aeaType` is “alert” or “update”.
`EventCode`	0..1	string	A code identifying the event type of the AEA message.
`@type`	0..1	string	A national-assigned string designating the domain of the code (e.g. SAME in US, …)
`EventDesc`	0..N	string	The short plain text description of the emergency event (e.g. “Tornado Warning” or “Tsunami Warning.”
`@lang`	1	lang	The code denoting the language of the respective element of the `EventDesc`.
`Location`	0..N	string	The geographic code delineating the affected area of the AEA message. It appears when the `@aeaType` is “alert” or “update” and can appear when the `@aeaType` is

			"cancel".
@type	1	string	A national-assigned string designating the domain of the code (e.g. FIPS in US or "SGC" in Canada)
AEAText	0..N	string	Contains the specific text of the emergency notification. It appears when the `@aeaType` is "alert" or "update" and can appear when the `@aeaType` is "cancel".
@lang	1	lang	The code denoting the language of the respective element of the AEA message text
LiveMedia	0..1		Contains the information of emergency-related live A/V service which is delivered via broadcast stream.
@bsid	1	aeat:listOfUnsignedShort	Identifier of the Broadcast Stream contains the emergency-related live A/V service.
@serviceId	1	unsignedShort	Integer number that identifies the emergency-related A/V Service.
ServiceName	0..N	string	A user-friendly name for the service where the LiveMedia is available
@lang	1	lang	The language of the text described in the ServiceName element
Media	0..N		Contains the component parts of the multimedia resource.
@lang	0..1	lang	The code denoting the language of the respective element **Media**
@mediaDesc	0..1	string	Text describing the type and content of the media file
@mediaType	0..1	string	Text identifying the intended use of the associated media.
@url	1	anyURI	URL of the media file
@alternateUrl	0..1	anyURI	Alternate URL of the media file when it is also available via non-broadcast delivery (i.e. via the Internet)
@contentType	0..1	string	IANA media type of media content referenced by **Media**`@url`
@contentLength	0..1	unsignedLong	Size in bytes of media content referenced by **Media**`@url`
@mediaAssoc	0..1	anyURI	URI of another Media element with which this attribute is associated

6.5.2 AEAT and AEA Semantics

The following text specifies the semantics of the elements and attributes in the AEAT.

AEAT – Root element of the AEAT.

AEA – Advanced Emergency Alerting Message. This element is the parent element that has `@aeaId`, `@issuer`, `@audience`, `@aeaType`, `@refAEAId`, `@priority`, and `@wakekup` attributes plus the following child elements: **Header**, **AEAtext**, **Media**, and optionally **LiveMedia** and **Media**.

AEA`@aeaId` – This attribute shall be a string value uniquely identifying the AEA message, assigned by the station (sender). The `@aeaId` shall be restricted to the 62 alphanumeric characters (basic Latin letters and Arabic digits) using the UTF-8/Unicode character set of 0x0030 through 0x0039, 0x0041 through 0x005A, 0x0061 though 0x007A, the dash (0x002D), dot (0x002E) and underscore (0x005F) characters.. This element is used to associate updates to this alert.

AEA`@issuer` – A string that shall identify the broadcast station originating or forwarding the message. `@issuer` shall include an alphanumeric value, such as call letters, station ID, group name, or other identifying value. This string shall not exceed 32 characters.

AEA`@audience` – This string shall identify the intended audience for the message. The value shall be coded according to Table 6.10.

Table 6.10 Code Values for **AEAT.AEA**@audience

Audience	Meaning
"public"	For general dissemination to unrestricted audiences. All alerts intended for public consumption must have the value of "public." (Required for AEA-MF public dissemination.)
"restricted"	For dissemination only to an audience with a defined operational requirement. Alerts intended for non-public dissemination may include the value of "restricted."
"private"	For dissemination only to specified addresses (conditional access requirement).
other values	ATSC Reserved

AEA@aeaType – This string shall identify the category of the AEA message. The value shall be coded according to Table 6.11.

Table 6.11 Code Values for **AEAT.AEA**@aeaType

aeaType	Meaning
"alert"	Indicates that AEA message is new. (Note, alert messages such as the U.S. required monthly test, RMT, are considered alert messages, and must contain the value of "alert"). In this case, @refAEAid shall not appear.
"update"	Indicates that AEA message is not new, but contains updated information from any previous emergency alert message. In this case, @refAEAid shall appear.
"cancel"	Indicates that AEA message is cancelling any previous emergency alert message, even when the message isn't expired. In this case, @refAEAid shall appear.
other values	ATSC Reserved

AEA@refAEAId – A string that shall identify the @aeaId of a referenced AEA message. It shall appear when the @aeaType is "update" or "cancel". When @aeaType is "alert", @refAEAId shall not be present. When @refAEAId is not present, there is no default value.

AEA@priority – The AEA message shall include an 8-bit unsigned integer value that indicates the priority of the alert when @aeaType is "alert" or "update". The value shall be coded according to Table 6.12. When it is not present, there is no default value.

Table 6.12 Code Values for **AEAT.AEA**@priority

priority	**Meaning**
4	Maximum Priority • Urgent or extreme message context • A highest level of alert (e.g. the U.S. Emergency Action Notification/EAN) • A Canadian "broadcast immediate" requirement in the source alert message. • Defined by station operator a time critical alert (e.g. earthquake/EQW or tornado/TOR
3	High Priority • Defined by station operator for messages of an important or severe context. • May also be used for a "broadcast immediate" message. • Overrides any previous messages.
2	Moderate Priority • Defined by station operator for messages of a moderate but actionable priority.
1	Low Priority • Defined by station operator for messages of an informative nature, or of minor and non-actionable status (e.g. weather watches).
0	Minor Priority • Defined by station operator for periodic or occasional messages of extremely minor context (e.g. test or administrative signals). • Messages should not interrupt the user from other interactive functions.
other values	ATSC Reserved

AEA@wakeup – This optional Boolean attribute, when present and set to "true" shall indicate that the AEA is associated with non-zero ea_wake_up bits (See Annex G.2). The default value, when not present, shall be "false".

Header – This element shall contain the relevant envelope information for the AEA message, including the type of alert (**EventCode**), the time the alert is effective (@effective), the time it expires (@expires), and the location of the targeted alert area (**Location**).

Header@effective – This date/time attribute shall contain the effective time of the alert message. The date and time shall be represented in the XML dateTime data type format (e.g., "2016-06-23T22:11:16-05:00" for 23 June 2016 at 11:15 am EDT). Alphabetic time zone designators such as "Z" shall not be used. The time zone for UTC shall be represented as "-00:00". When it is not present, it shall indicate "immediately effective."

Header@expires – This date/time attribute shall contain the expiration time of the AEA message. The date and time shall be represented in the XML dateTime data type format (e.g., "2016-06-23T22:11:16-05:00" for 23 June 2016 at 11:15 am EDT). Alphabetic time zone designators such as "Z" shall not be used. The time zone for UTC shall be represented as "-00:00". This attribute shall not appear for an AEA message with @aeaType as "cancel". When @aeaType is "cancel", it shall indicate "immediately expires." This attribute shall appear when the @aeaType is "alert" or "update".

EventCode – This string element shall identify the event type of the AEA message formatted as a string (which may represent a number) denoting the value itself (e.g., in the U.S., a value of "EVI" would be used to denote an evacuation warning). Values may differ from nation to nation, and may be an alphanumeric code, or may be plain text. Only one **EventCode** shall be present per AEA message. When **EventCode** is not present, there is no default value.

EventCode@type – This attribute shall be a national-assigned string value that shall designate the domain of the **EventCode** (e.g., in the U.S., "SAME" denotes standard FCC Part 11 EAS coding). Values of @type that are acronyms should be represented in all capital letters without periods.

If `@type="SAME"`, then the **`EventCode`** shall be defined as a three letter event code as defined in FCC's Part 11 rules on EAS (at 47 CFR 11.31(e)).

When `@type` is not present, there is no default value.

`EventDesc` – A string that shall contain a short plain text description of the emergency event. This string shall not exceed 64 characters. When the **`EventCode`** element is present, the **`EventDesc`** should correspond to the event code indicted in the **`EventCode`** element (e.g. an **`EventDesc`** of "Tornado Warning" corresponds to the EAS **`EventCode`** of `"TOR"`). When an **`EventCode`** element is not present, the **`EventDesc`** should provide a brief, user-friendly indication of the type of event (e.g. "School Closing").

`EventDesc``@lang` – This attribute shall identify the language of the respective **`EventDesc`** element of the alert message. This attribute shall be represented by formal natural language identifiers and shall not exceed 35 characters in length as defined by BCP 47 [32]. There shall be no implicit default value.

`Location` – This element shall describe a message target with a geographically-based code. When this element is not present, it shall indicate that the AEA message is geographically relevant throughout the transmission area.

`Location``@type` – A string that shall identify the domain of the **`Location`** code. Note that some receivers may not be capable of determining whether they are located within the signaled location area of the event. It is suggested that such receivers process the alert as if they were located within the area of the event.

- If `@type="FIPS",` then the Location shall be defined as a group of one or more numeric strings separated by commas. Each 6-digit numeric string shall be a concatenation of a county subdivision, state and county codes as defined in FIPS [57] in the manner defined in 47 CFR 11.31 as PSSCCC. Additionally, the code "000000" shall mean all locations within the United States and its territories, and the code "999999" shall mean all locations within the coverage area of the station from which this AEAT originated.
- If `@type="SGC"`, then the Location shall be defined as a group of one or more numeric strings separated by commas. Each numeric string shall be a concatenation of a 2-digit province (PR), a 2-digit census division (CD) and a 3 digit census subdivision (CSD) as defined in SGC [59]. Additionally, the code "00" shall mean all locations within Canada, and the code "9999" shall mean all locations within the coverage area of the station from which this AEAT originated.
- If `@type="polygon"`, then the **`Location`** shall define a geospatial space area consisting of a connected sequence of three or more GPS coordinate pairs that form a closed, non-self-intersecting loop. Each coordinate pair shall be expressed in decimal degrees.
- If `@type="circle"`, then the Location shall define a circular area represented by a central point given as a coordinate pair followed by a space character and a radius value in kilometers.

Textual values of `@type` are case sensitive, and shall be represented in all capital letters, with the exceptions of `"polygon"` and `"circle"`.

`AEAText` – A string of the plain text of the emergency message. Each **`AEAText`** element shall include exactly one `@lang` attribute. For **`AEAText`** of the same alert in multiple languages, this element shall require the presence of multiple **`AEAText`** elements. When **`AEAText`** is not present, there is no default value.

AEAText@lang– This attribute shall identify the language of the respective **AEAText** element of the AEA message. This attribute shall be represented by formal natural language identifiers as defined by BCP 47 [32], and shall not exceed 35 characters. There shall be no implicit default value.

LiveMedia – Identification of a broadcast delivered A/V service that may be presented to the user as a choice to tune for emergency-related information, e.g., ongoing news coverage. A **LiveMedia** element shall be present if **AEA**@wakeup is "true". When **LiveMedia** is not present, there is no default value.

LiveMedia@bsid – This list of one or more unsigned short integer values shall indicate identifier(s) of the Broadcast Stream(s) which contain essential portion of the emergency-related live A/V service. When the value of **LiveMedia**@bsid is a list of more than one unsigned short value it shall indicate multiple Broadcast Streams with channel bonding applied.

LiveMedia@serviceId – A 16-bit integer that shall uniquely identify the emergency-related live A/V service.

ServiceName – A user-friendly name for the service where the live media is available that the receiver can present to the viewer when presenting the option to tune to the **LiveMedia**, e.g., "WXYZ Channel 5." When **ServiceName** is not present, there is no default value.

ServiceName@lang – Shall identify the language of the respective **ServiceName** element of live media stream. This attribute shall be represented by formal natural language identifiers and shall not exceed 35 characters, as defined by BCP 47 [32]. There shall be no implicit default value.

Media – This element contains the component parts of the multimedia resource, including the language (@lang), description (@mediaDesc) and location (@url) of the resource. Refers to an additional file with supplemental information related to the **AEAtext**; e.g., an image or audio file. Multiple instances may occur within an **AEA** message block. Note that AEA media is made accessible to a broadcaster application using the AppContextIdList in the EFDT (**EFDT.FDT-Instance**@appContextIdList), see Section 9.2.10 of A/344 [45].

Media@lang – This attribute shall identify the respective language for each **Media** resource, to help instruct the recipient if different language instances of the same multimedia are being sent. This attribute shall be represented by formal natural language identifiers as defined by BCP 47 [32], and shall not exceed 35 characters. This element shall be present if the @mediaDesc element is present.

Media@mediaDesc – A string that shall, in plain text, describe the content of the **Media** resource. The description should indicate the media information. For example, "Evacuation map" or "Doppler radar image," etc. The language of the **Media**@mediaDesc shall be inferred to be same as the language indicated in **Media**@lang. This information may be used by a receiver to present a viewer with a list of media items that the viewer may select for rendering. If this field is not provided, the receiver may present generic text for the item in a viewer UI (e.g., if the @contentType indicates the item is a video, the receiver may describe the item as "Video" in a UI list).

Media@mediaType – This string shall identify the intended use of the associated media. Note that media items identified with this attribute are typically associated with items that are automatically handled by the receiver's alert user interface, as opposed to media that is presented in a list to the user for selection. The value shall be coded according to Table 6.13. When **Media**@mediaType is not present, there is no default value.

Table 6.13 Code Values for **AEAT.AEA.Media**@mediaType

MediaType	Meaning
"EventDescAudio"	The audio (voice) associated with the EventDesc element.
"AEAtextAudio"	The audio (voice) associated with the AEAtext element
"EventSymbol"	A symbol associated with the EventDesc
other values	ATSC Reserved

Media@url – A required attribute that shall indicate the location of the rich media resource delivered via broadcast ROUTE. The attribute is formed as a relative URL (see Section 8.1.1.2). The relative URL shall match the @Content-Location attribute of the corresponding **File** element in the Extended FDT Instance in the LCT [26] channel delivering the file, or the Entity header of the file.

Media@alternateUrl – An optional attribute that shall indicate the location of the rich media resource when it is also available via broadband. The attribute shall be formed as an absolute URL and shall reference a file on a remote server.

Media@contentType – A string that shall represent media type assigned by IANA for the media content referenced by Media@url. Media@contentType shall obey the semantics of the Content-Type header of HTTP/1.1 protocol RFC 7231 [31]. When **Media**@contentType is not present, there is no default value.

Media@contentLength – A string that shall represent the size in bytes of media content referenced by **Media**@url. When **Media**@contentLength is not present, there is no default value.

Media@mediaAssoc – An optional attribute that shall contain a **Media**@url of another rich media resource with which this media resource is associated. Examples include a closed caption track associated with a video. Construction of **Media**@mediaAssoc shall be as described in **Media**@url above. When **Media**@mediaAssoc is not present, there is no default value.

6.6 OnscreenMessageNotification Fragment

The **OnscreenMessageNotification** indicates to a receiver the broadcaster's preference for the receiver to avoid obscuring any part of the video images (e.g., when important text/visual information, including emergency-related, information is visually embedded in the video). The OnscreenMessageNotification LLS table is composed of **KeepScreenClear** elements. The @version of each **KeepScreenClear** element identifies unique **KeepScreenClear** notifications.

The **OnscreenMessageNotification** shall be represented as an XML document containing a **OnscreenMessageNotification** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/ONSCREEN/1.0/
```

The definition of this schema is in an XML schema file, ONSCREEN-1.0-20170920.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be "osmn".

6.6.1 OnscreenMessageNotification Syntax

While the indicated XML schema specifies the normative syntax of the **OnscreenMessageNotification** element, informative Table 6.14 below describes the structure of the **OnscreenMessageNotification** element in a more illustrative way. The specifications following the table give the normative semantics of the elements and attributes.

Table 6.14 `OnscreenMessageNotification` Element Structure

Element or Attribute Name	Use	Data Type	Description
`OnscreenMessageNotification`	1		
`KeepScreenClear`	0..N		Service Information related to Onscreen Message Notification
`@bsid`	1	unsignedShort	Identifier of the broadcast stream.
`@serviceId`	0..1	unsignedShort	Identifier of the service within the scope of the broadcast stream that the notification applies.
`@serviceIdRange`	0..1	unsignedShort	Identifier of a range of serviceId that this **`KeepScreenClear`** applies.
`@notificationDuration`	0..1	duration	Indicates duration of this KeepScreenClear for the signaled service(s).
`@kscFlag`	0..1	boolean	Indicates the status of **`KeepScreenClear`**
`@version`	1	unsignedByte	The version of **`KeepScreenClear`**

6.6.2 OnscreenMessageNotification Semantics

`OnscreenMessageNotification` – root element contains broadcaster and service information for on-screen important text/visual information, including emergency-related, information that has been rendered by broadcasters on their video service(s).

`KeepScreenClear` – Conveys information for service(s) regarding keep screen clear status corresponding to on-screen important text/visual information.

`@bsid` – This 16-bit identifier shall indicate the BSID of the Broadcast Stream associated with the values of `@serviceId` indicated in the message.

`@serviceId` – This 16-bit integer shall uniquely identify the service within the scope of this broadcast stream that **`KeepScreenClear`** applies to. If `@serviceId` is not present, the **`KeepScreenClear`** shall apply to all services within the broadcast stream identified by `@bsid`.

`@serviceIdRange` – Shall specify the range of services within the scope of the broadcast stream that this notification applies to. `@serviceIdRange` shall not be present when `@serviceId` is not present. If `@serviceId` is present and `@serviceIdRange` is not present, `@serviceIdRange` shall be set to 0. The **`KeepScreenClear`** element shall apply to the services identified by **`SLT.Service`**`@serviceId` in the range from `@serviceId` to `@ServiceId`+`@serviceIdRange` inclusive in the broadcast stream identified by `@bsid`.

`@notificationDuration` – This value shall be the duration of the **`KeepScreenClear`** element for the identified services within the identified broadcast stream. For the purpose of counting, time starts at the reception of the **`OnscreenMessageNotification`**. If not present, `@notificationDuration` shall be set to `"PT1M"`, i.e., 1 minute. A `@notificationDuration` value greater than 1 hour shall be set to `"PT1H"`, i.e., 1 hour. A `@notificationDuration` value of 0 or less shall be considered invalid. The `@kscFlag` of the identified services within the identified broadcast stream shall be set to `"false"` when current time reaches or exceeds (**`OnscreenMessageNotification`** reception time + `@notificationDuration`).

`@kscFlag` – indicates the status of the **`KeepScreenClear`** for the identified service(s) within the identified broadcast stream. If not present, `@kscFlag` shall be `"true"` for identified services and shall be `"false"` for all services for the broadcast stream identified by `@bsid` which are not identified by any **`KeepScreenClear`** element inside the parent **`OnscreenMessageNotification`** element. If an **`OnscreenMessageNotification`** element does not include any **`KeepScreenClear`** element then `@kscFlag` shall be `"false"` for all services for all broadcast streams.

`@version` – An 8-bit unsigned integer that shall be incremented by 1 whenever any data in the KeepScreenClear element changes. When the value reaches 0xFF, the value shall wrap to 0x00 upon incrementing. The scope of uniqueness of the `@version` is determined by the broadcaster.

6.7 Signaling Server

When an `sltInetUrl` with `urlType` attribute value `"1"` is present in the SLT, it can be used as a base URL to make HTTP requests for signaling metadata. The desired signaling metadata objects, as defined in Table 6.16, to be returned are indicated by appending path terms to the base URL (rather than using query terms). This makes the retrieval of the signaling metadata objects more efficient from the server standpoint, since no server side application is required to retrieve the desired objects. Each request simply fetches a file. To make such a request, the GET method is used, and the path appended to the end of the base URL contains terms indicating the desired signaling object or objects, as indicated in Table 6.15 below.

Table 6.15 Path Terms, in Order of Appearance in Path

Terms	Meaning
`<service_id>`	Identifies desired service
`normal|diff|template`	Identifies desired mode of files
`current|next`	Identifies desired current/next version
`ALL|RD|USBD|STSID|MPD|MMT|MPT|PAT|MPIT|CRIT|DCIT|APD|HELD|DWD|AEI|EMSG|EVTI`	Identifies desired type of object(s)

When an `sltInetUrl` with `urlType` attribute `"1"` base URL appears (at the SLT level), the `service_id` term is used to indicate the service to which the requested signaling metadata objects apply. If the `service_id` term is not present, then the signaling metadata objects for all services in the section are requested.

The `normal/diff/template` term indicates whether the normal form of the metadata object(s), the diff form of the metadata object(s), or the template form of the metadata object(s) is requested. If the normal form is desired, the `normal` term shall be omitted.

The `current/next` term indicates whether the current version of the metadata object(s) or the next version of the metadata object(s) after the current version is requested. If the current version is desired, then the `current` term shall be omitted.

The fourth term is used to indicate which types of metadata object(s) are desired. The supported types are listed in Table 6.16 below, with their descriptions.

Table 6.16 Metadata Object Types

Name	Values
ALL	All metadata objects for requested service(s)
RD	All applicable ROUTE SLS metadata fragments for the requested service(s), which shall include the USBD and S-TSID, and may include the APD and MPD fragments
APD	APD for requested service(s)
MMT	All MMT metadata objects for requested service(s)
USBD	USBD for requested service(s)
STSID	S-TSID for requested service(s)
MPD	DASH MPD for requested service(s)
PAT	MMT Package Access Table for requested service(s)
MPT	MMT Package Table for requested service(s)
MPIT	MMT Media Presentation Information Table for requested service(s)
CRIT	MMT Clock Relation Information Table for requested service(s)
DCIT	MMT Device Capabilities Information Table for requested service(s)
HELD	HTML Entry pages Location Description for requested service(s) (A/337 [7])
DWD	Distribution Window Description for requested service(s) (A/337 [7])
AEI	MMT Application Event Information for requested service(s)
EMSG	ROUTE/DASH Application Dynamic Event for requested service(s)
EVTI	MMT Application Dynamic Event for requested service(s)

Some examples of the URL for an HTTP request for signaling metadata objects would be:

- `<sltInetUrl urlType="1">/0x2107/RD` – returns the current, normal version of all ROUTE/DASH signaling objects for service with `service_id` 0x2107
- `<sltInetUrl urlType="1">/0x2103/next/MPD` – returns the next, normal version of the MPD for service with `service_id` 0x2103
- `<sltInetUrl urlType="1">/0x2104/template/HELD` – returns the current, template version of the AST for service with `service_id` 0x2104
- `<sltInetUrl urlType="1">/0x2110/template/ALL` – returns the current, template versions of all the metadata objects as denoted in Table 6.16, for the service with service_id 0x2110

When an `svcInetUrl` with `urlType` attribute `"1"` appears (at the service layer), then the same paths can be appended to the end of it, with the same semantics, except that no service term shall appear, since it is not needed to designate the desired service.

The response body for those HTTP requests shall include an MBMS metadata envelope per Section 11.1.3 of MBMS [14] containing an `item` element for each signaling object included in the response. The "Referenced" method of use of **`metadataEnvelope`** shall be used: signaling objects shall be referenced in their item elements, and they shall be packaged together with the metadata envelope in a multi-part MIME message, in the order in which they are referenced. The `validFrom` and `validUntil` attributes of the `item` elements should be present, to indicate the interval of validity of each signaling object.

The `item` element of the MBMS metadata envelope shall be extended by the addition of an optional attribute, `@nextURL`.

Thus, when the `validUntil` time approaches for a signaling object that was acquired via broadband, the device can acquire the next scheduled update to the signaling object by making an

HTTP GET request with the URL given by the `nextURL` attribute in the `item` element that was used to represent the signaling object in the metadata envelope.

The ATSC extension to the metadata envelope shall be represented as an XML data structure conforming to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/ATSC-Meta/1.0/
```

The definition of this schema is in an XML schema file, ATSC-META-1.0-20170313.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema `xmlns` short name should be `"meta"`. The indicated XML schema specifies the normative syntax of this extension.

Informative Table 6.17 below describes this attribute.

Table 6.17 ATSC-Defined Extension to MBMS Metadata Envelope `item` Element

Element or Attribute Name	Use	Data Type	Description
`@nextURL`	0..1	anyURI	The URL of the next scheduled update to the signaling object described in the item element.

`@nextURL` – A URL which when present, shall be the URL of the next scheduled update to the signaling object described in the `item` element.

If an unscheduled update is made to a signaling object, a dynamic event will be issued announcing the update, as specified in the ATSC 3.0 Application Signaling Standard [7]. A device can then acquire the updated signaling object, using the URL in the `data` attribute of the dynamic event.

When an `sltInetUrl` with `urlType` attribute `"2"` is present, the URL given by this element can be used to retrieve ESG data via broadband for all services in the SLT. When a `svcInetUrl` with `urlType` attribute `"2"` is present, the URL given by this attribute can be used to retrieve ESG data via broadband for the service with the same `serviceId` as the service element in which the `svcInetUrl` appears. In both cases the URL is used for queries as specified in the ATSC 3.0 Service Announcement Standard A/332 [5].

7. SERVICE LAYER SIGNALING

For ROUTE/DASH, the SLS for each service describes characteristics of the service, such as a list of its components and where to acquire them, and the receiver capabilities required to make a meaningful presentation of the service, and the availability and associated rules regarding access to file repair services by receivers. In the ROUTE/DASH system, the SLS includes the User Service Bundle Description (USBD), the S-TSID, the Associated Procedure Description (APD), the DASH Media Presentation Description (MPD), the HTML Entry pages Location Description (HELD) (see A/337 [7]), and the Distribution Window Description (DWD) (see A/337 [7]). The USBD and APD are based on the identically-named (i.e. User Service Bundle Description and Associated Procedure Description) service description metadata fragments as defined in MBMS [14], with extensions that support ATSC 3.0 requirements. Table 7.1 shows the elements and attributes of the ROUTE/DASH USBD that would be used in practice for ATSC 3.0 service delivery.

For MMTP, the SLS for each service describes characteristics of the service, such as a list of its components and where to acquire them, and the receiver capabilities required to make a

meaningful presentation of the service. In the MMTP system, the SLS includes the USBD fragment, the MMT Package (MP) table, the HTML Entry pages Location Description (HELD) (see A/337 [7]), and the Distribution Window Description (DWD) (see A/337 [7]). For hybrid delivery, the MMTP-specific SLS can further include the MPD for broadband components. Table 7.4 shows the elements and attributes of the MMTP USBD that would be used in practice for ATSC 3.0 service delivery.

The Service Signaling focuses on basic attributes of the service itself, especially those attributes needed to acquire the service. Properties of the service and programming that are intended for viewers appear as Service Announcement, or ESG data. Carriage of the ESG is specified in the ATSC 3.0 Service Announcement specification A/332 [5].

Having separate Service Signaling for each service permits a receiver to acquire the appropriate SLS for a service of interest without the need to parse the entire SLS carried within a Broadcast Stream.

For optional broadband delivery of Service Signaling, the SLT can include HTTP URLs where the Service Signaling files can be obtained. (See Section 6.3 above.)

Figure 7.1 provides an example of the use of the LLS to bootstrap SLS acquisition, and subsequently, the use of the SLS to acquire service components delivered on either ROUTE sessions or MMTP sessions. The figure illustrates the following signaling sequences. ATSC 3.0 receiver starts acquiring the SLT described in Section 6.3. Each service identified by `@serviceId` delivered over ROUTE sessions provides SLS bootstrapping information: source IP address (sIP1), destination IP address (dIP1), and destination port number (dPort1). Each service identified by `@serviceId` delivered over MMTP sessions provides SLS bootstrapping information: destination IP address (dIP2), and destination port number (dPort2).

For streaming services delivery using ROUTE, the receiver can acquire SLS fragments carried over the IP/UDP/LCT channel indicated in the SLT; whereas for streaming services delivery using MMTP, the receiver can acquire SLS fragments carried over an MMTP session. For service delivery using ROUTE, these fragments include USBD/USD fragments, S-TSID fragments, APD fragments, MPD fragments, HELD fragments, and the DWD fragments. They are relevant to one service. USBD/USD fragments describe service layer properties. For service delivery using MMTP, the USBD references the MMT Signaling's MPT Message, the MP Table of which provides identification of Package ID and location information for assets belonging to the service.

The S-TSID fragment provides component acquisition information associated with one Service. It also provides mapping between DASH Representations found in the MPD and in the TSI corresponding to the component of the Service. The S-TSID can provide component acquisition information in the form of a TSI and the associated DASH Representation identifier carrying DASH Segments associated with the DASH Representation. By the TSI value, the receiver collects the audio/video components from the Service and begins buffering DASH Segments then applies the appropriate decoding processes.

The APD fragment provides information regarding the HTTP file repair procedure, which may be initiated by the receiver to a file repair server, after broadcast delivery of the delivery object of interest has ended, but the entire object was not successfully received. The use of the file repair procedure is nominally used to support broadcast delivery of NRT content, e.g., application-related files that belong to an app-based enhancement, or data services such as the ESG or EAS content.

The HELD fragment (see A/337 [7]) provides application-related metadata that enables the loading and unloading of an application including the information about application-related files, such as an application entry page, files associated with the entry page, media assets expected to be

consumed by the application, or a combination of these content types for one or more applications associated with a service as defined in A/337, Application Signaling [7].

The DWD fragment (A/337 [7]) provides the broadcast delivery schedule of NRT files associated with broadcaster applications, and additional metadata such as Filter Codes that enable selective content reception, identification of the application to which the application-related files belong, and the specific collection of files delivered during a given distribution window instance.

For USBD listing service components delivered on MMTP sessions, as illustrated by "Service #2" in Figure 7.1, the receiver also acquires an MPT message with matching MMT_package_id to complete the SLS. An MPT message provides the full list of service components comprising a service and the acquisition information for each component. Component acquisition information includes MMTP session information and the packet_id within that session.

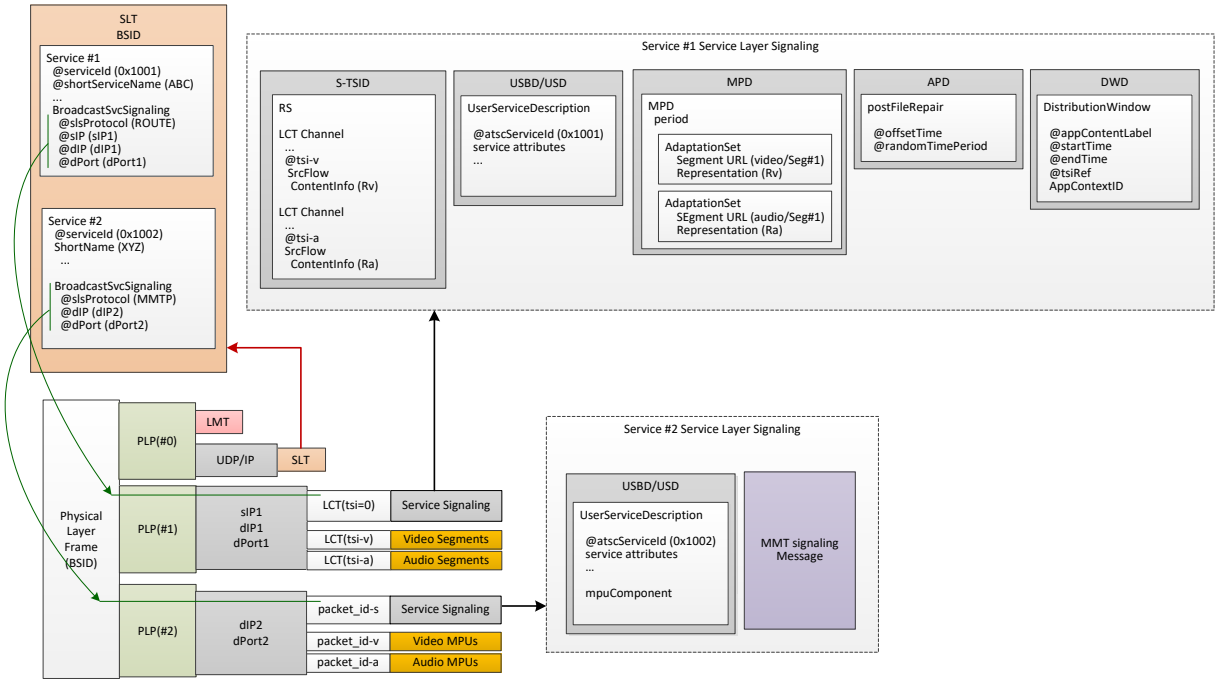


Figure 7.1 Example use of service signaling for bootstrapping and service discovery.

Annex B provides an example implementation of the ATSC 3.0 hierarchical signaling architecture featuring two separate S-TSID fragments, each of which provides the access information for the LCT channels carrying the contents of an individual ATSC 3.0 service.

SLS signaling supports the delivery of Service components in multiple PLPs. For example, a service could be constructed to transport video in one PLP and audio in a different, more robust, PLP. To accommodate expected limitations in some receiver designs in which no more than four ALP packet streams can be simultaneously monitored, and noting that receivers must continuously monitor the ALP stream transporting the LLS (in order to monitor for emergency alerts),

components of any given Service shall be transported in no more than three ALP packet streams in addition to the ALP stream transporting the LLS fragments describing that Service.[6]

7.1 ROUTE/DASH Service Layer Signaling

Service Layer Signaling provides detailed technical information to the ATSC 3.0 receiver to enable the discovery and access of ATSC 3.0 user services and their content components. It comprises a set of XML-encoded metadata fragments carried over a dedicated LCT channel. That LCT channel can be acquired using the bootstrap information contained in the SLT as described in Section 6.3. The SLS is defined on a per-service level, and it describes the characteristics and access information of the service, such as a list of its content components and how to acquire them, and the receiver capabilities required to make a meaningful presentation of the service, and the means to recover partially-received objects.. In the ROUTE/DASH system, for linear services delivery, the SLS consists of the following metadata fragments: (ROUTE-specific) USBD, S-TSID, DASH MPD, DWD (see A/337 [7]) and the HELD (see A/337 [7]). The SLS fragments shall be delivered on a dedicated LCT transport channel with TSI = 0.

The data model of the SLS fragments applicable to ROUTE/DASH linear services, shown using UML convention, is shown in Figure 7.2.

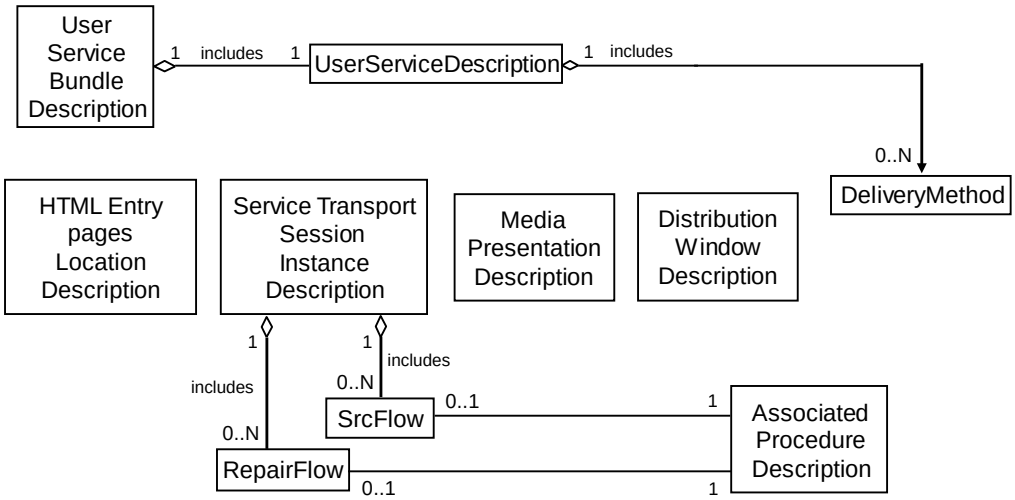


Figure 7.2 Service Layer Signaling data model for ROUTE/DASH Linear Services.

ROUTE/DASH Service Layer Signaling comprises the User Service Bundle Description (USBD), Service-based Transport Session Instance Description (S-TSID), Associated Procedure Description (APD), DASH Media Presentation Description (MPD), and HTML Entry pages Location Description (HELD), A/337 [7]), and Distribution Window Description (DWD, A/337

[6] Broadcasters should be aware that, depending upon choices made in the configuration of the PHY layer, specifically when using Layered Division Multiplexing (as specified in A/322 [4] Section 6.4), this constraint might not be sufficient to allow many receivers to operate properly because additional receiver resources may be required to access particular ALP packet stream(s). Such PHY-layer considerations may lead to the necessity of further limiting the number of ALP packet streams used to fewer than the three additional streams specified here.

[7]) metadata fragments. These service signaling fragments are applicable to linear services, and with the possible exception of the MPD fragment, also to application-based services. The USBD fragment contains service identification and metadata to enable the receiver to determine the transport mode (broadcast and/or broadband) of service components. The S-TSID fragment provides transport session descriptions for the LCT channel(s) of the one or more ROUTE sessions in which the media content components of an ATSC 3.0 service are delivered, and descriptions of the delivery objects carried in those LCT channels. In addition an instance of the APD fragment provides metadata for supporting HTTP file repair in conjunction with Extended FDT Instance parameters as defined in Annex A, Section A.3.3.1.1. The APD fragment, if present, shall be associated with either a source flow or a repair flow which carries the delivery object of interest. The HELD fragment provides application-related metadata that enables the loading and unloading of an application including the information of application-related files about one or more applications associated with a service. The HELD, USBD, S-TSID, APD and DWD are further detailed in Sections 7.1.2, 7.1.3, 7.1.4, 7.1.7 and 7.1.8 respectively.

7.1.1 Streaming Content Signaling

The streaming content signaling component of the SLS corresponds to the MPD fragment. The MPD is typically associated with linear services for the delivery of DASH Segments as streaming content. The MPD provides the resource identifiers for individual media components of the linear/streaming service in the form of Segment URLs, and the context of the identified resources within the DASH Media Presentation. It is further described in Section 7.2.2.

7.1.2 App-based Feature Signaling

App-based feature signaling component of the SLS corresponds to the HELD fragment which provides application-related metadata that enables the loading and unloading of an application including the information about application-related files of one or more applications associated with a service. It pertains to the delivery of app-based enhancement components, such as an application logic file, locally-cached media files, a network content item, or a notification stream. Note that an application can also retrieve locally-cached data over a broadband connection when available. Details about app-based feature signaling and the HELD fragment are specified in A/337, Application Signaling [7].

7.1.3 User Service Description

The top level or entry point SLS fragment is the USBD fragment. The USBD fragment for ATSC 3.0 is modeled on the USBD fragment as defined by MBMS [14], with the following extension:

- Child attributes `serviceId`, and `serviceStatus` under the element **`UserServiceDescription`**;

The following extensions defined in MBMS [14] are included:

- Child element **`BroadcastAppService`** and its child element **`BasePattern`** under the element **`DeliveryMethod`**;
- Child element **`UnicastcastAppService`** and its child element **`BasePattern`** under the element **`DeliveryMethod`**.

The **`BundleDescriptionROUTE`** shall be represented as an XML document containing a **`BundleDescriptionROUTE`** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/ROUTEUSD/1.0/
```

The definitions of this schema is in an XMLschema file, ROUTEUSD-1.0-20170920.xsd accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"routeusd"`.

While the XML schemas identified above specify the normative syntax of the elements specified in this ATSC 3.0 standard, informative Table 7.1 below describes the structure of the **`BundleDescriptionROUTE`** element in a more illustrative way. A large number of the attributes and elements in the MBMS USBD fragment are optional and not relevant to ATSC 3.0. Table 7.1 shows the elements and attributes that would be used in practice for ATSC 3.0 service delivery.

The media type corresponding to fragments containing these files shall be as specified in Annex H.2.

Table 7.1 Semantics of the User Service Bundle Description Fragment for ROUTE

Element or Attribute Name	Use	Data Type	Description
`BundleDescriptionROUTE`			Root element of the User Service Bundle Description for ROUTE/DASH.
`UserServiceDescription`	1		A single instance of an ATSC 3.0 Service.
`@serviceId`	1	unsignedShort	Reference to corresponding service entry in the (SLT).
`@serviceStatus`	0..1	boolean	Specify the status of this service as active or inactive.
`Name`	0..N	string	Name of the ATSC 3.0 service.
`@lang`	1	lang	Language of the ATSC 3.0 service name.
`ServiceLanguage`	0..N	lang	Available languages of the ATSC 3.0 service.
`DeliveryMethod`	0..N		Container of transport-related information pertaining to the contents of the service over broadcast and (optionally) broadband modes of access. This element is not applicable to and therefore shall be absent for ESG and EAS services.
`BroadcastAppService`	0..N		A DASH Representation delivered over broadcast containing the corresponding media component(s) belonging to the ATSC 3.0 Service.
`BasePattern`	1..N	string	A character pattern for use by the ATSC receiver to match against any portion of the Segment URL used by the DASH Client to request DASH Media Segments of a parent DASH Representation.
`UnicastAppService`	0..N		A DASH Representation delivered over broadband containing the constituent media content component(s) belonging to the ATSC 3.0 Service.
`BasePattern`	1..N	string	A character pattern for use by the ATSC receiver to match against any portion of the Segment URL used by the DASH Client to request DASH Media Segments of a parent DASH Representation.

7.1.3.1 User Service Description for ROUTE – Semantics

The following text specifies the semantics of the elements and attributes in the USBD fragment.

`BundleDescriptionROUTE` – Root element of the User Service Bundle Description for ROUTE.

`UserServiceDescription` – A complex element whose subordinate elements and attributes contain additional information for a single ATSC 3.0 Service.

`@serviceId` – A 16-bit integer value that shall identify the ATSC 3.0 service named in the parent USBD fragment and whose uniqueness pertains to the scope of this Broadcast area. Its value shall be identical to that of the **`SLT.Service`**`@serviceId` attribute in the SLT which references the LCT channel carrying the USBD fragment for this service.

`@serviceStatus` – A Boolean attribute which shall convey the current status of this service as being active or inactive. A value of `"true"` shall indicate that the service is active. A value of or `"false"` shall indicate that the service is inactive. The default value shall be `"true"`.

`Name` – This element shall contain the name of this ATSC 3.0 service as given by one or more languages as defined by its `lang` attribute. Absence of this attribute shall imply that the service name is unnecessary to be indicated in the USBD. For example, this service is the ESG or EAS service

`@lang` – This attribute shall indicate the language for the name of this ATSC 3.0 service, and which shall be represented by formal natural language identifiers as defined by BCP 47 [32].

`ServiceLanguage` – This element shall indicate the language for this ATSC 3.0 service, and which shall be represented by formal natural language identifiers as defined by BCP 47 [32]. Absence of this attribute shall imply that the service language is unknown.

`DeliveryMethod` – A complex element whose subordinate elements and attributes contain transport related information pertaining to the contents of this ATSC 3.0 service. This element shall contain information on the delivery mode (broadcast, broadband, or via both paths) for each of those content components. This element is not applicable to and therefore shall be absent for the ESG and EAS services.

At least one of **`BroadcastAppService`** or **`UnicastAppService`** child elements shall be present under each **`DeliveryMethod`** element.

`BroadcastAppService` – This element shall be an indication that the content item of this ATSC 3.0 service as requested by an application in the receiver is delivered over broadcast, i.e. via the ROUTE protocol. The presence of this element may be used alone, or in conjunction with its child element **`BasePattern`**, to determine whether the delivery path for that content is broadcast. Absence of this attribute shall imply that none of the content components of this service is delivered over broadcast.

`BasePattern` (under **`BroadcastAppService`**) – A string value, typically in the form of a base URI, which shall be used for matching against the resource URL provided by the application for making a request of a content component of the ATSC 3.0 service. The occurrence of a full match of a **`BroadcastAppService.BasePattern`** value to a contiguous portion of the request URL shall be an indication to the receiver that the requested content is delivered by broadcast.

`UnicastAppService` – This element shall be an indication that the content item of this ATSC 3.0 service as requested by an application in the receiver is delivered over broadband, via the HTTP protocol. The presence of this element may be used alone, or in conjunction with its child element **`BasePattern`**, to determine whether the delivery path for that content is broadband. Absence of this attribute shall imply that none of the content components of this service is delivered over broadband.

`BasePattern` (under **`UnicastAppService`**) – A string value, typically in the form of a base URI, which shall be used for matching against the resource URL provided by the application for making a request of a content component of the ATSC 3.0 service. The occurrence of a full match of a **`UnicastAppService.BasePattern`** value to a contiguous portion of the request URL shall be an indication to the receiver that the requested content is delivered by broadband.

7.1.4 Service-based Transport Session Instance Description (S-TSID)

The S-TSID is an SLS metadata fragment that contains the overall transport session description information for the zero or more ROUTE sessions and constituent LCT channels in which the media content components of an ATSC 3.0 Service are delivered. The S-TSID also includes file metadata for the delivery object or object flow carried in the LCT channels of the Service, as well as additional information on the payload formats and content components carried in those LCT channels.

If components of a Service are delivered by a single Broadcast Stream or by multiple Broadcast Streams without channel bonding, the S-TSID associated with the Service shall only describe the ROUTE sessions and LCT channels for the components delivered in the Broadcast Stream in which the S-TSID is delivered. However, if components of a Service are delivered by more than one Broadcast Stream and channel bonding is applied, the S-TSID associated with the Service may need to describe the ROUTE sessions and LCT channels for the components delivered in other Broadcast Streams than the one in which the S-TSID is delivered. More details are described in Section 5.3.

See Annex A.3 and A.4 for additional information.

Table 7.2 contains the semantics of the S-TSID fragment. Note that the **`SrcFlow`** and **`RepairFlow`** elements of the S-TSID are defined in Annex A (see Sections A.1.2, A.1.3, A.3 and A.4).

The S-TSID shall be represented as XML documents and other dependent data type definitions containing an **`S-TSID`** root element that conforms to the definitions in the XML schema files that have the namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/
```

The definition of this schema is in the file, S-TSID-1.0-20170920.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"stsid"`.

While the indicated XML schema specifies the normative syntax of the **`S-TSID`** element, informative Table 7.2 below describes the structure of the **`S-TSID`** element in a more illustrative way. See Annex A.3 and A.4 for additional information.

The media type corresponding to fragments containing these files shall be as specified in Annex H.1.

Table 7.2 Semantics of the Service-based Transport Session Instance Description Fragment

Element and Attribute Names	Use	Data Type	Description
S-TSID			Service Transport Session Instance Description
RS	1..N		ROUTE session
@sIpAddr	0..1	stsid:IPv4addressType	Source IP address of this ROUTE session; mandatory for ROUTE session other than session carrying SLS (session signaled in SLT); defaults to session carrying SLS.
@dIpAddr	0..1	stsid:IPv4addressType	Destination IP address of this ROUTE session; mandatory for ROUTE session other than session carrying SLS (session signaled in SLT); defaults to session carrying SLS.
@dPort	0..1	unsignedShort	Destination port of this ROUTE session; mandatory for ROUTE session other than session carrying SLS (session signaled in SLT); defaults to session carrying SLS.
LS	1..N		LCT channel
@tsi	1	unsignedInt	TSI value
@bw	0..1	unsignedInt	Maximum bandwidth
@startTime	0..1	dateTime	Start time
@endTime	0..1	dateTime	End time
SrcFlow	0..1	stsid:srcFlowType	Information about the source flow
RepairFlow	0..1	stsid:rprFlowType	Information about the repair flow

The following text specifies the semantics of the portion of elements and attributes in the S-TSID fragment represented in Table 7.2.

S-TSID – Root element of the Service-based Transport Session Instance Description.

RS – A complex element whose subordinate elements and attributes contain information about the one or more ROUTE sessions in which the content components of this ATSC 3.0 service are carried.

@sIpAddr – A dotted-IPv4 source IP address whose value shall represent the source IP address for this ROUTE session. When this attribute is absent, the default value shall be the source IP address of the ROUTE session whose LCT channel carries the SLS fragments for this service. This attribute shall be present if this ROUTE session, whose LCT channel carries content component(s) of this service, is not the ROUTE session/LCT channel carrying the SLS fragments. The syntax shall be as defined in RFC 3986 [19] Section 3.2.2.

@dIpAddr – A dotted-IPv4 destination IP address whose value shall represent the destination IP addess for this ROUTE session. When this attribute is absent, the default value shall be the destination IP address of the ROUTE session whose LCT channel carries the SLS fragments for this service. This attribute shall be present if this ROUTE session, whose LCT channel carries content component(s) of this service, is not the ROUTE session/LCT channel carrying the SLS fragments. The syntax shall be as defined in RFC 3986 [19] Section 3.2.2.

@dPort – A destination port number whose value shall be associated with the dotted-IPv4 destination IP address for this ROUTE session. When this attribute is absent, the default value shall be the destination port number of the ROUTE session whose LCT channel carries the SLS fragments for this service. This attribute shall be present if this ROUTE session, whose

LCT channel carries content component(s) of this service, is not the ROUTE session/LCT channel carrying the SLS fragments.

LS – A complex element whose subordinate elements and attributes contain information about each of the one or more LCT channels that carry content component(s) of the ATSC 3.0 service to which the ROUTE session(s) identified in this S-TSID are associated. Any given LCT Channel shall carry either real-time content (DASH Media Segments and Initialization Segments) or non-real-time (locally-cached) content, but not both.

`@tsi` – A 32-bit unsigned integer which shall represent the value of the Transport Session Identifier (TSI) for this LCT channel and whose uniqueness shall be scoped by all ROUTE sessions described in this S-TSID.

`@bw` – A 32-bit unsigned integer which when present, shall represent the maximum bit-rate required by this LCT channel. It shall be specified by using the Application Specific (AS) bandwidth modifier, at the media level, in RFC 4566 [20]. The AS bandwidth for an LCT channel of the parent ROUTE session shall be the largest value among the sum of the sizes of all packets transmitted during any one second long period of the session, expressed in kilobits. The size of the packet shall be the complete packet, i.e. comprising IP, UDP and ROUTE headers, and the data payload. Absence of this attribute shall imply that the maximum bit-rate required by this LCT channel is unknown.

`@startTime` – This attribute shall represent the start time of this LCT channel, defined in accordance to the time description, i.e. “t=” line in RFC 4566 [20], and representing the “t=<start-time>” value, The only difference from RFC 4566 [20] is in the format of the session start time – i.e., instead of decimal representation of the NTP (Network Time Protocol) time value, in this specification, the session start time shall be represented by the “dateTime” XML datatype as defined in XSD Datatypes [43]. Absence of this attribute shall be an indication that the start time of this LCT channel occurred at some time in the past.

`@endTime` – This attribute shall represent the end time of this LCT channel, defined in accordance to the time description, i.e. “t=” line in RFC 4566 [20], and representing the “t=<end-time>” value, The only difference from RFC 4566 [20] is in the format of the session end time – i.e., instead of decimal representation of the NTP (Network Time Protocol) time value, in this specification, the session end time shall be represented by the dateTime XML datatype as defined in XSD Datatypes [43]. Absence of this attribute shall be an indication that the end time of this LCT channel will occur at an undefined time in the future.

SrcFlow – A complex element whose subordinate elements and attributes contain shall information about the source flow (as defined in Annex A, Section A.3), if present, within its parent LCT channel. Absence of this element shall imply that no source flow, but only a repair flow, is carried in this LCT channel.

RepairFlow – A complex element whose subordinate elements and attributes shall contain information about the repair flow (as defined in Annex A, Section A.4), if present, within its parent LCT channel. Absence of this element shall imply that no repair flow, but only a source flow, is carried in this LCT channel.

7.1.5 DASH Media Presentation Description (MPD)

The MPD is an SLS metadata fragment which contains a formalized description of the DASH-IF [12] profile of a DASH Media Presentation, corresponding to a linear service of a given duration defined by the broadcaster (for example a single TV program, or the set of contiguous linear TV programs over a period of time). The contents of the MPD provide the resource identifiers for

Segments and the context for the identified resources within the Media Presentation. The data structure and semantics of the MPD fragment shall be according to the DASH Media Presentation Description.

In the context of ATSC 3.0 services, one or more of the DASH Representations conveyed in the MPD are carried over broadcast. The MPD may describe additional Representations delivered over broadband, e.g. in the case of a hybrid service, or to support service continuity in handoff from broadcast to broadcast due to broadcast signal degradation (e.g. driving through a tunnel).

7.1.5.1 Signaling for Staggercast Audio Representation

Staggercast is a robustness feature that can be optionally added to audio components. It consists of delivering a redundant version of a main audio component, possibly coded with lower quality (e.g. lower bitrate, number of channels, etc.) and with a significant delay ahead of the audio with which it is associated.

> Note*:* For live content, staggercast audio stream may be sent ahead of the main audio stream by, for instance, taking advantage of the internal delay of encoding a video GoP.

Receivers that support Staggercast feature can switch to the Staggercast stream should main audio become unavailable. The delivery delay between Staggercast audio and main audio should be chosen high enough to provide robustness given the sufficient time diversity between both audios.

7.1.5.2 Content ID for Content in ROUTE/DASH Services

When it is desired to associate a Content ID with linear content in an ATSC 3.0 service which uses ROUTE/DASH to deliver broadcast linear content, it shall be done as defined in DASH-IF [12].

7.1.6 Service Signaling Delivery

Service Signaling of a service shall be carried in an ALC/LCT transport channel as signaled in the SLT.

7.1.6.1 Signaling Description Encapsulation

One or more Service Layer Signaling fragments, along with the **`metadataEnvelope`** as defined in MBMS [14] Section 11.1.3, shall be included in a multipart/related container per RFC 2387 [17]. The **`metadataEnvelope`** fragment shall be the first object in the package. SLS fragments shall not be embedded in the **`metadataEnvelope`**; "referenced" mode shall be used. See MBMS [14] Section 5.2.6. In this structure, the **`metadataEnvelope`** is used to provide the identification, versioning and expiration of the associated SLS metadata fragments. The multipart/related package may be compressed using the generic gzip algorithm specified in RFC 1952 [16] as content/transport encoding. In addition, ATSC 3.0 receivers may utilize the template-based compression scheme as specified in Annex D.

7.1.6.2 Signaling Description Filtering

When processing SLS fragments, ATSC 3.0 receivers are expected to utilize a filtering scheme by inspecting the TOI field of the LCT header, which identifies the type of the Service Layer Signaling fragment(s) contained within the referenced object and indicates the version of the package containing one or more Service Layer Signaling fragment(s). The TOI field of the LCT header shall be constructed according to the rules specified in Annex C. This enables quick filtering for target LCT packets which carry Service Layer Signaling fragments of the desired type before recovering whole Service Layer Signaling fragment from those packets. An Extended FDT

Instance transported in the LCT channel referencing broadcast SLS fragments with TOI=0 shall be present. The TOI values in the Extended FDT Instance shall match the encoding specified in Annex C.

7.1.7 Associated Procedure Description (APD)

The APD is an SLS metadata fragment containing information for use in conjunction with certain parameters in the **`EFDT`** element of the S-TSID fragment (as described in Section A.3.3.2.6) to govern the optional use by the receiver of the HTTP file repair functionality. The file repair procedure corresponds to an HTTP request/response transaction whereby a receiver, unable to acquire the entire object delivered by ROUTE, can request and obtain the missing data via broadband from a file repair server.

The APD fragment provides temporal information, under the **`postFileRepair`** element, for the receiver, if it wishes to perform the file repair procedure to obtain missing data. The `@offsetTime` attribute of **`postFileRepair`** represents the time interval in seconds that the receiver shall wait, after the end of transmission for the file of interest has occurred, before it can start the file repair procedure. The means by which the receiver could determine the end of file transmission, and the associated time window within which it is allowed to perform file repair, are described in Section 7.1.7.2. The `randomTimePeriod` attribute of **`postFileRepair`** defines a time window within which the receiver shall calculate a random value. This value represents an additional wait time, after the initial, fixed delay conveyed by `@offsetTime` has transpired, by the receiver before it submits the file repair request. The purpose of the random wait is to better ensure statistically uniform distribution of file repair request traffic arriving at the file repair server, from multiple receivers.

Details of the file repair procedure, which makes use of information in the APD fragment as well as in the **`EFDT`**, are described in Section 8.3 below.

Table 7.3 contains the semantics of the APD fragment.

The APD shall be represented as an XML document containing an **`AssociatedProcedureDescription`** root element that conforms to the definitions in the XML schema that has the namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/APD/1.0/
```

The definition of this schema is in an XML schema file, APD-1.0-20170209.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"apd"`.

While the indicated XML schema specifies the normative syntax of the **`APD`** element, informative Table 7.3 below describes the structure of the **`APD`** element in a more illustrative way.

The media type corresponding to fragments containing these files shall be as specified in Annex H.4.

Table 7.3 Semantics of the Associated Procedure Description Fragment

Element or Attribute Name	Use	Data Type	Description
`AssociatedProcedureDescription`			Root element of the Associated Procedure Description.
`PostFileRepair`	1		Container for the temporal parameters pertaining to the file repair procedure.
`@offsetTime`	0..1	unsignedLong	A first wait interval for the receiver related to the file repair procedure.
`@randomTimePeriod`	1	unsignedLong	A second wait interval for the receiver related to the file repair procedure.

7.1.7.1 APD Semantics

The following text specifies the semantics of the elements and attributes in the APD fragment.

`AssociatedProcedureDescription` – Root element of the Associated Procedure Description.

`PostFileRepair` – Container of temporal information that govern the start time of the file repair procedure.

`@offsetTime` – A time interval in seconds that the receiver shall wait, after broadcast file transmission has ended, before it can begin the file repair procedure. If this attribute is absent or set to `"0"`, the receiver should not employ a wait time before computing a random time within the time window given by `@randomTimePeriod` to initiate the file repair request.

`@randomTimePeriod` – After the fixed delay conveyed by `@offsetTime` has transpired, this attribute, as part of the file repair procedure, defines a time window in seconds within which the receiver shall calculate a random value. The value of `@randomTimePeriod` represents an additional wait time by the receiver before it is permitted to to initiate the file repair request.

7.1.7.2 End of File Transmission and Start of File Repair Procedure

The permitted start and end times for the receiver to perform the file repair procedure, in case of unsuccessful broadcast file reception, and associated rules and parameters are as follows:

- The latest time that the file repair procedure may start is bound by the `@Expires` attribute of the **`EFDT.FDT-Instance`** element in the S-TSID metadata fragment.
- The receiver may choose to start the file repair procedure earlier, if it detects the occurrence of any on the following events:
  - Presence of the `Close Object` flag (B) in the LCT header [26] for the file of interest;
  - Presence of the `Close Session` flag (A) in the LCT header [26] before the nominal expiration of the Extended FDT Instance as defined by the `@Expires` attribute.

7.1.8 Distribution Window Description (DWD)

The Distribution Window Description provides the broadcast delivery schedule of NRT files associated with broadcaster applications. A broadcaster application may pertain to app-based enhancements for a linear service, or constitute a standalone, app-based service. The DWD contains identifiers of applications to which the NRT files delivered during the broadcast time intervals belong. It may include Filter Codes to enable selective download and caching of those files to support personalization, as well as indication of the specific files that will be transmitted during a given distribution window. Given the application signaling nature of this SLS fragment, additional details on the DWD are specified in A/337, Application Signaling [7].

7.2 MMTP-Specific Service Layer Signaling

MMTP-Specific Service Layer Signaling for linear services comprises the USBD fragment, the MMT Package (MP) table, the DWD fragment (see A/337 [7]) and the HELD fragment (see A/337 [7]). The USBD fragment contains service identification, references to other SLS information required to access the service and constituent media components, and the metadata to enable the receiver to determine the transport mode (broadcast and/or broadband) of the service components. The MP table for MPU components, referenced by the USBD, provides transport session descriptions for the MMTP sessions in which the media content components of an ATSC 3.0 service are delivered and the descriptions of the Assets carried in those MMTP sessions. The HELD fragment provides application(s) properties to load and unload an application including the information of application-related files of one or more applications associated with a service. The DWD fragment provides the broadcast delivery schedule of NRT files associated with broadcaster applications. Details about app-based enhancement signaling, including the HELD fragment and the DWD fragment, are specified in A/337, Application Signaling [7]. For hybrid delivery, the MMT-specific SLS can further include the MPD for broadband components as specified in Section 7.1.5. The data model of the SLS applicable to MMTP/MPU linear services, shown using UML convention, is shown in Figure 7.3.

The streaming content signaling component of the SLS for MPU components shall correspond to the MP table defined in subclause 10.3.9 of ISO/IEC 23008-1 [37]. The MP table provides a list of MMT Assets where each Asset corresponds to a single service component and the description of the location information for this component. When an ATSC 3.0 service contains more than one video Asset, the value of default_asset_flag in the asset-loop of an MP table shall be set to “0” for the video Asset corresponding to the Default Video and shall be set to “1” for the other video Assets. The MP table is further described as in Section 7.2.3.

USBD fragments may also contain a reference to the S-TSID as specified in Section 7.1.4 for locally-cached service content delivered by the ROUTE protocol.

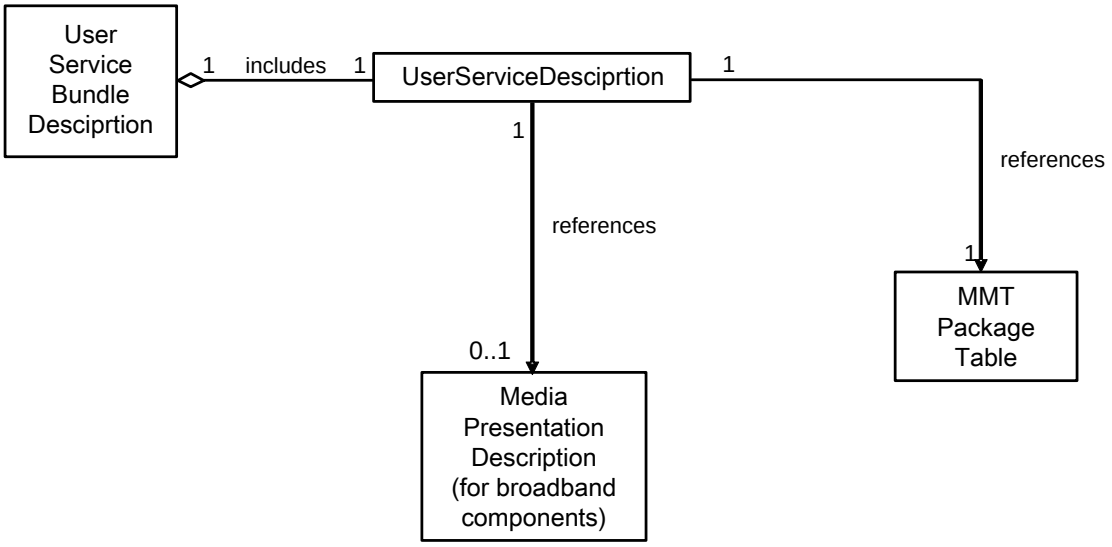


Figure 7.3 Service Layer Signaling data model for MMTP/MPU Linear Services.

7.2.1 User Service Description for MMTP

The top level or entry point SLS fragment is the USBD fragment. The USBD fragment for ATSC 3.0 is modeled on the USBD fragment as defined by MBMS [14], with the following extensions:

- Child attributes `@serviceId` and `@serviceStatus` under the element **`UserServiceDescription`**;
- Child elements **`ContentAdvisoryRating`** and **`OtherRatings`** under the element **`UserServiceDescription`**;
- Child element **`Channel`** and its child attributes `@serviceGenre`, `@serviceIcon`, and child element **`ServiceDescription`** and its child attributes `@serviceDescrText`, `@serviceDescrLang` under the element **`UserServiceDescription`**;
- Child element **`MPUComponent`** and its child attributes `@mmtPackageId` and `@nextMMTPackageId`, `@contendIdSchemdIdUri`, `@contentIdvalue`, `@nextContentIdSchemeIdUri` and `@nextContentIdValue` under the element **`UserServiceDescription`**;
- Child element **`ROUTEComponent`** and its child attributes `@sTSIDUri`, `@apdURI` `@sTSIDDestinationIpAddress`, `@sTSIDDestinationUdpPort`, `@sTSIDSourceIPAddress`, `@sTSIDMajorProtocolVersion`, `@sTSIDMinorProtocolVersion` under the element **`UserServiceDescription`**, as service signaling data to support the delivery of locally-cached service content via ROUTE protocol;
- Child element **`BroadbandComponent`** and its child attribute `@fullMPDUri` under the element **`UserServiceDescription`**; and
- Child element **`ComponentInfo`** and its child attributes `@componentType`, `@componentRole`, `@componentProtectedFlag`, `@componentId`, `@componentName` under the element **`UserServiceDescription`.**

It is recommended that the same information should not be repeated in the MMTP USBD when it is carried in the service announcement. In this case information in Service Announcement should take precedence.

A large number of the attributes and elements in the MBMS USBD fragment are optional and not relevant to ATSC 3.0. Table 7.4 shows the elements and attributes that would be used in practice for ATSC 3.0 service delivery.

The **`BundleDescriptionMMT`** shall be represented as an XML document containing a **`BundleDescriptionMMT`** root element that conforms to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/MMTUSD/1.0/
```

The definition of this schema is in the XML schema file, MMTUSD-1.0-20170922.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"mmtusd"`.

While the XML schemas identified above specify the normative syntax of the elements specified in this ATSC 3.0 standard, informative Table 7.4 below describes the structure of the **`BundleDescriptionMMT`** element in a more illustrative way.

The media type corresponding to fragments containing these files shall be as specified in Annex H.3.

Table 7.4 XML Format of the User Service Bundle Description Fragment for MMTP

Element or Attribute Name	Use	Data Type	Description
BundleDescriptionMMT			Root element of the User Service Bundle Description for MMTP.
UserServiceDescription	1		A single instance of an ATSC 3.0 Service.
@serviceId	1	unsignedShort	Reference to corresponding service entry in LLS(SLT).
@serviceStatus	0..1	boolean	Specify the status of this service as active or inactive.
Name	0..N	string	Name of the ATSC 3.0 service.
@lang	1	lang	Language of the ATSC 3.0 service name.
ServiceLanguage	0..N	lang	Available languages of the ATSC 3.0 service
ContentAdvisoryRating	0..N	sa:CARatingType	Specifies the RRT-based content advisory rating, as defined in the ATSC 3.0 Service Announcement specification A/332 [5].
OtherRatings	0..N	sa:OtherRatingType	Specifies the Non-RRT content advisory rating, as defined in the ATSC 3.0 Service Announcement specification
Channel	1		Contains information about the service
@serviceGenre	0..1	unsignedByte	Attribute indicates primary genre of the service.
@serviceIcon	1	anyURI	Attribute indicates the Uniform Resource Locator (URL) for the icon used to represent this service.
ServiceDescription	0..N		Contains service description possibly in multiple languages.
@serviceDescrText	1	string	Attribute indicates description of the service.
@serviceDescrLang	0..1	lang	Attribute indicates the language of the `serviceDescrText`.
MPUComponent	0..1		A description about the contents components of ATSC 3.0 Service delivered as MPUs
@mmtPackageId	1	string	Reference to a MMT Package for content components of the ATSC 3.0 Service delivered as MPUs.
@contentIdSchemeIdUri	0..1	anyURI	Attribute indicates a URI to identify the scheme for Content ID associated to the current MMT Package.
@contentIdValue	0..1	string	Attribute indicates the value for Content ID associated to the current MMT Package.
@nextMMTPackageId	0..1	string	Reference to a MMT Package to be used after the one referenced by

			`@mmtPackageId` in time for content components of the ATSC 3.0 Service delivered as MPUs.
`@nextContentIdSchemeIdUri`	0..1	anyURI	Attribute indicates a URI to identify the scheme for Content ID associated to the next MMT Package.
`@nextContentIdValue`	0..1	string	Attribute indicates the value for Content ID associated to the next MMT Package.
`ROUTEComponent`	0..1		A description about locally-cached service content of ATSC 3.0 Service delivered by ROUTE.
`@sTSIDUri`	1	anyURI	Reference to the S-TSID fragment which provides access related parameters to the Transport sessions carrying contents of this ATSC 3.0 Service.
`@apdUri`	0..1	anyURI	Reference to the APD fragment which contains file repair related information.
`@sTSIDDestinationIpAddress`	0..1	IPv4address	A string containing the dotted-IPv4 destination address of the packets carrying S-TSID for this service.
`@sTSIDDestinationUdpPort`	1	unsignedShort (port)	Port number of the packets carrying S-TSID for this service.
`@sTSIDSourceIpAddress`	1	IPv4address	A string containing the dotted-IPv4 source address of the packets carrying S-TSID for this service.
`@sTSIDMajorProtocolVersion`	0..1	unsignedByte	Major version number of the protocol used to deliver the S-TSID for this service.
`@sTSIDMinorProtocolVersion`	0..1	unsignedByte	Minor version number of the protocol used to deliver the S-TSID for this service.
`BroadbandComponent`	0..1		A description about the contents components of ATSC 3.0 Service delivered by broadband.
`@fullMPDUri`	1	anyURI	Reference to an MPD fragment which contains descriptions for contents components of the ATSC 3.0 Service delivered over broadband.
`ComponentInfo`	0..N		Contains information about components available in the service. For each component includes information about component type, component role, component name, component identifier, component protection flag.
`@componentType`	1	unsignedByte	Attribute indicates the type of this component.
`@componentRole`	1	unsignedByte	Attribute indicates the role or kind of this component.

			`@componentProtectedFlag`	0..1	boolean	Attribute indicates if this component is protected
			`@componentId`	1	string	Attribute indicates the identifier of this component.
			`@componentName`	0..1	string	Attribute indicates the human readable name of this component.

7.2.1.1 User Service Description for MMTP – Semantics

The following text specifies the semantics of the elements and attributes in the User Service Description for MMTP.

`BundleDescriptionMMT` – Root element of the User Service Bundle Description for MMTP.

`UserServiceDescription` – Single instance of an ATSC 3.0 Service.

`@serviceId` – Reference to corresponding service entry in LLS (SLT). The value of this attribute is the same value of `serviceId` assigned to the service entry. (Same as given in Table 7.1).

`@serviceStatus` – A Boolean attribute which shall convey the current status of this service as being active or inactive. A value of `"true"` shall indicate that the service is active. A value of `"false"` shall indicate that the service is inactive. The default value shall be `"true"`. (Same as given in Table 7.1).

`Name` – Name of the ATSC 3.0 service in the language specified by `@lang` attribute. (Same as given in Table 7.1). When **`Name`** is not present, there is no default value for the name of the ATSC 3.0 service.

`@lang` – Language of the ATSC 3.0 service name. The language shall be specified according to BCP 47 [32]. (Same as given in Table 7.1).

`ServiceLanguage` – Available languages of the ATSC 3.0 service. The language shall be specified according to BCP 47 [32]. (Same as given in Table 7.1). When **`serviceLanguage`** is not present, there is no default value.

`ContentAdvisoryRating` – Specifies the content advisory rating, as defined in the ATSC 3.0 Service Announcement specification A/332 [5]. The syntax and semantics of this element shall be the same as the **`ContentAdvisoryRatings`** element specified in the Service fragment of the ATSC 3.0 Service Announcement specification A/332 [5]. When **`ContentAdvisoryRating`** is not present, there is no default value for the RRT-based content advisory rating of the service.

`OtherRatings` – Specifies the non-RRT content advisory rating, as defined in the ATSC 3.0 Service Announcement specification A/332 [5]. The syntax and semantics of this element shall be the same as the **`OtherRatings`** element specified in the Service fragment of the ATSC 3.0 Service Announcement specification A/332 [5]. Each **`OtherRatings`** element shall have a unique `@ratingScheme` value. When **`OtherRatings`** is not present, there is no default value for the non-RRT based content advisory rating of the service.

`Channel` – This element contains information about the service.

`@serviceGenre` – This optional attribute indicates the primary genre category of the service. The `<classificationSchemeURI>` is `http://www.atsc.org/XMLSchemas/mh/2009/1.0/genre-cs/` and the value of `serviceGenre` shall match a `termID` value from the classification schema in Annex B of A/153 Part 4 [2]. When `@serviceGenre` is not present, there is no default value for the primary genre of the service.

`@serviceIcon` – This attribute indicates the Uniform Resource Locator (URL) for the icon used to represent this service.

`ServiceDescription` – Contains service description possibly in multiple languages.

`@serviceDescrText` – This attribute indicates description of the service.

`@serviceDescrLang` – This attribute indicates the language of the `@serviceDescrText`. Semantics of `xml:lang` shall be followed. When `@serviceDescrLang` is not present, the language is identified to be "en".

`MPUComponent` – A description about the contents components of ATSC 3.0 Service delivered as MPUs.

`@mmtPackageId` – Reference to a MMT Package for content components of the ATSC 3.0 Service delivered as MPUs.

`@contentIdschemeIdUri` – This attribute shall indicate a URI to identify the scheme for Content ID associated to the current MMT Package. The semantics of `@contentIdValue` attribute are specific to the scheme specified by this attribute. The allowed values are:

- `urn:eidr` for EIDR Contend ID; and
- the "Designator" for either the "full" or "compact" encoding as defined in SMPTE 2092-1 [41].

Other schemes may be used, including user private schemes, by using appropriately unique values of `@contentIdSchemeIdUri`.

When `@contentIdschemeIdUri` is not present, there is no default value and there is no information present in this User Service Description about content ID for the current MMT package.

`@contentIdvalue` – This attribute shall specify the value of Content ID associated to the current MMT Package according to the Contend ID system identified by `@contentIdSchemeIdUri` attribute. The "EIDR Content ID" shall be a valid canonical EIDR entry as defined in [42]. The "Ad-ID Content ID" shall be a valid Ad-ID entry as defined in [41].

When `@contentIdschemeIdUri` is not present, `@contentIdschemeIdvalue` shall not be present. When `@contentIdschemeIdUri` is present, `@contentIdschemeIdvalue` shall be present.

When `@contentIdschemeIdValue` is not present, there is no default value and there is no information present in this User Service Description about content ID for the current MMT Package.

`@nextMMTPackageId` – Reference to a MMT Package to be used after the one referenced by `@mmtPackageId` in time for content components of the ATSC 3.0 Service delivered as MPUs.

`@nextContentIdSchemeIdUri` – This attribute shall indicate a URI to identify the scheme for Content ID associated to the next MMT Package. The semantics of `@nextContentIdValue` attribute are specific to the scheme specified by this attribute. The allowed values are:

- `urn:eidr` for EIDR Contend ID; and
- the "Designator" for either the "full" or "compact" encoding as defined in SMPTE 2092-1 [41].

Other schemes may be used, including user private schemes, by using appropriately unique values of `@nextContentIdSchemeIdUri`.

When `@nextContentIdSchemeIdUri` is not present there is no default value and there is no information present in this User Service Description about content ID for the next MMT Package.

`@nextContentIdvalue` – This attribute shall specify the value of Content ID associated to the next Package according to the Contend ID system identified by `@nextContentIdSchemeIdUri` attribute. The "EIDR Content ID" shall be a valid canonical EIDR entry as defined in [42]. The "Ad-ID Content ID" shall be a valid Ad-ID entry as defined in [41].

When `@nextContentIdSchemeIdUri` is not present, `@nextContentIdSchemeIdValue` shall not be present. When `@nextContentIdSchemeIdUri` is present, `@nextContentIdSchemeIdValue` shall be present.

When `@nextContentIdSchemeIdValue` is not present, there is no default value and there is no information present in this User Service Description about content ID for the current MMT Package.

`ROUTEComponent` – A description about the contents components of ATSC 3.0 Service delivered by ROUTE.

`@sTSIDUri` – Reference to the S-TSID fragment which provides service access related parameters to the Transport sessions carrying contents of this ATSC 3.0 Service.

`@apdUri` – This optional attribute shall provide a reference to the APD fragment which provides file repair related information for the content components of ATSC 3.0 Service delivered by ROUTE. This attribute points to an APD fragment as described in Section 7.1.7.

When `@apdURI` is present, at least one **`Alternate-Content-Location-1`** element shall be present in the **`EFDT`** element of the S-TSID fragment pointed by the **`ROUTEComponent`**`@sTSIDUri`. When `@apdUri` is not present, there is no default value.

`@sTSIDDestinationIpAddress` – A string containing the dotted-IPv4 destination address of the packets carrying S-TSID for this service. When not present the value of this attribute is inferred to be current MMTP session's destination IP address. The syntax shall be as defined in RFC 3986 [19] section 3.2.2.

`@sTSIDDestinationUdpPort` – A string containing the UDP port number of the packets carrying S-TSID for this service.

`@sTSIDSourceIpAddress` – A string containing the dotted-IPv4 source address of the packets carrying S-TSID for this service. The syntax shall be as defined in RFC 3986 [19] section 3.2.2.

`@sTSIDMajorProtocolVersion` – Major version number of the protocol used to deliver the S-TSID for this service. When not present the value of this attribute is inferred to be 1.

`@sTSIDMinorProtocolVersion` – Minor version number of the protocol used to deliver the S-TSID for this service. When not present the value of this attribute is inferred to be 0.

`BroadbandComponent` – A description about the contents components of ATSC 3.0 Service delivered by broadband. At least one of **`MPUComponent`**, **`ROUTEComponent`** or **`BroadbandComponent`** shall be present.

`@fullMPDUri` – Reference to an MPD fragment which contains descriptions for content components of the ATSC 3.0 Service delivered over broadband.

`ComponentInfo` – Contains information about components available in the service. For each component includes information about component type, component role, component name, component identifier, component protection flag. This element shall be present when **`MPUComponent`** is present.

`@componentType` – This attribute indicates the type of this component. Value of 0 indicates an audio component. Value of 1 indicates a video component. Value of 2 indicates a closed caption component. Values 3 to 7 are reserved.

`@componentRole` – This attribute indicates the role or kind of this component.

For audio (when `componentType` attribute above is equal to 0): values of `componentRole` attribute are as follows: 0 = Complete main, 1 = Music and Effects, 2 = Dialog, 3 = Commentary, 4 = Visually Impaired, 5 = Hearing Impaired, 6 = Voice-Over, 7-254= reserved, 255 = unknown.

For Video (when `componentType` attribute above is equal to 1) values of `componentRole` attribute are as follows: 0 = Primary video, 1-254 = reserved, 255 = unknown.

For Closed Caption component (when `componentType` attribute above is equal to 2) values of `componentRole` attribute are as follows: 0 = Normal, 1 = Easy reader, 2-254 = reserved, 255 = unknown.

When `@componentType` attribute above has a value between 3 to 7, inclusive, the `@componentRole` value shall be equal to 255.

`@componentProtectedFlag` – This attribute indicates if this component is protected (e.g. encrypted). When this flag is set to a value of 1 this component is protected (e.g. encrypted). When this flag is set to a value of 0 this component is not protected (e.g. encrypted). When not present the value of `componentProtectedFlag` attribute is inferred to be equal to 0.

`@componentId` – This attribute indicates the identifier of this component. The value of this attribute shall be the same as the asset_id in the MP table corresponding to this component.

`@componentName` – This attribute indicates the human readable name of this component. When `@componentName` is not present, there is no default value.

7.2.2 DASH Media Presentation Description (MPD)

The DASH Media Presentation Description is an SLS metadata fragment corresponding to a linear service of a given duration defined by the broadcaster (for example a single TV program, or the set of contiguous linear TV programs over a period of time). The contents of the MPD provide the resource identifiers for Segments and the context for the identified resources within the DASH Media Presentation. The data structure and semantics of the MPD shall be according to the DASH Media Presentation Description.

To ensure seamless playback across Period boundaries, ahead of the Period boundary an MPD must be provided which includes both the current Period and the next Period. The MPD describing the next Period must be provided at minimum some amount of time before the end of the current Period (i.e., the Period for which DASH Media Segments are currently being presented). The minimum timing is specified in DASH-IF [12] profile.

In the context of ATSC 3.0 services, an MPD delivered by an MMTP session shall only describe Representations delivered over broadband, e.g. in the case of a hybrid service, or to support service continuity in handoff from broadcast to broadband due to broadcast signal degradation (e.g. driving under a mountain or through a tunnel).

7.2.3 MMTP-Specific Signaling Message

When MMTP sessions are used to carry an ATSC 3.0 streaming service, MMTP-specific signaling messages specified in Clause 10 of ISO/IEC 23008-1 [37] are delivered in binary format by MMTP packets according to Signaling Message Mode specified in subclause 9.3.4 of ISO/IEC 23008-1 [37]. The value of the packet_id field of MMTP packets carrying Service Layer Signaling shall be set to 0x0000 except for MMTP packets carrying MMTP-specific signaling messages specific to an Asset, which shall be set to 0x0000 or the same packet_id value as the MMTP packets carrying the Asset. Identifiers referencing the appropriate Package for each ATSC 3.0 Service are signaled by the USBD fragment as described in Table 7.4. MMT Package Table (MPT) messages with matching MMT_package_id shall be delivered on the MMTP session signaled in the SLT. Each

MMTP session carries MMTP-specific signaling messages specific to its session or each asset delivered by the MMTP session.

The following MMTP messages shall be delivered by the MMTP session signaled in the SLT:

- MMT Package Table (MPT) message: This message carries an MP (MMT Package) table which contains the list of all Assets and their location information as specified in subclause 10.3.4 of ISO/IEC 23008-1) [37].
- MMT ATSC3 (MA3) message mmt_atsc3_message(): This message carries system metadata specific for ATSC 3.0 services including Service Layer Signaling as specified in Section 7.2.3.1.

The following MMTP messages shall be delivered by the MMTP session signaled in the SLT, if required:

- Media Presentation Information (MPI) message: This message carries an MPI table which contains the whole document or a subset of a document of presentation information. An MP table associated with the MPI table also can be delivered by this message (see subclause 10.3.3 of ISO/IEC 23008-1) [37];

The following MMTP messages shall be delivered by the MMTP session carrying an associated Asset and the value of the packet_id field of MMTP packets carrying them shall be set to the same as the MMTP packets carrying the Asset::

- Hypothetical Receiver Buffer Model message: This message carries information required by the receiver to manage its buffer (see subclause 10.4.2 of ISO/IEC 23008-1 [37]);
- Hypothetical Receiver Buffer Model Removal message: This message carries information required by the receiver to manage its MMT de-capsulation buffer (see subclause 10.4.9 of ISO/IEC 23008-1) [37];

7.2.3.1 mmt_atsc3_message() MMTP-Specific Signaling Message

An MMTP-specific signaling message mmt_atsc3_message() is defined to deliver information specific to ATSC 3.0 services. The value assigned to message_id for the mmt_atsc3_message() is in the “user private” range per subclause 10.7 of ISO/IEC 23008-1 [37]. The syntax of this message shall be as provided in Table 7.5.

The text following Table 7.5 shall describe the semantics of each field in mmt_atsc3_message().

Table 7.5 Bit Stream Syntax for mmt_atsc3_message()

Syntax	No. of Bits	Format
mmt_atsc3_message() {		
message_id	16	uimsbf
version	8	uimsbf
length	32	uimsbf
message payload {		
service_id	16	uimsbf
atsc3_message_content_type	16	uimbsf
atsc3_message_content_version	8	uimbsf
atsc3_message_content_compression	8	uimbsf
URI_length	8	uimbsf
for (i=0;i< URI_length;i++) {		
URI_byte	8	uimsbf
}		
atsc3_message_content_length	32	uimsbf
for (i=0;i<atsc3_message_content_length;i++) {		
atsc3_message_content_byte	8	uimsbf
}		
for (i=0;i<length-11-URI_length-atsc3_message_content_length) {		
reserved	8	uimsbf
}		
}		
}		

message_id – A 16-bit unsigned integer field that shall uniquely identify the mmt_atsc3_message(). The value of this field shall be 0x8100.

version – An 8-bit unsigned integer field that shall be incremented by 1 any time there is a change in the information carried in this message. When the version field reaches its maximum value of 255, its value shall wraparound to 0.

length – A 32-bit unsigned integer field that shall provide the length of mmt_atsc3_message() in bytes, counting from the beginning of the next field to the last byte of the mmt_atsc3_message().

service_id – A 16-bit unsigned integer field that shall associate the message payload with the service identified in the `serviceId` attribute given in the SLT.

atsc3_message_content_type – A 16-bit unsigned integer field that shall uniquely identify the type of message content in the mmt_atsc3_message() payload, coded per Table 7.6 below.

Table 7.6 Code Values for atsc3_message_content_type

atsc3_message_content_type	Meaning
0x0000	ATSC Reserved
0x0001	`UserServiceDescription` as given in Table 7.4.
0x0002	MPD as given in DASH-IF [12].
0x0003	HELD as given in A/337, Application Signaling [7].
0x0004	Application Event Information as given in A/337, Application Signaling [7].
0x0005	Video Stream Properties Descriptor (Sec. 7.2.3.2)
0x0006	ATSC Staggercast Descriptor (Sec. 7.2.3.3)
0x0007	Inband Event Descriptor as given in A/337, Application Signaling [7].
0x0008	Caption Asset Descriptor (Sec. 7.2.3.5)
0x0009	Audio Stream Properties Descriptor (Sec. 7.2.3.4)
0x000A	DWD as given in A/337, Application Signaling [7].
0x000B~0xFFFF	ATSC Reserved

atsc3_message_content_version – An 8-bit unsigned integer field that shall be incremented by 1 any time there is a change in the mmt_atsc3_message content identified by a service_id, and atsc_message_content_type pair and URI if present. When the atsc3_message_content_version field reaches its maximum value, its value shall wrap around to 0.

atsc3_message_content_compression – An 8-bit unsigned integer field that shall identify the type of compression applied to the data in atsc3_message_content_byte**.**

Table 7.7 Code Values for atsc3_message_content_compression

atsc3_message_content_compression	Meaning
0x00	ATSC Reserved
0x01	No compression has been applied
0x02	gzip specified in RFC 1952 [16] has been applied
0x03	The template-based compression scheme as specified in Annex D has been applied
0x04~0xFF	ATSC Reserved

URI_length – An 8-bit unsigned integer field that shall provide the length of the URI uniquely identifying the message payload across services. If the URI is not present, the value of this field shall be set to 0. When this mmt_atsc3_message() carries an MPD (i.e. atsc3_message_content_type = 0x0002), the URI shall be present.

URI_byte – An 8-bit unsigned integer field that shall contain a UTF-8 character of the URI associated with the content carried by this message excluding the terminating null character, as per RFC 3986 [19]. This field when present shall be used to identify delivered message payloads. The URI can be used by system tables to reference tables made available by delivered message payloads.

atsc3_message_content_length – A 32-bit unsigned integer field that shall provide the length of the content carried by this message.

atsc3_message_content_byte – An 8-bit unsigned integer field that shall contain a byte of the content carried by this message.

7.2.3.2 Video Stream Properties Descriptor

Each video asset video_stream_properties_descriptor() provides information about its associated video stream. This includes information about resolution, chroma format, bit depth, temporal scalability, bit-rate, picture-rate, 3D, color characteristics, profile, tier, and level.

7.2.3.2.1 Syntax

The syntax for the video_stream_properties_descriptor() shall conform to Table 7.8. The semantics of the fields in the video_stream_properties_descriptor() shall be as given immediately below the table.

Table 7.8 Bit Stream Syntax for Video Stream Properties Descriptor

Syntax	No. of Bits	Format
video_stream_properties_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
number_of_assets	8	uimsbf
for (i=0;i<number_of_assets;i++) {		
asset_id_length	32	uimsbf
for (i=0; i<asset_id_length; i++) {		
asset_id_byte	8	uimsbf
}		
codec_code	4*8	uimsbf
temporal_scalability_present	1	bslbf
scalability_info_present	1	bslbf
multiview_info_present	1	bslbf
res_cf_bd_info_present	1	bslbf
pr_info_present	1	bslbf
br_info_present	1	bslbf
color_info_present	1	bslbf
reserved	1	‘1’
if (temporal_scalability_present) {		
max_sub_layers_instream /* s */	6	uimsbf
sub_layer_profile_tier_level_info_present	1	bslbf
temporal_filter_present	1	bslbf
tid_max	3	uimsbf
tid_min	3	uimsbf
If (temporal_filter_present) {		
tfweight	2	uimsbf
} else {		
reserved2	2	‘11’
}		
}		
if (scalability_info_present) {		
scalability_info()	8	Table 7.10
}		
if (multiview_info_present) {		
multiview_info()	40	Table 7.11
}		
if (res_cf_bd_info_present) {		

res_cf_bd_prop_info()	48	Table 7.12
}		
if (pr_info_present) {		
if (sub_layer_profile_tier_level_info_present) {		
pr_info(max_sub_layers_instream-1)	var	Table 7.13
} else {		
pr_info(0)	var	Table 7.13
}		
}		
if (br_info_present) {		
if (sub_layer_profile_tier_level_info_present) {		
br_info(max_sub_layers_instream-1)	32*(s-1)	Table 7.14
} else {		
br_info(0)	32	Table 7.14
}		
}		
if (color_info_present) {		
color_info()	var	Table 7.15
}		
if (sub_layer_profile_tier_level_info_present) {		
profile_tier_level(1,max_sub_layers_instream-1)	var	H.265
} else {		
profile_tier_level(1,0)	var	H.265
}		
}		
}		

descriptor_tag – This 16-bit unsigned integer shall have the value 0x0005, identifying this descriptor as the video_stream_properties_descriptor().

descriptor_length – This 16-bit unsigned integer shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

number_of_assets – An 8-bit unsigned integer field that shall specify the number of video assets described by this descriptor.

asset_id_length – This 32-bit unsigned integer field shall specify the length in bytes of the video asset id.

asset_id_byte – An 8-bit unsigned integer field that shall contain a byte of the video asset id.

codec_code – This field shall specify a 4-character code for a codec. The value of these four characters shall be one of 'hev1', 'hev2', 'hvc1', 'hvc2', 'lhv1' or 'lhe1' with semantic meaning for these codes as specified in ISO/IEC 14496-15 [35] as amended.

temporal_scalability_present – This 1-bit Boolean flag shall indicate, when set to ‘1’, that the elements max_sub_layers_present and sub_layer_profile_tier_level_info_present are present and temporal scalability is provided in the asset. When set to ‘0’, the flag shall indicate that the elements max_sub_layers_present and sub_layer_profile_tier_level_info_present are not present and temporal scalability is not provided in the asset.

scalability_info_present – This 1-bit Boolean flag shall indicate, when set to ‘1’, that the elements in the scalability_info() structure are present. When set to ‘0’, the flag shall indicate that the elements in the scalability_info() structure are not present.

multiview_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that the elements in the multiview_info() structure are present. When set to '0', the flag shall indicate that the elements in the multiview_info() structure are not present.

res_cf_bd_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that the elements in the res_cf_bd_info() structure are present. When set to '0', the flag shall indicate that the elements in the res_cf_bd_info() structure are not present.

pr_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that the elements in the pr_info() structure are present. When set to '0', the flag shall indicate that the elements in the pr_info() structure are not present.

br_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that the elements in the br_info() structure are present. When set to '0', the flag shall indicate that the elements in the br_info() structure are not present.

color_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that the elements in the color_info() structure are present. When set to '0', the flag shall indicate that the elements in the color_info() structure are not present.

max_sub_layers_instream – This 6-bit unsigned integer shall specify the maximum number of temporal sub-layers that may be present in each Coded Video Sequence (CVS) in the asset. The value of max_sub_layers_instream shall be in the range of 1 to 7, inclusive.

sub_layer_profile_tier_level_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that profile, tier, and level information be present for temporal sub-layers in the asset. When set to '0', the flag shall indicate that the profile, tier, and level information is not present for temporal sub-layers in the asset. When not present sub_layer_profile_tier_level_info_present shall be inferred to be equal to 0.

temporal_filter_present – This 1-bit Boolean flag shall indicate, when set to '1', that the element tfweight is present and temporal filtering information is provided for the asset. When set to '0', the flag shall indicate that the element tfweight is not present and temporal filtering information is not provided for the asset.

tid_max – This 3-bit field shall indicate the maximum value of TemporalId (as defined in Rec. ITU-T H.265 [38]) of all access units for this video asset. tid_max shall be in the range of 0 to 6, inclusive. tid_max shall be greater than or equal to tid_min.

tid_min – This 3-bit field shall indicate the minimum value of TemporalId (as defined in Rec. ITU-T H.265 [38]) of all access units for this video asset. tid_min shall be in the range of 0 to 6, inclusive.

tfweight – This 2-bit unsigned integer field shall indicate the values of temporal filtering parameters temporal_filter_w1 and temporal_filter_w2 as shown in Table 7.9. temporal_filter_w1 parameter shall indicate the weight of the temporally preceding temporal sub-layer 1 picture that contributes to the current temporal sub-layer 0 picture. The value of temporal_filter_w1 shall be as shown in Table 7.9. temporal_filter_w2 parameter shall indicate the weight of the high frame rate picture (not provided in the raw stream) in the current temporal position that contributes to the current temporal sub-layer 0 picture. The value of temporal_filter_w2 shall be as shown in Table 7.9.

Table 7.9 Values of Multiple Frame Rate Temporal Filtering Parameters

tfweight	temporal_filter_w2	temporal_filter_w1
‘00’	4/5	1/5
‘01’	2/3	1/3
‘10’	4/7	3/7
‘11’	1/2	1/2

The value of temporal_filter_w1 plus temporal_filter_w2 shall equal 1.

profile_tier_level(profileFPResentFlag, maxSubLayersMinus1) – This variable-sized field shall provide the profile, tier, level syntax structure as described in H.265 (10/2014) HEVC specification Section 7.3.3 [38].

7.2.3.2.1.1 Scalability Information

Table 7.10 Bit Stream Syntax for Scalability Information

Syntax	No. of Bits	Format
scalability_info() {		
asset_layer_id	6	uimsbf
reserved	2	‘11’
}		

asset_layer_id – This 6-bit unsigned integer field shall specify the nuh_layer_id [38] for this asset. The value of asset_layer_id shall be in the range of 0 to 62, inclusive.

When the value of scalable_info_present is equal to 1 or the value of multiview_info_present is equal to 1, the Dependency Descriptor specified in subclause 10.5.3 of MMT specification [37] shall be included in MPT for each asset. In this case the num_dependencies element in MMT Dependency Descriptor shall indicate the number of layers that the asset_layer_id for this asset is dependent on.

7.2.3.2.1.2 MultiView Information

Table 7.11 Bit Stream Syntax for Multi-View Information

Syntax	No. of Bits	Format
multiview_info() {		
view_nuh_layer_id	6	uimsbf
view_pos	6	uimsbf
reserved	4	‘1111’
min_disp_with_offset	11	uimsbf
max_disp_range	11	uimsbf
reserved	2	‘11’
}		

view_nuh_layer_id – This 6-bit unsigned integer field shall specify the nuh_layer_id [38] for the view represented by this asset. The value of view_nuh_layer_id shall be in the range of 0 to 62, inclusive.

view_pos – This 6-bit unsigned integer field shall specify the order of the view with nuh_layer_id [38] equal to view_nuh_layer_id among all the views from left to right for the purpose of display, with the order for the left-most view being equal to 0 and the value of the order increasing by 1 for next view from left to right. The value of view_pos[i] shall be in the range of 0 to 62, inclusive.

Note: Multiple video assets having the same view_nuh_layer_id value can have different view_pos values.

min_disp_with_offset – This 11-bit unsigned integer field shall specify minus 1024 of the minimum disparity, in units of luma samples, between pictures of any spatially adjacent views among the applicable views in an access unit. The value of min_disp_with_offset shall be in the range of 0 to 2047, inclusive.

max_disp_range – This 11-bit unsigned integer field shall specify the maximum disparity, in units of luma samples, between pictures of any spatially adjacent views among the applicable views in an access unit. The value of max_disp_range shall be in the range of 0 to 2047, inclusive.

7.2.3.2.1.3 Resolution, Chroma Format, Bit-depth and Video Properties Information:

Table 7.12 Bit Stream Syntax for Resolution, Chroma Format, Bit-Depth

Syntax	No. of Bits	Format
res_cf_bd_prop_info() {		
pic_width_in_luma_samples	16	uimsbf
pic_height_in_luma_samples	16	uimsbf
chroma_format_idc	2	uimsbf
if (chroma_format_idc == 3) {		
separate_colour_plane_flag	1	bslbf
reserved	3	‘111’
} else {		
reserved	4	‘1111’
}		
video_still_present	1	bslbf
video_24hr_pic_present	1	bslbf
bit_depth_luma_minus8	4	uimsbf
bit_depth_chroma_minus8	4	uimsbf
}		

pic_width_in_luma_samples, pic_width_in_chroma_samples, chroma_format_idc, separate_colour_plane_flag, bit_depth_luma_minus8, bit_depth_chroma_minus8 elements respectively shall have the same semantic meanings as the elements with the same name in H.265 (10/2014) HEVC specification Section 7.4.3.2 (Sequence parameter set RBSP semantics) [38].

video_still_present – This 1-bit Boolean flag when set to ‘1’, shall indicate that the video asset may include HEVC still pictures as defined in ISO/IEC 13818-1 [33]. When set to ‘0’, the flag shall indicate that the video asset shall not include HEVC still pictures as defined in ISO/IEC 13818-1 [33].

video_24hr_pic_present – This 1-bit Boolean flag when set to ‘1’, shall indicate that the video asset may include HEVC 24-hour pictures as defined in ISO/IEC 13818-1 [33]. When set to ‘0’, the flag shall indicate that the video asset shall not include any HEVC 24-hour picture as defined in ISO/IEC 13818-1 [33].

7.2.3.2.1.4 Picture Rate Information

Table 7.13 Bit Stream Syntax for Picture Rate Information

Syntax	No. of Bits	Format
pr_info(maxSubLayersMinus1) {		
for (i = 0; i <= maxSubLayersMinus1; i++) {		
picture_rate_code[i]	8	uimsbf
if(picture_rate_code[i] == 255) {		
average_picture_rate[i]	16	uimsbf
}		
}		
}		

picture_rate_code[i] – This 8-bit unsigned integer field shall provide information about the picture rate for the ith temporal sub-layer of this video asset. The picture_ rate_code[i] code indicates following values for picture rate for the ith temporal sub-layer: 0= unknown, 1 = 23.976 Hz, 2 = 24 Hz, 3 = 29.97 Hz, 4 = 30 Hz, 5 = 59.94 Hz, 6 = 60 Hz, 7 = 25 Hz, 8 = 50 Hz, 9 = 100 Hz, 10 = 120/1.001 Hz, 11 = 120 Hz, 12-254 = reserved, 255 = Other. When picture_rate_code[i] is equal to 255 the actual value of picture rate shall be indicated by the average_picture_rate[I] element.

average_picture_rate[i] – This 16-bit unsigned integer field shall indicate the average picture rate, in units of picture per 256 seconds, of the ith temporal sub-layer. The semantics of avg_pic_rate[0][i] defined in H.265 (10/2014) HEVC specification Section F.7.4.3.1.4 (VPS VUI Semantics) [38] shall apply.

The average_picture_rate[i] shall not have a value corresponding to any of the picture rate values: 23.976 Hz, 24 Hz, 29.97 Hz, 30 Hz, 59.94 Hz, 60 Hz, 25 Hz, 50 Hz, 100 Hz, 120/1.001 Hz, 120 Hz. In this case the picture_rate_code[i] shall be used to indicate the picture rate.

7.2.3.2.1.5 Bit Rate Information

Table 7.14 Bit Stream Syntax for Bit Rate Information

Syntax	No. of Bits	Format
br_info(maxSubLayersMinus1) {		
for (i = 0; i <= maxSubLayersMinus1; i++) {		
average_bitrate[i]	16	uimsbf
maximum_bitrate[i]	16	uimsbf
}		
}		

average_bitrate[i] – This 16-bit unsigned integer field shall indicate the average bit rate of the ith temporal sub-layer of this video asset in bits per second. The semantics of avg_bit_rate[0][i] defined in H.265 (10/2014) HEVC specification Section F.7.4.3.1.4 (VPS VUI Semantics) [38] shall apply.

maximum_bitrate[i] – This 16-bit unsigned integer field shall indicate the maximum bit rate of the ith temporal sub-layer in any one-second time window. The semantics of max_bit_rate[0][i] defined in H.265 (10/2014) HEVC specification Section F.7.4.3.1.4 (VPS VUI Semantics) [38] shall apply.

7.2.3.2.1.6 Color Information

Table 7.15 Bit Stream Syntax for Color Information

Syntax	No. of Bits	Format
color_info() {		
colour_primaries	8	uimsbf
transfer_characteristics	8	uimsbf
matrix_coeffs	8	uimsbf
if (colour_primaries>=9) {		
cg_compatibility	1	bslbf
reserved	7	'1111111'
}		
if (transfer_characteristics>=16) {		
eotf_info_present	1	bslbf
if (eotf_info_present) {		
eotf_info_len_minus1	15	uimsbf
eotf_info()	(eotf_info_len_minus1+1)*8	
}		
else {		
reserved	7	'1111111'
}		
}		
}		

colour_primaries, transfer_characteristics, matrix_coeffs elements respectively shall have the same semantics meaning as the elements with the same name in H.265 (10/2014) HEVC specification Section E.3.1 (VUI Parameter Semantics) [38].

cg_compatibility – This 1-bit Boolean flag shall indicate, when set to '1', that the video asset is coded to be compatible with Rec. ITU-R BT.709-5 [40] color gamut. When set to '0', the flag shall indicate that the video asset is not coded to be compatible with Rec. ITU-R BT.709-5 [40] color gamut.

eotf_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that the elements in eotf_info() structure are present. When set to '0', the flag shall indicate that the elements in eotf_info() structure are not present.

eotf_info_len_minus1 –This 15-bit unsigned integer plus 1 shall specify the length in bytes of the eotf_info() structure immediately following this field.

eotf_info() – Provides data as described in Section 7.2.3.2.1.7 and Table 7.16.

For each video asset, a video_stream_properties_descriptor() shall be signaled which includes the video asset id for the video asset in the asset_id_byte fields of the video_stream_properties_descriptor(). An mmt_atsc3_message() carrying a video_stream_properties_descriptor() shall be signaled for a video asset before delivering any MPU corresponding to the video asset. A video_stream_properties_descriptor() may be used to signal the properties of more than one video asset (by use of the syntax element number_of_assets and the associated for loop with asset_id_bytes field) to which the video_stream_properties_descriptor() applies.

7.2.3.2.1.7 Transfer Function Information

Table 7.16 Bit Stream Syntax for Transfer Function Information

Syntax	No. of Bits	Format
eotf_info() {		
num_SEIs_minus1	8	uimsbf
for (i = 0; i <= num_SEIs_minus1; i++) {		
SEI_NUT_length_minus1[i]	16	uimsbf
SEI_NUT_data[i]	8*(SEI_NUT_length_minus1[i]+1)	
}		
}		

num_SEIs_minus1 – This 8-bit unsigned integer value, plus 1, indicates the number of supplemental enhancement information message [38] NAL unit [38] data signaled in this eotf_info().

SEI_NUT_length_minus1[i] – This 16-bit unsigned integer value, plus 1, indicates the number of bytes of data in the SEI_NUT_data[i] field.

SEI_NUT_data[i] contains the data bytes for the supplemental information message NAL unit [38]. The nal_unit_type of NAL unit in SEI_NUT_data[i] shall be equal to 39. The payloadType value of SEI message in SEI_NUT_data[i] shall be equal to 137 or 144.

7.2.3.3 ATSC Staggercast Descriptor

In order to explicitly signal that an Asset is only suitable for Staggercast, the following shall be done in MMTP-specific signaling:

- Signal the Staggercast audio Asset in the list of Assets in the MP table of the service to which the Staggercast Asset belongs to;
- Deliver an ATSC_staggercast_descriptor() (see Table 7.17) using mmt_atsc3_message() that allows signaling the dependency between the Staggercast Asset and the main Asset protected by it thanks to the presence of the packet_id of the main Asset in the ATSC_staggercast_descriptor().

Table 7.17 Bit Stream Syntax for ATSC Staggercast Descriptor

Syntax	No. of Bits	Format
ATSC_staggercast_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
number_of_staggercast_assets	8	uimsbf
for (i=0;i<number_of_assets;i++) {		
staggercast_packet_id	16	uimsbf
main_asset_packet_id	16	uimsbf
}		
}		

descriptor_tag - This 16-bit unsigned integer shall have the value 0x0006, identifying this descriptor as being the ATSC_staggercast_descriptor().

descriptor_length – This 16-bit unsigned integer shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

number_of_assets – An 8-bit unsigned integer field that shall specify the number of Staggercast Assets described by this descriptor.

staggercast_packet_id – This 16-bit unsigned integer shall indicate the packet_id of a Staggercast Asset.

main_asset_packet_id – indicates which main Asset is protected by the Staggercast Asset. Its value shall be set to the value of the packet_id of the main Asset protected by the Staggercast Asset.

If an Asset is annotated with an ATSC_staggercast_descriptor(), the ATSC 3.0 client is expected to not select such Asset for regular playout. If the ATSC 3.0 receiver supports the Staggercast mechanism, it is expected to buffer both the main audio and the Staggercast audio in order to be able to switch down to the Staggercast audio, should main audio become unavailable.

7.2.3.4 Audio Stream Properties Descriptor

Properties of audio streams shall be signaled by the audio_stream_properties_descriptor(). It provides information about the audio streams associated with the asset IDs signaled in the descriptor.

The audio_stream_properties_descriptor() is embedded in the MMTP-specific signaling message mmt_atsc3_message() with an atsc3_message_content_type value of 0x0009 as defined in Table 7.6.

7.2.3.4.1 Syntax and Semantics

The syntax for the audio_stream_properties_descriptor() shall conform to Table 7.18. The semantics of the fields in the audio_stream_properties_descriptor() shall be as given immediately below the table.

Table 7.18 Bit Stream Syntax for the Audio Stream Properties Descriptor

Syntax	No. of Bits	Format
audio_stream_properties_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
number_of_assets	8	uimsbf
for (i=0; i<number_of_assets; i++) {		
asset_id_length	32	uimsbf
for (j=0; j<asset_id_length; j++) {		
asset_id_byte	8	uimsbf
}		
codec_code	4*8	uimsbf
num_presentations	8	uimsbf
multi_stream_info_present	1	bslbf
emergency_info_time_present	1	bslbf
reserved	6	‘111111’
for (j=0; j<num_presentations; j++) {		
presentation_id	8	uimbsf
interactivity_enabled	1	bslbf
profile_channel_config_present	1	bslbf
profile_long	1	bslbf
channel_config_long	1	bslbf
audio_rendering_info_present	1	bslbf
language_present	1	bslbf
accessibility_role_present	1	bslbf
label_present	1	bslbf
if (profile_channel_config_present) {		

if (**profile_long** == 1) {		
profile_level_indication	3*8	uimsbf
} else {		
profile_level_indication	8	uimsbf
}		
if (channel_config_long == 1) {		
audio_channel_config	3*8	uimsbf
} else {		
audio_channel_config	8	uimsbf
}		
}		
if (audio_rendering_info_present) {		
audio_rendering_indication	8	uimsbf
}		
if (language_present) {		
num_languages_minus1	8	uimbsf
for(k=0; k< num_languages_minus1+1; k++) {		
language_length	8	uimbsf
for (l=0; l< language_length; l++) {		
language_byte	8	uimbsf
}		
}		
}		
if (accessibility_role_present) {		
for(k=0; k< num_languages_minus1+1; k++) {		
accessibility	8	uimbsf
}		
role	8	uimsbf
}		
if (label_present) {		
label_length	8	uimsbf
for(k=0; k< label_length; k++) {		
label_data_byte	8	uimbsf
}		
}		
if (multi_stream_info_present) {		
presentation_aux_stream_info()		Table 7.26
}		
} /* end of for num_presentations loop*/		
if (multi_stream_info_present) {		
multi_stream_info()		Table 7.25
}		
if (emergency_info_time_present) {		
emergency_information_time_info()		Table 7.24
}		
}		
}		

descriptor_tag – A 16-bit unsigned integer field that shall have the value 0x0009, identifying this descriptor as the audio_stream_properties_descriptor().

descriptor_length – A 16-bit unsigned integer field that shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

number_of_assets – An 8-bit unsigned integer field that shall specify the number of audio assets described by this descriptor.

asset_id_length – An 32-bit unsigned integer field that shall specify the length in bytes of the audio asset id.

asset_id_byte – An 8-bit unsigned integer field that shall contain a byte of the audio asset id.

codec_code – A 32-bit unsigned integer field that shall specify a 4-character code for a codec. For AC-4 Audio [8], the value of the codecs element shall be created according to the syntax described in RFC6381 [29] and consists of the "fourCC" `'ac-4'`. For MPEG-H Audio [9], the value of these four characters shall be one of `'mhm1'` or `'mhm2'` with semantic meaning for these codes as specified in ISO/IEC 23008-3 [39] as amended.

multi_stream_info_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the elements in the multi_stream_info() structure are present and that the stream is part of a bundle of streams that together form an audio program. When set to '0', this flag shall indicate that the elements in the multi_stream_info() structure are not present and that the audio stream is a complete main stream that contains all audio components of all presentations.

emergency_info_time_present – A 1-bit Boolean flag that when set to '1' shall indicate that the elements in the emergency_information_time_info() structure are present. And that when set to '0' shall indicate that the elements in the emergency_information_time_info() structure are not present. The value of this field shall be equal to '0' when none of the Presentations in the audio asset contain an audio/aural representation of the emergency information.

num_presentations – An 8-bit unsigned integer field that shall indicate the number of Presentations that are available within the main stream and all auxiliary streams. The minimum number of num_presentations shall be '1' for the main stream. For auxiliary streams num_presentations shall have the value '0' so that for auxiliary streams no presentation information is present in the descriptor.

In the case of AC-4 Audio [8], this field shall contain the value of the n_presentations field from the ac4_dsi_v1 structure as specified in Annex E.6 of ETSI TS 103 190-2 [13].

In the case of MPEG-H Audio [9], this field contains a `'mae_numGroupPresets'` field as specified in ISO/IEC 23008-3 subclause 15.3 [39] as amended.

presentation_id – An 8-bit unsigned integer field that shall identify the ID of this Presentation. The first presentation in the loop shall have the lowest presentation_id and shall be the default Presentation.

In the case of AC-4 Audio [8] this field shall contain the value of the presentation_group_index in the ac4_presentation_v1_dsi associated with an AC-4 presentation as specified in Annex E.10 of ETSI TS 103 190-2 [13].

In the case of MPEG-H Audio [9], this field contains a `'mae_GroupPresetID'` field as specified in ISO/IEC 23008-3 subclause 15.3 [39] as amended.

interactivity_enabled – A 1-bit Boolean flag that shall indicate, when set to '1', that the audio presentation contains elements with associated metadata, which enable user interactivity. When set to '0', the flag shall indicate that no user interactivity of any kind is available. This flag may be used to determine the need for initializing a user interface for audio interactivity.

profile_channel_config_present – A 1-bit Boolean flag that shall indicate, when set to '1', that profile and channel configuration information for this Presentation is present. When set to '0', this flag shall indicate that no profile and channel configuration information for this Presentation is present.

The profile_channel_config_present flag shall be always set to '1' for the default presentation.

profile_long – A 1-bit Boolean flag that shall indicate, when set to '1', that profile level information for this Presentation is signaled using 3 bytes ("long format"). When set to '0', this flag shall indicate that profile level information for this Presentation is signaled using 1 byte ("short format"). In the case of AC-4 Audio [8] this field shall be set to '1'. In the case of MPEG-H Audio [9] this field shall be set to '0'.

channel_config_long – A 1-bit Boolean flag that shall indicate, when set to '1', that channel configuration information for this Presentation is signaled using 3 bytes ("long format"). When set to '0', this flag shall indicate that channel configuration information for this Presentation is signaled using 1 byte ("short format"). In the case of AC-4 Audio [8] this field shall be set to '1'. In the case of MPEG-H Audio [9] this field shall be set to '0'.

audio_rendering_info_present – A 1-bit Boolean flag that shall indicate, when set to '1', that additional audio rendering info for this Presentation is present. When set to '0', this flag shall indicate that no additional audio rendering info for this Presentation is present.

language_present – A 1-bit Boolean flag that shall indicate, when set to '1', that language information for this Presentation is present. When set to '0', this flag shall indicate that no language information for this Presentation is present.

accessibility_role_present – A 1-bit Boolean flag that shall indicate, when set to '1', that accessibility and role information for this Presentation is present. When set to '0', this flag shall indicate that no accessibility and role information for this Presentation is present.

label_present – A 1-bit Boolean flag that shall indicate, when set to '1', that a text label for this Presentation is present. When set to '0', this flag shall indicate that the text label for this Presentation is not present.

profile_level_indication – This field shall indicate the audio profile and level of the associated presentation, i.e. it indicates the complexity and decoder requirements.

If profile_long is set to '1', this field is a 3*8bit unsigned integer field. In the case of AC-4 Audio [8], this field shall contain the values of the bitstream_version, presentation_version, and mdcompat fields of the presentation from the ac4_dsi_v1 structure as specified in Annex E of ETSI TS 103 190-2 [13]. The syntax of the profile_level_indication field shall be as specified in Table 7.19.

Table 7.19 Syntax for AC-4 profile_level_indication()

Syntax	No. of Bits	Format
profile_level_indication() {		
bitstream_version	7	uimbsf
presentation_version	8	uimbsf
mdcompat	3	uimbsf
reserved	6	'111111'
}		

If profile_long is set to ‘0’, this field is an 8-bit unsigned integer field. In the case of MPEG-H Audio [9], this field shall contain an `'mpegh3daProfileLevelIndication'` field as specified in ISO/IEC 23008-3 subclause 5.3.2 [39] as amended.

audio_channel_config – This field shall indicate the audio channel configuration of the encoded audio stream.

If channel_config_long is set to ‘1’, this field is a 3*8bit unsigned integer field. In the case of AC-4 Audio [8], this field shall indicate the channel configuration and layout, and shall contain a six-digit hexadecimal representation of the 24-bit speaker group index bit field, which describes the channel assignment of the referenced AC-4 bit stream according to table A.27 in Annex A.3 of ETSI TS 103 190-2 [13]. A value of ‘0’ indicates that a single audio_channel_config value is not meaningful, because the encoded audio e.g. contains a combination of audio elements that are not channel-based, like audio objects that could be rendered to positions/coordinates independent from speaker configurations. Instead, the audio_rendering_indication field may be used to retrieve some meaningful information about the available rendering options.

If channel_config_long is set to ‘0’, this field is an 8-bit unsigned integer field. In the case of MPEG-H Audio [9], the syntax of the audio_channel_config field shall be as specified in Table 7.20.

Table 7.20 Syntax for MPEG-H audio_channel_config()

Syntax	No. of Bits	Format
audio_channel_config () {		
channel_configuration	6	uimbsf
reserved	2	’11’
}		

channel_configuration – This field is a 6-bit unsigned integer field that contains the `'ChannelConfiguration'` field as specified in ISO/IEC 23001-8 [36]. A value of ‘0’ indicates that a single channel_configuration value is not meaningful, because the encoded audio e.g. contains a combination of audio elements that are not channel-based, like audio objects that could be rendered to positions/coordinates independent from speaker configurations. Instead, the audio_rendering_indication field may be used to retrieve some meaningful information about the available rendering options.

audio_rendering_indication – An 8-bit unsigned integer field that may contain a hint for a preferred reproduction channel layout, if audio_channel_configuration is set to ‘0’ or channel_configuration is set to ‘0’. Values of audio_rendering_indication shall be as specified in Table 7.21.

Table 7.21 Meaning of audio_rendering_indication Values

audio_rendering_indication value	Meaning
0x00	There is no preference given for the reproduction channel layout.
0x01	The content is pre-rendered for consumption with headphones. In this case, the following field reference_channel_layout shall be set to '0'.
0x02	The content has only elements to be rendered in a plane, i.e. the preferred reproduction channel layout is a two-dimensional layout
0x03	the content has elements with heights, i.e. the preferred reproduction channel layout is a three-dimensional layout
0x04-0xFF	ATSC Reserved

num_languages_minus1 – An 8-bit unsigned integer field plus 1 that shall specify the number of languages that are available within this Presentation. When not present the value of num_languages_minus1 shall be inferred to be equal to 0.

language_length – An 8-bit unsigned integer field that shall specify the length in bytes of each language supported in the Presentation. The first language in the loop (k is equal to 0) shall be the primary language for the Presentation. The remaining language(s) in the loop (k is not equal to 0) shall indicate the additional language(s) available in the Presentation.

language_byte – An 8-bit unsigned integer field that shall contain a UTF-8 character of the k-th language of the Presentation. The language of the Presentation shall be given by a language tag as defined by IETF BCP 47 [32].

In the case of AC-4 Audio [8], the language indicated by this field should correspond to the information conveyed in the language_tag_bytes field of the ac4_substream_group_dsi structure (within the ac4_dsi_v1 structure) for the presentation, that is tagged as dialog or complete main. The ac4_substream_group_dsi structure is specified in Annex E.11 of ETSI TS 103 190-2 [13].

In the case of MPEG-H Audio [9], the language indicated by this field should correspond to the information conveyed in mae_contentLanguage of the default dialog element: the maeGroup which is marked as default in mae_switchGroupDefaultGroupID and is tagged in mae_contentKind as dialog. This information is carried in the AudioSceneInformation() of the MPEG-H Audio stream as specified in ISO/IEC 23008-3 [39].

accessibility – An 8-bit unsigned integer field that shall identify the accessibility support for each language in this Presentation. Accessibility support is signaled for each language in this Presentation. The k-th occurrence of accessibility element in this accessibility "for" loop shall indicate the accessibility information for k-th language in the language "for" loop above. Table 7.22 specifies the bit used to indicate if the Presentation contains support for a particular audio accessibility service. When the bit specified in Table 7.22 is set to '1' it indicates the Presentation contains the corresponding audio accessibility service. When the bit specified in Table 7.22 is set to '0' it indicates the Presentation does not contain the corresponding audio accessibility service.

Table 7.22 accessibility Bits

Bit	Audio Accessibility Service
0 (MSB)	For Visually Impaired (Video Description Service)
1	Dialog Enhancement enabled
2	Audio/Aural representation of emergency information
3 – 7	ATSC Reserved zero bits
Note: Bit 0, which indicates the presence of the audio accessibility service for visually impaired for the Presentation is the MSB of the accessibility field. For example, to indicate that the Presentation contains audio accessibility service for Visually Impaired as well as aural/audio representation of emergency information, the value of the accessibility field would be 0xA0 (the hexadecimal equivalent of the binary value 1010 0000). Reserved bits shall be set to zero. Note that this is different than the ATSC default value for reserved bits as stated in Section 3.2.1 of this document.	

Bit 1 set to ‘1’ indicates that the Presentation enables the ability for a receiver to change the relative level of dialog to enhance dialog intelligibility.

In the case of AC-4 Audio [8], the setting of the bits in the accessibility field should correspond to the information conveyed in the content_classifier field of the ac4_substream_group_dsi structures (within the ac4_dsi_v1 structure) as specified in Annex E.11 of ETSI TS 103 190-2 [13]. The mapping from AC-4 metadata fields should be done as follows:

- Bit 0 should be set to ‘1’ if the value of the content_classifier field of one of the ac4_substream_group_dsi structures of the presentation is set to ‘010’ (“Associated service: visually impaired (VI)”).
- Bit 1 should be set to ‘1’ if the content_classifier field of one or more ac4_substream_group_dsi structures are set to a value other than ‘001’ (“Main audio service: music and effects (ME)”).
- Bit 2 should be set to ‘1’ if the value of the content_classifier field of one of the ac4_substream_group_dsi structures of the presentation is set to ‘110’ (“Associated service: emergency (E)”).

In the case of MPEG-H Audio [9], the setting of the bits in the accessibility field should correspond to the mae_groupPresetKind value in the mae_GroupPresetDefinition() structure and the mae_contentKind values in the mae_ContentData() structures in the AudioSceneInformation() of the MPEG-H Audio stream as specified in ISO/IEC 23008-3 [39]. The mapping from the MPEG-H Audio meta-data fields should be done as follows:

- Bit 0 should be set to ‘1’, if the mae_contentKind value of at least one Audio Element is set to ‘9’ (“audio description/visually impaired”).
- Bit 1 should be set to ‘1’, if at least the dialog Audio Elements with a mae_contentKind value of ‘2’ (“dialogue”) have mae_allowGainInteractivity set to ‘1’ and mae_interactivityMaxGain set to a non-zero value in the corresponding mae_GroupDefinition() structure.
- Bit 2 should be set to ‘1’, if the mae_contentKind value of at least one Audio Element is set to ‘12’ (“emergency“).

role – An 8-bit unsigned integer field that shall indicate the role or service type of the Presentation. The role index values shall be as specified in Table 7.23. The role values shall correspond to the role scheme schemeIdUri="urn:mpeg:dash:role:2011" defined in ISO/IEC 23009-1:2014 MPEG DASH [55]. For a description of the role values, see Section 5.8.4.4 in ISO/IEC 23009-1:2014.

Table 7.23 Meaning of role Values

role index value	role value
0x00	main
0x01	alternate
0x02	supplementary
0x03	commentary
0x04	dub
0x05-0xFF	ATSC Reserved

label_length – An 8-bit unsigned integer field that shall specify the length in bytes of this Presentation text label.

label_data_byte – An 8-bit unsigned integer field that shall contain a UTF-8 character of the Presentation text label.

7.2.3.4.2 Emergency Information Time Information

Table 7.24 Bit Stream Syntax for Emergency Information Time Information

Syntax	No. of Bits	Format
emergency_information_time_info() {		
emergency_information_start_time_present	1	bslbf
emergency_information_end_time_present	1	bslbf
reserved	6	’11 1111’
If (emergency_information_start_time_present) {		
emergency_information_start_time	32	uimsbf
reserved	6	’11 1111’
emergency_information_start_time_ms	10	uimsbf
}		
If (emergency_information_end_time_present) {		
emergency_information_end_time	32	uimsbf
reserved	6	’11 1111’
emergency_information_end_time_ms	10	uimsbf
}		
}		

emergency_information_start_time_present – A 1-bit Boolean flag that shall indicate, when set to ‘1’, that the fields emergency_information_start_time and emergency_information_start_time_ms are present. When set to ‘0’, the fields emergency_information_start_time and emergency_information_start_time_ms shall not be present and the start time of the aural/audio representation of the emergency information shall be inferred to be equal to the presentation time of first media sample in presentation order in the audio asset.

emergency_information_end_time_present – A 1-bit Boolean flag that shall indicate, when set to ‘1’, that the fields emergency_information_end_time and emergency_information_end_time_ms are present. When set to ‘0’, the fields emergency_information_end_time and emergency_information_end_time_ms shall not be present and the end time of the aural/audio representation of the emergency information shall be inferred to be equal to the presentation time of the last media sample in presentation order in the audio asset.

emergency_information_start_time – A 32-bit unsigned integer that shall indicate the start time of the aural/audio representation of the emergency information, as the least-significant 32 bits of the count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).

emergency_information_start_time_ms – A 10-bit unsigned integer in the range 0 to 999 that shall indicate the milliseconds offset from the time indicated in emergency_information_start_time, such that the formula emergency_information_start_time + (emergency_information_start_time_ms/1000) yields the start time of the audio/aural emergency information to the nearest 1 millisecond.

emergency_information_end_time – A 32-bit unsigned integer that shall indicate the end time of the aural/audio representation of the emergency information, as the least-significant 32 bits of the count of the number of seconds since January 1, 1970 00:00:00, International Atomic Time (TAI).

emergency_information_end_time_ms – A 10-bit unsigned integer in the range 0 to 999 that shall indicate the milliseconds offset from the time indicated in emergency_information_end_time, such that the formula emergency_information_end_time + (emergency_information_end_time_ms/1000) yields the end time of the audio/aural emergency information to the nearest 1 millisecond.

7.2.3.4.3 Multi-Stream Information

If the multi-stream information is present, the audio program is delivered in more than one elementary stream. All streams that are part of a bundle of streams together form an audio program. The main stream shall always be delivered as an MMTP/MPU broadcast stream. There are two options to deliver auxiliary streams:

- within the MMTP/MPU broadcast stream using different asset_IDs; they are signaled within this audio_stream_properties_descriptor(), or
- on broadband using DASH delivery; details are signaled in an MPD that is delivered as described in Section 8.2.2. The bundle_id and the stream_id shall be used to link audio adaptation sets in the MPD as auxiliary streams to this audio program.

Table 7.25 Bit Stream Syntax for Multi-Stream Information

Syntax	No. of Bits	Format
multi_stream_info() {		
this_is_main_stream	1	bslbf
this_stream_id	7	uimsbf
reserved	1	‘1’
bundle_id	7	uimsbf
if (this_is_main_stream) {		
reserved	1	‘1’
num_auxiliary_streams	7	uimbsf
for(m=0; m< num_auxiliary_streams;m++) {		
delivery_method	1	bslbf
auxiliary_stream_id	7	uimbsf
}		
}		
}		

this_is_main_stream – A 1-bit Boolean flag that shall indicate, when set to ‘1’, that this stream contains a main stream that can be presented on its own, or that can be combined with

additional audio components from an auxiliary stream. When set to '0', this flag shall indicate that this stream contains an auxiliary stream.

this_stream_id – A 7-bit unsigned integer field that shall indicate the ID of this audio stream. This ID shall be unique within one bundle, i.e., for all streams that have the same bundle_id.

bundle_id – A 7-bit unsigned integer field that shall identify a unique ID for one bundle of audio streams. A bundle consists of exactly one main stream and one or more additional auxiliary streams that shall have the same bundle_id. The auxiliary streams contain additional audio components that can be combined with the main stream. The auxiliary streams can be delivered via broadband or broadcast. The main stream shall always be delivered via broadcast and signaled within this audio_stream_properties_descriptor() with this_is_main_stream set to '1'.

num_auxiliary_streams – A 7-bit unsigned integer field that shall indicate the number of auxiliary streams that are available on broadband or broadcast to be combined with the main stream.

delivery_method – A 1-bit Boolean flag that shall indicate, when set to '1', that the corresponding auxiliary stream is delivered using DASH on HTTP. Access information to locate the stream is given in the DASH MPD. When set to '0', this flag shall indicate that the corresponding auxiliary stream is delivered using MMTP/MPU, signaled within the same audio_stream_properties_descriptor() and located through the asset id signaled for this stream.

auxiliary_stream_id – A 7-bit unsigned integer field that shall identify the ID of the auxiliary stream. The ID of all auxiliary streams shall be unique within one bundle.

7.2.3.4.4 Presentation Aux-Stream Information

If the multi-stream information is present, the audio program is delivered in more than one elementary stream. This structure lists all IDs of auxiliary streams that carry audio components that are required for a specific presentation.

Table 7.26 Bit Stream Syntax for presentation_aux_stream Information

Syntax	No. of Bits	Format
presentation_aux_stream_info() {		
num_presentation_aux_streams	8	uimbsf
for(m=0; m< num_presentation_aux_streams;m++) {		
aux_stream_id	8	uimbsf
}		
}		

num_presentation_aux_streams – An 8-bit unsigned integer field that shall indicate the number of auxiliary streams that are required for this specific Presentation.

aux_stream_id – An 8-bit unsigned integer field that shall identify the ID of the auxiliary stream that is required for this specific Presentation.

7.2.3.5 Caption Asset Descriptor

The closed caption metadata associated with caption assets shall be signaled by using the caption_asset_descriptor().

7.2.3.5.1 Syntax and Semantics

The syntax for the caption_asset_descriptor() shall conform to Table 7.27. The semantics of the fields in the caption_asset_descriptor() shall be as given immediately below the table. As shown, additional bytes following the last defined field may be present.

Table 7.27 Bit Stream Syntax for caption_asset_descriptor()

Syntax	No. of Bits	Format
caption_asset_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
number_of_assets	8	uimsbf
for (i=0; i<number_of_assets; i++) {		
asset_id_length	32	uimsbf
for (j=0; j<asset_id_length; j++) {		
asset_id_byte	8	uimsbf
}		
language_length	8	uimsbf
for (j=0; j<language_length; j++) {		
language_byte	8	uimsbf
}		
role	4	bslbf
aspect_ratio	4	bslbf
easy_reader	1	bslbf
profile	2	bslbf
3d_support	1	bslbf
reserved	4	bslbf
}		
for (i=0; i<N; i++) {		
reserved	8	bslbf
}		
}		

descriptor_tag – This 16-bit unsigned integer shall have the value 0x0008 identifying this descriptor as being the caption_asset_descriptor().

descriptor_length – This 16-bit unsigned integer shall specify the length (in bytes) immediately following this field up to the end of this descriptor.

number_of_assets – An 8-bit unsigned integer field that shall specify the number of caption MMT Assets.

asset_id_length – An 32-bit unsigned integer field that shall specify the length in bytes of the caption MMT Asset ID. For each i in the range of 0 to (number_of_assets-1) inclusive, the cumulative sum of asset_id_length value and (language_length+7) shall be less than (2^16-1).

asset_id_byte – An 8-bit unsigned integer field that shall contain a byte of the caption MMT asset id.

language_length – An 8-bit unsigned integer field that shall specify the length in bytes of the language of the caption MMT Asset.

language_byte – An 8-bit unsigned integer field that shall contain a UTF-8 character of the language of the caption MMT Asset. The language of a caption MMT Asset shall be given by a language tag as defined by IETF BCP 47 [32].

role – A 4-bit field that shall specify the purpose of the closed caption text as given in Table 7.28.

Table 7.28 Code Values for role

role	Meaning
0x0	main
0x1	alternate
0x2	commentary
0x3~0xF	ATSC Reserved

aspect_ratio – A 4-bit field that shall specify the display aspect ratio used by the caption author and as given in Table 7.29.

Table 7.29 Code Values for aspect_ratio

aspect_ratio	Meaning
0x0	16:9
0x1	4:3
0x2	21:9
0x3~0xF	ATSC Reserved

profile – A 2-bit field that when set to '01' shall indicate image captions, and when set to '00' shall indicate text captions. Field values '10' and '11' shall be reserved for future ATSC use.

easy_reader – A 1-bit field that when set to '1' shall indicate an easy reader caption MMT Asset, and otherwise not. Provisions of easy_reader captions are specified in A/343 [60].

3d_support – A 1-bit field that when set to '1' shall indicate closed caption support for both 2D and 3D video, and otherwise support for 2D video only. Provisions of 2D and 3D captions are specified in A/343 [60].

7.3 Content Advisory Ratings in Service Signaling

Content advisory program metadata may refer to a rating system defined in a Rating Region Table, or it may refer to another rating system.

7.3.1 DASH Signaling of RRT-based Content Advisories

For RRT-based program ratings, the content advisory rating XML data as specified in the **`ContentAdvisoryRatings`** element defined in the ATSC Service Announcement specification A/332 [5] is represented in string format using the following rules. The content advisory rating string corresponding to a given rating region shall consist of three parts separated by a comma as a delimiter:

- The first part shall indicate the region identifier coded as an unsigned byte value as specified by **`ContentAdvisoryRatings.RegionIdentifier`** element. If the **`ContentAdvisoryRatings.RegionIdentifier`** element is not present, an empty first part shall be included for the region identifier. If not present, the value of **`RegionIdentifier`** shall be coded as '1'.
- The second part shall indicate the rating description text coded as a string, as specified by the **`ContentAdvisoryRatings.RatingDescription`** element, enclosed further inside single quotes. If the **`ContentAdvisoryRatings.RatingDescription`** element is not present, an empty second part shall be included.
- The third part shall consist of one or more tuples enclosed within curly braces ("{}"). Each tuple shall consist of a rating dimension separated by a space (0x20 UTF-8)

followed by a rating value. The rating dimension shall be coded as an unsigned byte, as specified by the **`RatingDimension`** element. The rating value shall be coded as a string, as specified by the **`RatingValueString`** element, enclosed further inside single quotes. If a `RatingDimension` element is not present, it shall be coded as '0'. The third part shall not be empty.

To specify content advisory information data for multiple rating regions, additional three-part strings (one for each region) shall be concatenated to create one string consisting of multiple concatenated three part strings. In this case, the third part of each content advisory information string except the last shall be followed by a comma (","). Thus the last character of the entire content advisory ratings string is a right curly brace ("}").

The content advisory information XML data as specified in each `ContentAdvisoryRatings` element may be transformed into content advisory ratings string using following XPath transformation:

```
concat(concat(//RegionIdentifier,',',concat(concat('''',//RatingDescr
   iption),'''')),',',normalize-space(string-join(//RatingDimVal /
concat("{",concat(RatingDimension ,' ''', RatingValueString, "'}")),'
                                  ')))
```

Alternatively, the content advisory information XML data as specified in each `ContentAdvisoryRatings` element may be transformed into a content advisory ratings string using following XSL transformation:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="text"/>
<xsl:variable name="squote">'</xsl:variable>
<xsl:template match="/">
  <xsl:value-of select="concat(//RegionIdentifier,',')" />
  <xsl:value-of select="concat(concat(concat($squote,//RatingDescription),$squote),',')" />
  <xsl:for-each select="//RatingDimVal">
    <xsl:value-of select="concat(concat('{', RatingDimension),' ''' />
        <xsl:value-of select="concat(concat(concat($squote,RatingValueString),$squote),'}')" />
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

Table 7.30 below illustrates some example cases.

Table 7.30 Example Content Advisory Rating Strings

Rating Region	Program Rating	RRT Reference		Content Advisory Rating String
1	TV-PG-D-L-S-V	Dimension	Level	`"1,'TV-PG D-L-S-V',{0 'TV-PG'}{1 'D'}{2 'L'}{3 'S'}{4 'V'}"`
		0	3	
		1	1	
		2	1	
		3	1	
		4	1	
2	Canadian English Language Rating: 14+	Dimension	Level	`"2,'13+(fr)/14+(en)',{0 '14+'}{1 '13+'}"`
		0	5	
	Canadian French Language Rating: 13+	1	3	

In the case that a single program is rated for both rating regions 1 and 2, for the example in Table 7.30, the string would be

```
"1,'TV-PG D-L-S-V', {0 'TV-PG'}{1 'D'}{2 'L'}{3 'S'}{4 'V'},
          2,'13+(fr)/14+(en)',{0 '14+'}{1 '13+'}"
```

The MPD signaling of ratings shall be as defined in DASH-IF [12].

7.3.2 MMTP-Specific Signaling of RRT-Based Content Advisories

When using MMTP, the content advisory rating shall be specified by the **`ContentAdvisoryRating`** element in the MMTP USBD as defined in Table 7.4. The format of this element is identical to the **`ContentAdvisoryRatings`** element specified in the Service fragment of the ATSC 3.0 Service Announcement specification A/332 [5].

7.3.3 MMTP-Specific Signaling of Non-RRT Content Advisories

When using MMTP, the non-RRT content advisory rating shall be specified by the **`OtherRatings`** element in the MMTP USBD as defined in Table 7.4. The format of this element is identical to the **`OtherRatings`** element specified in the Service fragment of the ATSC 3.0 Service Announcement specification A/332 [5].

8. DELIVERY AND AL-FEC

8.1 Broadcast Delivery

Two delivery methods for streaming broadcast content are specified in the sections that follow. A method based on ROUTE and ROUTE/DASH is specified in Section 8.1.1. A method based on MMTP/MPU is specified in Section 8.1.2.

Source data may be associated with repair data that is generated from the source data by applying an AL-FEC algorithm. The repair data is only useful to application-layer forward error correction by forward error correction-capable receivers.

8.1.1 ROUTE/DASH

The DASH-IF [12] profile specifies formats and methods that enable the delivery of streaming service(s) from standard HTTP servers to DASH Client(s). DASH specifies the description of a collection of DASH Media Segments and auxiliary metadata (all referenced by HTTP URLs) through a DASH Media Presentation Description (MPD).

The ROUTE protocol is designed to deliver, via broadcast, DASH-formatted streaming content, such as linear TV services, to a large receiver population. It is capable of delivering an arbitrary number of different types of media objects over one or more source flows on LCT channels. Repair flows may be additionally used for certain deployment scenarios, such as to mobile devices, in certain geographical areas, only for certain service, etc.

8.1.1.1 Streaming Services Delivery

When `MPD@minimumUpdatePeriod` is present, the receiver should expect MPD updates to be carried in the LCT channel which provide carousel delivery of Service Layer Signaling (SLS) metadata fragments. The objects delivered by LCT channel of the ROUTE protocol in Annex A shall be formatted according to the announcement in the MPD. The MPD and the described DASH Media Presentation should conform to the DASH-IF [12] live profile.

If it is not possible that the Extended FDT parameters are determined prior to the object delivery, the Entity Mode (see Annex A Section A.3.3.3) shall be used. In this case, the Extended FDT parameters (as entity-headers) are sent in-band with the delivery object (as the entity-body), in the form of a compound object.

In addition, use of the Entity Mode enables partial or chunked delivery in the same way as HTTP provides a means to reduce the sender delay and therefore possibly the end-to-end delay. If the File Mode (see Annex A Section A.3.3.1.1) is used, then the file may also be sent in a progressive manner using the regular ROUTE sending operation in order to reduce the sender delay.

In the File Mode, the file/object metadata as represented by the Extended FDT shall either be embedded within, or shall be referenced as a separate delivery object by, the signaling metadata. In the Entity Mode, the file/object metadata shall be carried by one or more `entity-header` fields associated with the `entity-body`. If the repair flow is used in conjunction with the source flow for streaming content delivery, it may use one of the Package modes as the delivery object format as defined in Section A.3.3.4 and A.3.3.5 of Annex A. Doing so enables more robust Application Layer FEC recovery, by applying FEC protection across a collection of delivery objects for enhanced time diversity and constant QoS.

The `EXT_TIME` LCT extension header defined in RFC 5651 [26] shall be used to provide Sender Current Time (SCT), (`SCT-High` and `SCT-Low` are both set). For byte range delivery (MDE mode), the timing information optionally carried shall be the UTC time of the completion of the emission of the entire MDE data block and all related encapsulation in the broadcast emission necessary to decode it. For object delivery, the timing information optionally carried shall be the current UTC time at the sender when the last of byte of an object (last LCT packet) is emitted/radiated. This labeling shall be based on the same time base as is used to deliver time in the PHY layer.

When the first byte of an MDE data block with Random Access Point (RAP) is emitted/radiated, the LCT extension header of the first LCT packet shall include `EXT_ROUTE_PRESENTATION_TIME`, which shall be the only condition whereby this header is present. The value of `EXT_ROUTE_PRESENTATION_TIME` (see Annex A A.3.7.1) shall be set such that the delay (`EXT_ROUTE_PRESENTATION_TIME` - SCT) is the duration in time that the ROUTE receiver must hold this MDE data block with RAP prior to delivery to the ROUTE output to achieve stall-free playback. This delay time can be derived via characterization of the equipment generating the broadcast emission. The delay value is specific to a particular configuration of the scheduler and encoder, e.g. GOP duration, sequence of RAPs etc. Delivery of data blocks in MDE mode is by

definition in-order delivery, so any subsequent data blocks are held until after such time as the leading MDE data block with RAP is delivered to the ROUTE output.

8.1.1.2 Locally-Cached Content Delivery

This section specifies file-based delivery of locally-cached service content, and service metadata such as SLS fragments or the ESG.

File contents comprising discrete (i.e. untimed) media are downloaded by the receiver either in response to user selection, or acquired without need for user intervention. The source flow, when delivering file content, should use the File Mode as defined in Annex A as the delivery object format. The file metadata, or equivalently, the Extended FDT, is assumed to be known, available, and delivered to the receiver before the scheduled delivery of the file. The Extended FDT is delivered in advance by the S-TSID metadata fragment as defined in Table 7.2. The source flow for the subsequently delivered file shall provide a reference to the LCT channel and file URI.

`Content-Location` URIs referencing broadcast-delivered resources shall be relative URIs that do not resolve to any directory location above or outside the "base URI," where "base URI" is as defined in Section 5.1 of RFC 3986 [19]. Accordingly, URIs referencing broadcast-delivered resources cannot be absolute path references (ones beginning with "/"), cannot begin with "..", and always begin with a filename or directory path followed by the filename.

Receivers are expected to use base URIs of their choosing when converting the relative URIs in `Content-Location` and **`FDT-Instance`**`@fileTemplate` to absolute URLs. For application files (see A/337 [7] and A/344 [45]), the algorithm receivers use to establish the base URI involves Application Context ID (in the Extended FDT Instance as **`File`**`@appContextIdList`), as specified in A/344 [45].

From the application transport and IP delivery perspective, service metadata such as SLS or ESG fragments are no different from locally-cached files which belong to an locally-cached content item of an ATSC 3.0 application service. Service metadata delivery by the ROUTE protocol operates the same way as described above for files of locally-cached service contents.

8.1.1.3 Synchronization and Time

ROUTE/DASH requires accurate wall clock for synchronization. When a receiving device is connected to ATSC 3.0 broadcast reception, it is expected to utilize UTC as established via the physical layer for wall clock.

Network servers for both broadcast and broadband components of a Service shall synchronize to a common wall clock (UTC) source. GPS or similar accuracy and stability shall be provided.

The broadcast-established wall clock is expected to be utilized for both broadband and broadcast served media components at the receiver at all times. Wall clock (UTC) established via broadband connection is expected to be utilized at the receiver in the absence of the broadcast wall clock source.

Several use cases of receiver presentation time are supported for **`MPD`**`@type='dynamic'`. These are defined as: Earliest MDE ROUTE presentation time, Earliest Segment level DASH Media Presentation time, and Synchronized Segment level DASH Media Presentation timeline. The receiver is expected to calculate the offset of presentation timeline relative to receiver wall clock for two use cases as follows:

1) Earliest MDE ROUTE presentation time: MDE data block with T-RAP ROUTE reception time plus (`EXT_ROUTE_PRESENTATION_TIME -` SCT). Initial media request based on adjusted segment availability start time per Section 5.3.9.5.3 of [55][7].
2) Segment level DASH Media Presentation time: Segment request based on DASH Segment "availability start time" per the DASH-IF [12] profile in which the receiver calculates the presentation timeline based on UTC timing given in the MPD.

Note that DASH provides a mode supporting synchronized playback among different receivers using a parameter called `MPD@suggestedPresentationDelay`. Broadcast emission of `MPD@suggestedPresentationDelay` is optional. This mode is not described here as there is no ATSC 3.0 requirement for synchronized playback across multiple devices, and this mode causes an additional delay in channel change times.

8.1.1.4 ROUTE/DASH System Model

Figure 8.1 below depicts the ROUTE/DASH System Model. Definitions of terms are given below the figure after a discussion of the distinctions of the ROUTE/DASH System Model as compared to MPEG-2 Systems [33]. Note that there is a single unified model for all types of traffic i.e. there is no required distinction between video, audio, data and system control. Figure 8.1 depicts real time services. For locally-cached and system control services, ROUTE is implemented as depicted by the left hand section of Figure 8.1 with the object(s) terminating in a memory location visible to the appropriate application. In Figure 8.1, the appropriate memory location for media objects is the ROUTE Output Buffer, as shown below.

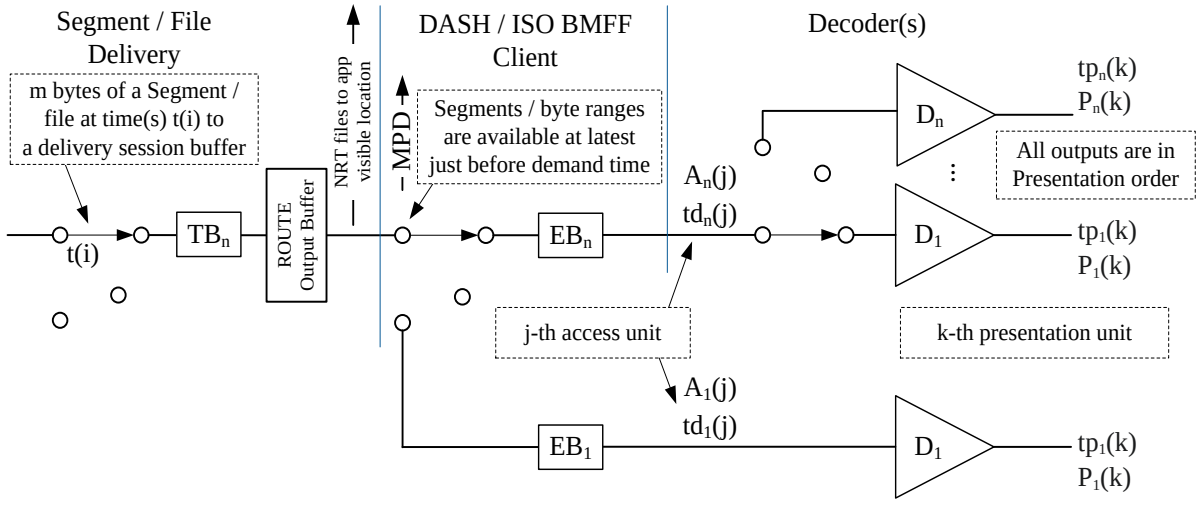


Figure 8.1 ROUTE/DASH system model.

There are a number of distinctions for the ROUTE/DASH system relative to the MPEG-2 Systems [33] model. The Data Delivery Events (DDEs) at specific t(i)s are not individual bytes, but rather the result of the delivery of a block of data by the PHY/MAC at a specific time i.e. a discrete data size in bytes at a discrete delivery time.

[7] Adjusted segment availability start time is computed by subtracting the value of `@availabilityTimeOffset` from the Segment availability start time.

A TB_n buffer is for a specific ROUTE session, rather than an elementary stream. A ROUTE session may deliver multiple objects and related AL-FEC. The specific TB_n buffer applies to all the objects and AL-FEC as encapsulated by IP/UDP/ROUTE. The size of TB_n is derived via the attribute `@minBuffSize` as described in the ROUTE specification, Table A.3.1. The size of TB_n is the sum of the `minBuffSize` values for each of the constituent LCT channels.

All events in a ROUTE/DASH model are discrete time based, i.e. "m" bytes input to or "n" bytes output from the buffer at a specific time. Since all events are discrete time for ROUTE/DASH, there are no leakage rates specified.

The delivered objects for media services are very briefly buffered in the ROUTE Output Buffer before they are consumed by the DASH Client. The minimum size of this buffer is slightly smaller than the corresponding TB_n, as the data in TB_n is wrapped in IP/UDP/ROUTE and may include AL-FEC packets. The output objects/files are of course both de-encapsulated and decoded.

The EB_n buffer is defined in MPEG-2 Systems [33] as an elementary stream buffer. This buffer, as applied to ROUTE/DASH, is associated with the DASH Segment handler, which holds data while it waits to be parsed by the decoders. Given that there may exist multiple object/file streams in an LCT channel, there may be n total EB_n (s) associated with a given LCT channel. There may also be "i" media types within a single DASH Segment stream and hence the multiple decoders D_n are each connected to a single instance of EB_n. The required size of EB_n is included as part of the definition of an appropriate DASH profile.

The task of scheduling media to the decoder(s) in the receiver is exclusively the responsibility of the DASH Segment handler (within the DASH Client). The scheduling of the objects/files to the DASH Segment handler is part of the DASH Client function. This described series of steps result in the DASH Segment handler receiving Media Delivery Events or object(s) that makes contextual and temporal sense and allows it to deliver samples (media frames) to the decoders, such that the DASH Media Presentation timeline is met.

Unstated so far, but implicit, is that any given LCT channel carries a single type of continuous content, and that a Service comprised of one or more types of content is transported within a collection of separate LCT channels, which belong to one or more ROUTE sessions. It is also significant that the ROUTE/DASH System Model supports continuous streaming with live ad insertions, which may impose some specific constraints with respect to behaviors at DASH Period boundaries.

The operation of the ROUTE/DASH system is defined such that none of the constituent buffers (TB_n, EB_n, or the ROUTE Output Buffer) are allowed to overflow, nor are the decoder(s) allowed to stall due to lack of input media. Each buffer starts at zero and can likely go to zero briefly during operation.

t(i) indicates the time in UTC at which m bytes of the transport stream i.e. a Data Delivery Event enters the target ROUTE transport buffer (exits top of the PHY/MAC layers). The value t(0) is a specific time in UTC, which is known by the physical layer scheduler.

TB_n is the transport buffer for the current LCT channel(s) in the ROUTE session, including all source object(s) and all related AL-FEC, as delivered in IP/UDP/ROUTE packets.

EB_n is the buffer in the DASH Client that holds the data received from the ROUTE Output Buffer, until the media is streamed to decoder(s) D1-Dn according to ISO BMFF semantics.

$A_n(j)$ is the jth access unit in a ROUTE delivery per decoder. $A_n(j)$ is indexed in decoding order.

$td_n(j)$ is the decoding time in UTC of the jth decoded media frame

D_n is the decoder for a specific media type per Service definition.

$P_n(k)$ is the kth presentation unit for a specific content type. $P_n(k)$ results from decoding $A_n(j)$. $P_n(k)$ is indexed in presentation order.

$tp_n(k)$ is the presentation time in UTC of the kth presentation unit.

8.1.1.5 ROUTE System Buffer Model

A valid broadcast emission shall have physical layer resources allocated in such a manner as to ensure that defined constraints on the maximum sizes of TB_n, ROUTE Output Buffer and EB_n (see Figure 8.1) are not exceeded. A valid broadcast emission shall have physical layer resources allocated in such a manner that buffers in devices that comply with an agreed-upon set of receiver characteristics, e.g. as determined by a standards organization such as CTA, will not overflow or underflow (cause a stall during playback).

The total transport buffer size of the TB_n required for a specific ROUTE session shall be defined by adding together the size of the buffer requirements for each LCT channel comprising the ROUTE session. The parameter `@minBuffSize` specifies the minimum buffer size for each LCT channel. A service may be comprised of multiple ROUTE sessions.

8.1.1.5.1 Data Delivery Event (DDE)

A Data Delivery Event (DDE) occurs when a block-based MAC/PHY delivers relevant contents of a specific physical layer block to a specific ROUTE session at a specific time. Each DDE has a specific time slot at the physical layer and its reception time at the receiver is known by the physical layer scheduler.

8.1.1.5.2 Media Delivery Event (MDE)

A Media Delivery Event (MDE) is the arrival of a collection of bytes that is meaningful to the upper layers of the stack for example the media player and decoder(s). The broadcast emission shall be constructed such that the delivery deadlines for MDE data blocks are met.

Figure 8.2 below depicts the concept of an MDE data block starting with a Random Access Point (RAP), which becomes a T-MDE starting with a T-RAP once encapsulated in IP/UDP/ROUTE. In the figure, "Meta" represents the collective set of metadata boxes located at the start of a DASH Media Segment, such as styp and sidx. "IS" stands for "Initialization Segment" and contains initializing data that is necessary to present the media streams encapsulated in DASH Media Segments. Figure 8.3 illustrates a "meaningful to upper layer" grouping of video frames. A key aspect is that it has a "required delivery time" to the DASH Client at the interface to ROUTE.

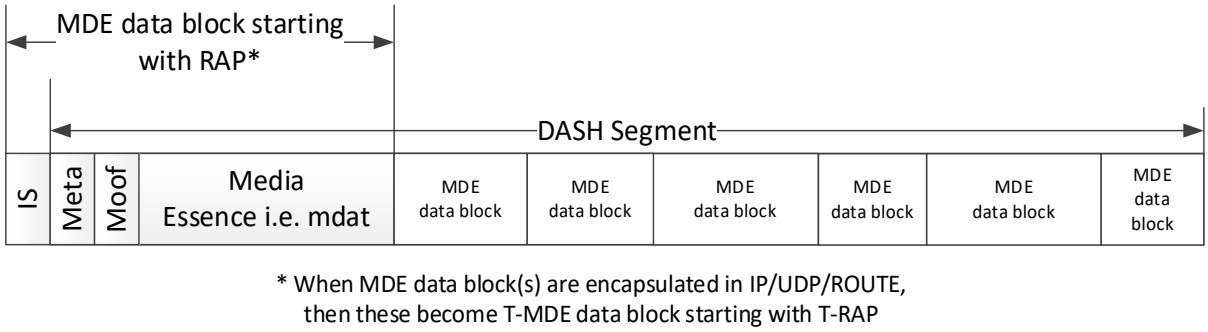


Figure 8.2 Concept of an MDE data block starting with a RAP.

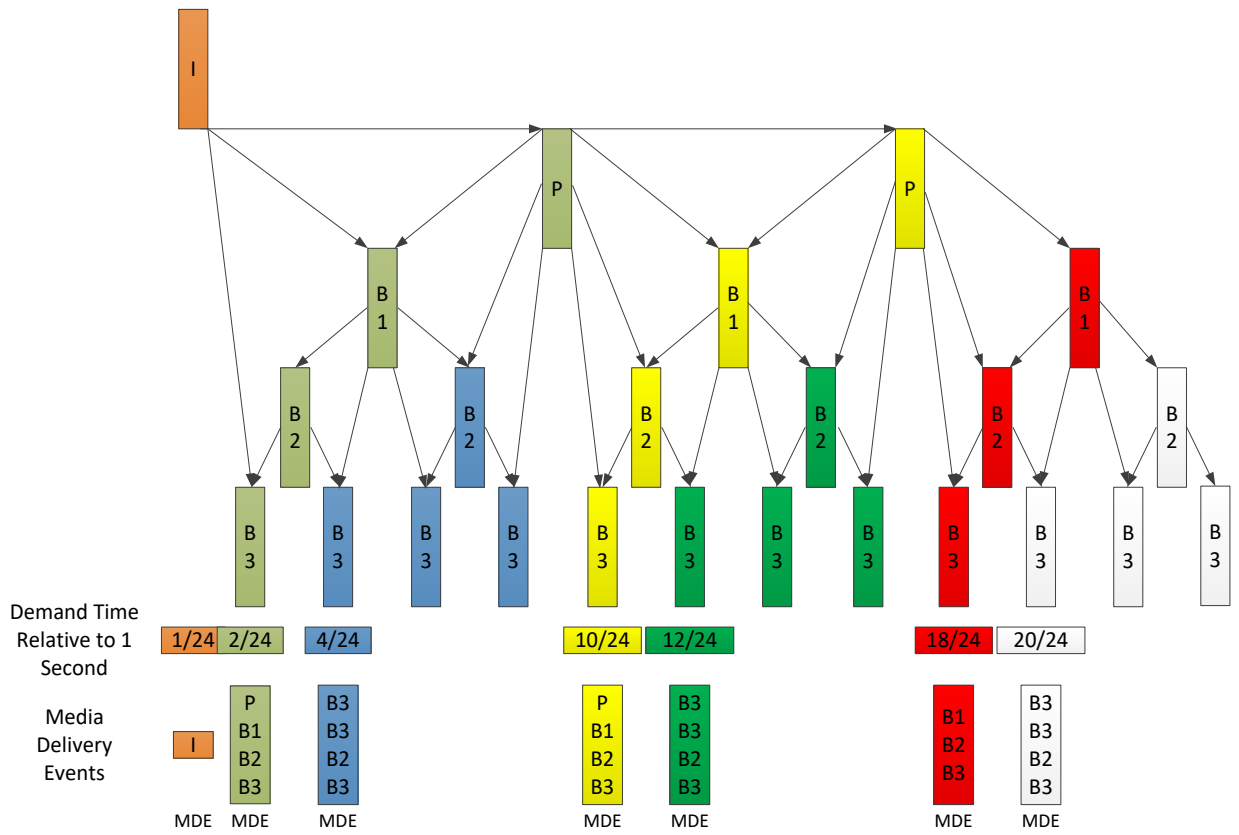
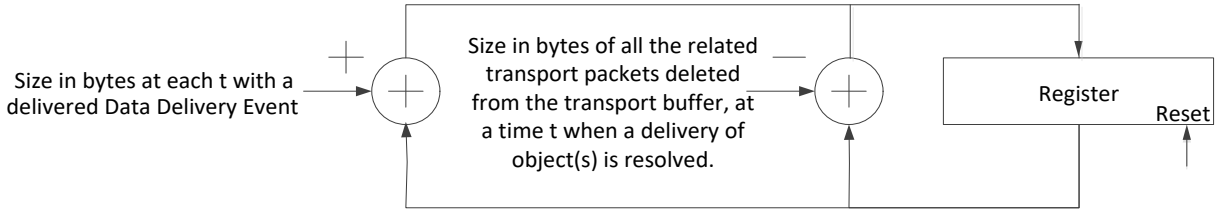


Figure 8.3 Illustration of an MDE data block at the video level.

8.1.1.5.3 ROUTE Transport Buffer Model

The transport buffer fullness is completely described by the ROUTE transport buffer model as shown in Figure 8.4 below. The value in the register on the right hand side of the figure denotes transport buffer fullness in bytes.



Notes:
- The register value is the buffer model fullness in bytes for time t at receiver
- Service start clears the register, a T-MDE with a T-RAP starts Service
- The buffer model updates at every t that transport data will be delivered to or removed from the transport buffer

Figure 8.4 ROUTE transport buffer model.

A notable aspect of the ROUTE/DASH System Model is the lack of a physical layer buffer. The ROUTE/DASH System Model has no object delivery event, i.e. a complete file required to play the T-MDE data block starting with T-RAP, as the MDE data block with RAP is delivered to the DASH Client as soon as allowed, rather than waiting for a complete object.

All the transport data associated with a ROUTE delivery of an MDE data block is retained in the transport buffer until the outcome of that specific ROUTE Segment delivery(s) are resolved. Resolution in this context means that the specific ROUTE delivery of the object(s) succeeded in whole or part, all the related MDE data block deliveries are complete, and all the results have been either posted up to the output buffer or have been abandoned. Upon resolution, all the related packets in the related transport buffer(s) are deleted. If there is FEC as part of the ROUTE delivery, the "related packets" include all those packets in buffers for all the related LCT channels. If an object is exclusively related to a single LCT channel, then the associated per-LCT transport buffer may be managed independently. In all cases the last object/MDE data block in a ROUTE delivery to be resolved and delivered determines when all the related IP/UDP/ROUTE packets get deleted from the associated LCT buffers.

8.1.1.6 Application of AL-FEC (Informative)

The availability of AL-FEC in ROUTE is crucial to a number of important applications, for example:

- Large NRT file delivery
- DVR applications
- Enhanced robustness for collection of smaller objects; e.g., a web page

The application of AL-FEC is of marginal benefit for linear streaming service content that have object physical layer durations in the range of the soft-decision FEC Frame in the physical layer, e.g. on the order of 250 msec. AL-FEC is applicable to ROUTE applications for both QoS and efficiency improvements. Users may tolerate up to 1% lost time for streaming video[8], i.e., a 99% success rate on a per-second basis, however they would prefer that their DVR recording be error-free. Assuming a live streaming video service bit rate of 4 Mbps and with a FEC Frame of 16 kilobits requires a successful block reception probability of ~$(1- 0.99^{(1/250)})$, i.e., a per-FEC Frame error rate of ~ 4e-5 is required to achieve 1% lost time. For this FEC Frame loss rate, the probability of a successful recording of an hour-long show with no lost seconds is ~ 2e-16 without AL-FEC being applied. The addition of just 5% AL-FEC across 30-second blocks increases the probability of success for a one-hour recording to ~99.99988%

Another application of AL-FEC is the broadcast delivery of app-based services, i.e., file-based delivery of application specific content objects (downloaded songs, video clips, interactivity related media assets, etc.), and service metadata such as the ESG. For NRT content delivery over ROUTE, the source flow, when delivering file content, should use the File Mode as defined by ROUTE. As an example, delivery of a 60-second video clip may achieve ~50% success of reception at a service quality that nominally supports video without AL-FEC. Under error conditions of the previous example, a fraction of a percent of AL-FEC assures complete delivery of the 60-second video clip.

Details on the implementation of AL-FEC in the ROUTE/DASH protocol can be found in Section A.4 of Annex A.

[8] One percent lost time is a more stringent requirement than ESR 5 as described in ITU-R Document 6E/64-E [56].

8.1.2 MMTP/MPU

MMTP is a protocol designed to deliver MPUs as specified in Clause 9 of ISO/IEC 23008-1 [37]. MMTP provides a number of useful features for real-time streaming delivery of MPUs via a unidirectional delivery network such as:

- The media-aware packetization of MPUs
- The multiplexing of various media components into a single MMTP session
- The removal of jitter introduced by the delivery network at the receiver within the constraints set by the sender
- The management of the buffer fullness of the receiver by the server to avoid any buffer underflow or overflow and the fragmentation/aggregation of payload data
- The detection of missing packets during delivery

For MMTP/MPU services, streaming contents shall be encapsulated into MPUs and delivered by MMTP. An MPU is an ISO BMFF formatted file compliant to the '`mpuf`' brand as specified in Clause 7 of ISO/IEC 23008-1 [37]. Restrictions applied to the '`mpuf`' brand enable efficient streaming delivery of MPUs. For example, an MPU is self-contained, i.e. initialization information and metadata required to fully decode the media data in each MPU is carried in the MPU. In addition, each MPU contains a globally unique ID of media components called the Asset ID and a sequence number to enable unique identification of each MPU regardless of the delivery mechanism. Details on the implementation of MMTP/MPU can be found in ISO/IEC TR 23008-13 [54].

8.1.2.1 Broadcast Streaming Delivery of MPUs

8.1.2.1.1 Mapping Between an ATSC 3.0 Service and MMT Packages

Each content component is considered as an MMT Asset having a unique identifier as defined in ISO/IEC 23008-1 [37]. Each MMT Asset is a collection of one or more MPUs with the same Asset ID which is provided in the ‘`mmpu`’ box as specified in Clause 7 of ISO/IEC 23008-1 [37]. MPUs associated with the same Asset ID do not overlap in presentation time. An MMT Package is a collection of one or more Assets, and an ATSC 3.0 Service delivered by MMTP shall be composed of one or more MMT Packages, where MMT Packages do not overlap in presentation time.

Multiple Assets may be delivered over a single MMTP session. Each Asset shall be associated with a packet_id which is unique within the scope of the MMTP session. This enables efficient filtering of MMTP packets carrying a specific Asset. The mapping information between MMT Packages and MMTP sessions shall be delivered to the receiver by MPT messages as specified in subclause 10.3.4 of ISO/IEC 23008-1 [37].

Figure 8.5 and Figure 8.6 show examples of the mapping between a Package and MMTP sessions. In these figures, the MMT Package has three assets: Asset A, Asset B and Asset C. Although all the MMT Signaling messages required to consume and deliver the MMT Package are delivered to the receiver, only a single MPT message is shown for simplicity. In Figure 8.5, all the Assets of the MMT Package and its associated MPT message are multiplexed into a single MMTP session. This configuration provides the simplest way to deliver a MMT Package. In Figure 8.6, a MMT Package is delivered over two MMTP sessions.

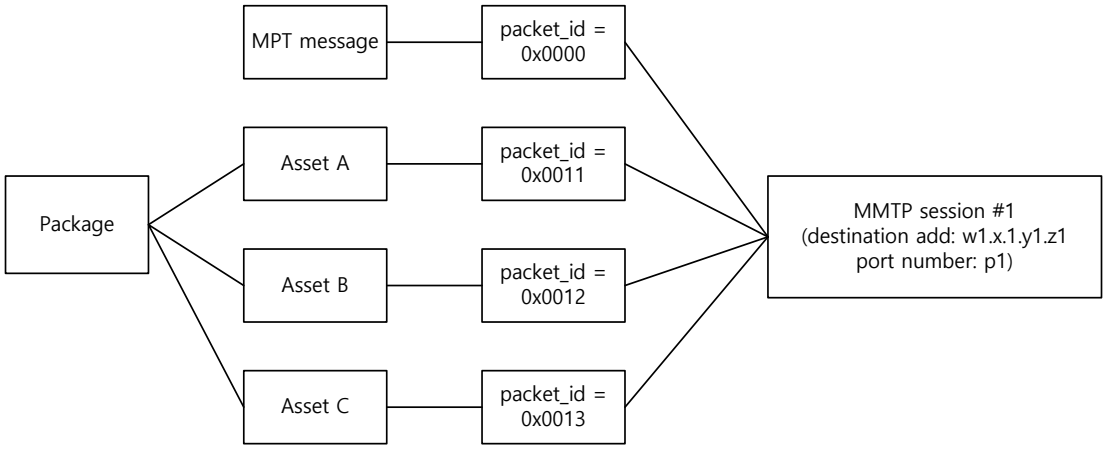


Figure 8.5 An MMT Package delivered over a single MMTP packet flow.

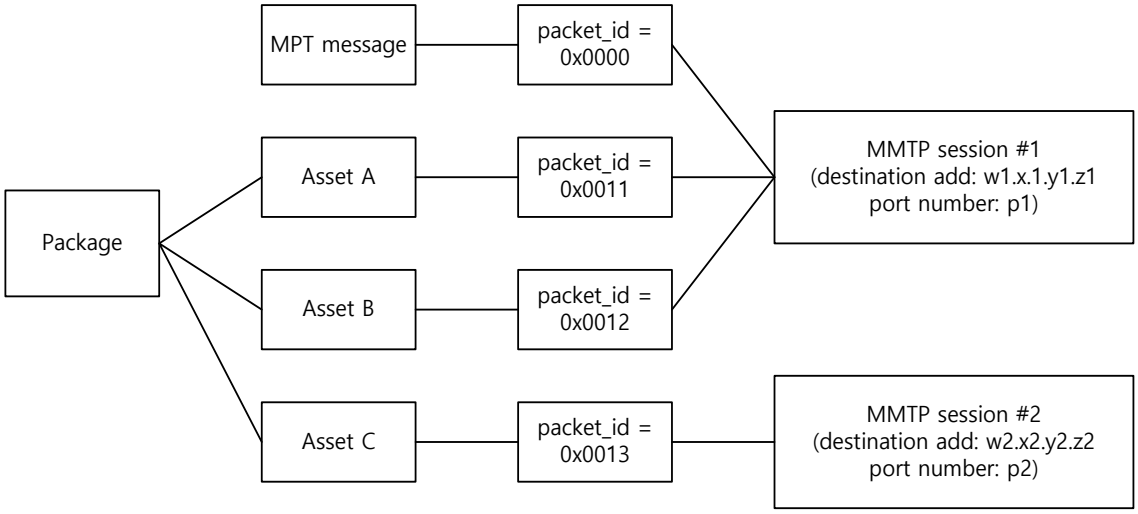


Figure 8.6 An MMT Package delivered over two MMTP packet flows.

8.1.2.1.2 Constraints on MPU

MPUs as specified in Clause 7 of ISO/IEC 23008-1 [37] shall be used as the encapsulation format for broadcast streaming media delivery via MMTP with the following additional constraint:

- The `'elst'` box shall not be used in an MPU.

8.1.2.1.3 Constraints on MMTP

MMTP as specified in Clause 9 of ISO/IEC23008-1 [37] shall be used to deliver MPUs. The following constraints shall be applied to MMTP:

- The value of the version field of MMTP packets shall be `'01'`.
- The value of the packet_id field of MMTP packets shall be between 0x0010 and 0x1FFE for easy conversion of an MMTP stream to an MPEG-2 TS. The value of the packet_id field of MMTP packets for each content component can be randomly assigned, but the value of the packet_id field of MMTP packets carrying two different components shall not be the same in a single MMTP session.

- The value of '0x01' for the type field of an MMTP packet header shall not be used, i.e. the Generic File Delivery (GFD) mode specified in subclauses 9.3.3 and 9.4.3 of [37] shall not be used.
- The value of the timestamp field of an MMTP packet shall represent the UTC time when the first byte of the MMTP packet is passed to the UDP layer and shall be formatted in the "short format" as specified in Clause 6 of RFC 5905, NTP version 4 [23].
- The value of the RAP_flag of MMTP packets shall be set to 0 when the value of the FT field is equal to 0.
- The Hypothetical Receiver Buffer Model (HRBM) specified in Clause 11 of ISO/IEC 23008-1 [37] shall be applied to the MMTP stream with the following constraints:
  - An independent set of HRBM buffers shall be used for each MMT Asset, i.e. MMTP packets with a different value of the packet_id field shall not be delivered to the same HRBM buffer.
  - A decoder buffer configured according to the type of MMT Asset shall be connected to each MMTP De-capsulation Buffer of the HRBM as shown in Figure 8.7.

    The value of max_buffer_size field of the HRBM message shall be obtained by the product of the maximum jitter this buffer can handle and the maximum bitrate of the MMTP packet stream transmitted during any 1-second period, and shall not be greater than

    $$5 \text{ seconds} * (1.2 * R_{max}).$$

    The value of R_{max} shall be determined as follows:
    - For video Assets, the value of R_{max} shall be given in units of bytes/s and obtained by converting the maximum bit rate given in bits/s specified in Annex A of ISO/IEC 23008-2 [38] for the given video Profile, Level and Tier to bytes/s. Any fractional value as the result of the conversion from bits/s to bytes/s shall be rounded up to the next larger integer.
    - For audio Assets, the value of R_{max} is given by 1.2 Mbps.
    - For closed-caption Assets, the value of R_{max} is 400 kbps.
  - MMTP streams shall be constructed to be compliant with the following client operation:
    - Each MMTP packet with the value of the type field equal to 0x00 shall be available at an MMTP De-capsulation Buffer at time `ts+Δ`, where `ts` is the value of the timestamp field of the MMTP packet and Δ is the fixed_end_to_end_delay signaled in milliseconds by an HRBM message.
    - At the time when an MMTP packet is available, i.e., when the UTC is `ts+Δ`, the header of the MMTP packet and the payload of the MMTP packet are immediately processed and the payload of the packet is copied to the MMTP De-capsulation Buffer.
    - For video and audio Assets, the media data available in an MMTP De-capsulation Buffer is constantly being delivered to the associated media decoder buffer at the average bit-rate of the associated MMT Asset. The average bit-rate of an Asset can be signaled by MMT ADC message specified in subclause 10.4.10 of [37].

For other Assets, the media data available in MMTP De-capsulation Buffer is delivered to the associated media buffer as soon as they are available

- The value of max_decapsulation_buffer_size field of the HRBM Removal message shall be obtained by the product of the maximum duration an MMTP packet can stay in this buffer and maximum bit rate of the MMTP packet stream transmitted during any 1-second period, and shall not be greater than

$$1 \text{ second} * (1.2 * R_{max}).$$

The value of R_{max} shall be determined as follows:

- For video Assets, R_{max} shall be given in units of bytes/s and be obtained by converting the maximum bit rate in bits/s specified in Table A.5 of ISO/IEC 23008-2 [38] for the given video Profile, Level and Tier to bytes/s. Any fractional value as the result of the conversion from bits/s to bytes/s shall be rounded up to the next larger integer.
- For audio Assets, the value of R_{max} is given as 1.2 Mbps.
- For closed-caption Assets, the value of R_{max} is given as 400 kbps.

- The MMTP Decapsulation Buffer shall not overflow.
- No MMTP packets shall stay longer than 1 second in the MMTP De-capsulation Buffer.

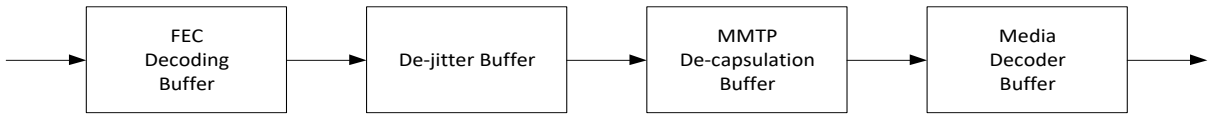


Figure 8.7 Receiver buffer model consuming MMTP sessions.

8.1.2.2 Packetization of MPU

ISO BMFF formatted files with the ‘`mpuf`’ brand are delivered by MMTP as MPUs. An MPU is delivered by MMTP in a media-aware packetized manner similar to an MPEG-2 TS [33] delivery. The media-aware packetization of MPUs into MMTP packets allows interleaving of packets thereby reducing latency by the receiver during initial acquisition of service while meeting bandwidth constraints of the delivery network. In addition, media-aware packetization allows a receiver to efficiently handle delivery errors. When an MPU is delivered by MMTP, the `FT` (MPU Fragment Type) field in the MMTP payload header shall be used to identify the type of MPU fragment as specified in subclause 9.3.2.2 of ISO/IEC23008-1 [37]. As an MPU is composed of both metadata and media data, it is packetized into two types of MMTP packets, packets with metadata and packets with media data.

MMTP packets with metadata in which the value of the `FT` field is set to 0 or 1 shall carry various boxes of an MPU, except media samples or sub-samples. The header of the ‘`mdat`’ box shall be carried by a MMTP packet with metadata.

MMTP packets with media data in which the value of the `FT` field is set to 2 shall carry samples or sub-samples of media data from the ‘`mdat`’ box. Each MMTP packet with media data has an explicit indication of the boundaries of the media samples or sub-samples. In addition, MMTP packets with media data carry the minimum information about media data needed to recover the

association between the media data and the metadata, such as a movie fragment[9] sequence number and a sample number.

Figure 8.8 shows one example of the media-aware packetization of an MPU with HEVC coded video data. In this example, each Coded Video Sequence (CVS) contains a single IDR picture as its first picture. Then each CVS of the video data is mapped to a single movie fragment. In Figure 8.8 one MPU has more than one movie fragment. When an MPU is fragmented, an '`ftyp`' box, an `'mmpu'` box and a `'moov'` box and other boxes applied to the whole MPU are carried by the MMTP packet with the value of the `FT` field set to 0. A `'moof'` box and the header of the following '`mdat`' box are carried by an MMTP packet with the value of the `FT` field set to 1. NAL Units in an '`mdat`' box are carried by MMTP packets with the value of the FT field set to 2. One NAL Unit can be fragmented into more than two MMTP packets. Or more than two complete NAL Units can be aggregated into a single MMTP packet.

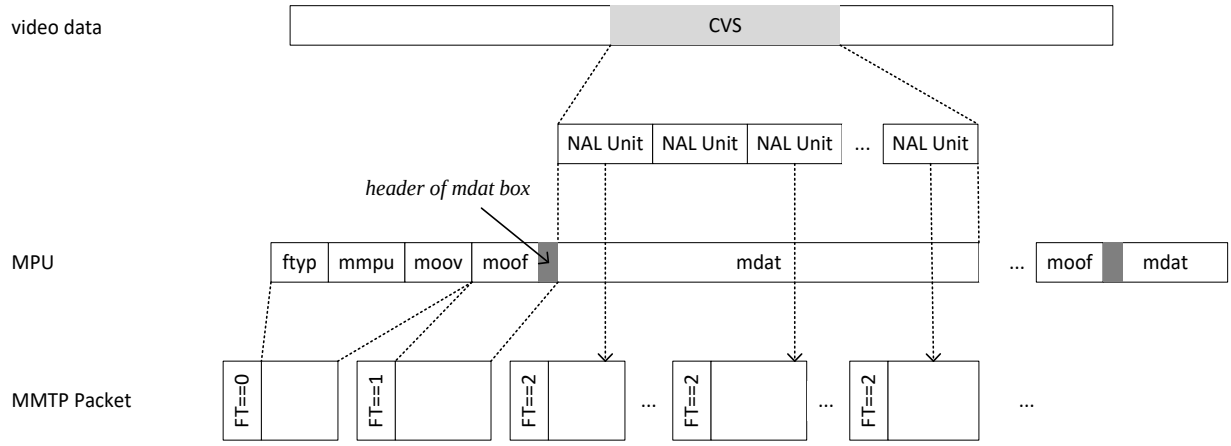


Figure 8.8 Example of media aware packetization of MPU.

8.1.2.3 Synchronization

The synchronization of MPUs shall be done by using timestamps referencing UTC delivered by ATSC 3.0 PHY layer.

The MPU_timestamp_descriptor as defined in subclause 10.5.2 of ISO/IEC 23008-1 [37] shall be used to represent the presentation time of the first media sample in presentation order in each MPU. The presentation time of each media sample of an MPU shall be calculated by adding the presentation time of the first media sample of an MPU to the value of the composition time of each sample in the MPU. The rule to calculate the value of the composition time of media samples in an MPU shall be be calculated by using the rule in the ISO BMFF specification [34].

8.1.2.4 Delivery of Locally-Cached Service Content

The ROUTE protocol shall be used to deliver files for locally-cached service content as specified in Section A.4 as informed by Section 8.1.1.6 when MMTP is used for streaming delivery. The file metadata, or equivalently, the Extended FDT, is assumed to be known, available, and delivered to the receiver before the scheduled delivery of the file. The Extended FDT is delivered in advance by the S-TSID metadata fragment as defined in Table 7.2. The access information for the S-TSID

[9] "movie fragment" is as defined by ISO/IEC 23009-1:2014 [55].

metadata fragment shall be provided by a **`ROUTEComponent`** element in the MMT-specific USBD as defined in Table 7.4.

To synchronize the presentation of locally-cached service content delivered by ROUTE with the presentation of MPUs delivered by MMTP, the presentation time for locally-cached service content shall be represented by a timestamp referencing UTC.

8.1.2.5 AL-FEC

Application layer forward error correction (AL-FEC) may be optionally applied. When AL-FEC is applied, the MMT AL-FEC framework shall be used as specified in Annex C of ISO/IEC 23008-1 [37] for MMTP sessions. When AL-FEC is applied to file delivery via ROUTE sessions, the repair protocol specified in Section A.4 of the ROUTE specification in Annex A shall be used.

8.2 Hybrid (Broadcast/Broadband) Delivery

Hybrid mode service delivery involves transport of one or more program elements over a broadband (Internet) path. This section specifies hybrid mode delivery. Section 8.2.1 specifies hybrid mode operation employing ROUTE/DASH in the broadcast path and DASH over HTTP(s) in the broadband path. Section 8.2.2 specifies a hybrid service delivery mode using MMTP/MPU in the broadcast path and DASH over HTTP(s) in the broadband path.

8.2.1 ROUTE/DASH Modes of Hybrid Service Access

Two modes of hybrid service delivery can be defined for the ROUTE/DASH system. The first features combined broadcast delivery via ROUTE, and unicast delivery via HTTP, of different content components, and this is described in Section 8.2.1.1. The second, as described in Section 8.2.1.2, features strictly broadband delivery of service components, and may comprise one of two sub-scenarios: a) all content components of the ATSC 3.0 service are exclusively delivered by broadband, as described in Section 8.2.1.2.1, and b) the broadcast signal becomes unavailable (e.g., due to device movement), resulting in a complete hand-off to broadband and subsequently, may involve a return to broadcast service, as described in Section 8.2.1.2.2.

8.2.1.1 Synchronization

Figure 8.9 illustrates hybrid mode operation and synchronization within the DASH Client.

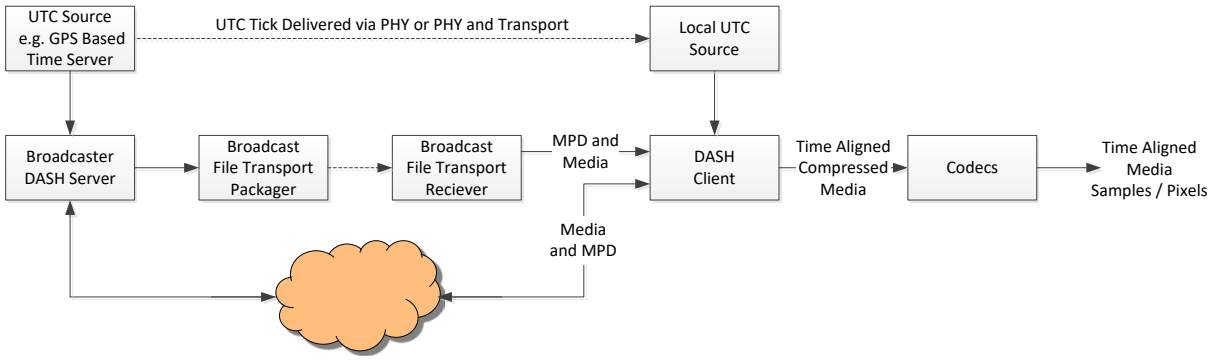


Figure 8.9 ROUTE/DASH hybrid synchronization.

The presence of an unmanaged broadband network introduces the possibility of a variable amount of extra delay (latency) in the delivery, as network congestion can impact the availability of the broadband-delivered content. MPEG DASH [55] includes a number of sophisticated methods to handle network jitter. The ROUTE/DASH proposal as specified in Annex A allows

the broadcaster to employ techniques specified in the DASH-IF [12] profile to optimize the user experience in the hybrid delivery case:

- A device with an excellent Internet connection may be able to get media data via broadband at least as fast as via broadcast. However, the broadcaster can ensure that broadband-delivered service components are made available to the broadcaster's broadband server well ahead of the broadcast-delivered components of the service to accommodate slower connections. This is especially easy for pre-produced (non-live) content and can be achieved for live content.

Buffering is handled slightly differently for the different delivery modes. DASH Segments for unicast broadband components are requested by receivers in advance of the time they are needed, as indicated by the timeline set by the DASH MPD, and they are buffered until their presentation times. The receivers control delivery timing and buffering. Segments for broadcast components are delivered from the broadcast source according to the buffer model, which ensures that receivers get the Segments soon enough to avoid decoder stall, and not so early that buffers could overflow. The broadcast source uses the DASH MPD timeline to determine the appropriate delivery timing.

8.2.1.2 Broadband DASH-only Service Access

8.2.1.2.1 Exclusive Delivery of Service Components via Broadband

In this mode, all content components of the ATSC 3.0 service shall be carried exclusively over broadband – i.e. no content components are delivered via broadcast. Such implementation is indicated by Service Layer Signaling (SLS) as specified in Section 7, specifically, by the combination of the unified MPD fragment and the **`UserServiceDescription.DeliveryMethod`** element of the User Service Bundle Description fragment. In this case, only the child element **`UnicastAppService`** shall be present under the **`DeliveryMethod`** element (i.e. **`BroadcastAppService`** is absent).

8.2.1.2.2 Hand-off from Broadcast to Broadband Service Access

This operational scenario involves the mobile ATSC 3.0 receiver, which due to user mobility, may move temporarily outside broadcast coverage and whereby only fallback service reception via broadband is possible. Similar to the scenario described in Section 8.2.1.2.1, support for such hand-off between different access modes may be indicated by the MPD fragment in SLS. In this case, as defined by the SLS protocols in Section 7, the **`UserServiceDescription.DeliveryMethod`** element in the User Service Bundle Description fragment shall contain both the child elements **`BroadcastAppService`** and **`UnicastAppService`**, which represent the broadcast and broadband delivered components, respectively, of the named service. Furthermore, these broadcast and unicast delivered components may be substituted for one another in the case of hand-off from broadcast to broadband service access, or vice versa, from broadband to broadcast service access.

8.2.2 MMTP/MPU Modes of Hybrid Service Access

8.2.2.1 Introduction

The overall hybrid streaming over broadcast and broadband delivery architecture for both the transmission system and the receiver system is shown in Figure 8.10. All the components of the system are locked to UTC for synchronization.

For the broadcast network, media data are encapsulated into MPUs, which are packetized into MMTP packets as specified in Section 8.1.2 of this document. For the broadband network, media data are encapsulated into DASH Segments. DASH Segments are delivered by an HTTP session through the network interface by a regular HTTP server while a DASH MPD is delivered via the

broadcast network. The URI reference for a DASH MPD shall be provided by an **`BroadbandComponent`** element in the MMT-specific USBD as defined in Table 7.4.

For the client it is assumed that the MMTP packets delivered through the broadcast network are de-packetized and the media data are decoded by the appropriate media decoders, and that the DASH Segments are delivered through the broadband network. To synchronize the presentation of a DASH Segment delivered via the broadband network with an MPU delivered via the broadcast network, the presentation time of the DASH Segment is represented by a timestamp referencing UTC.

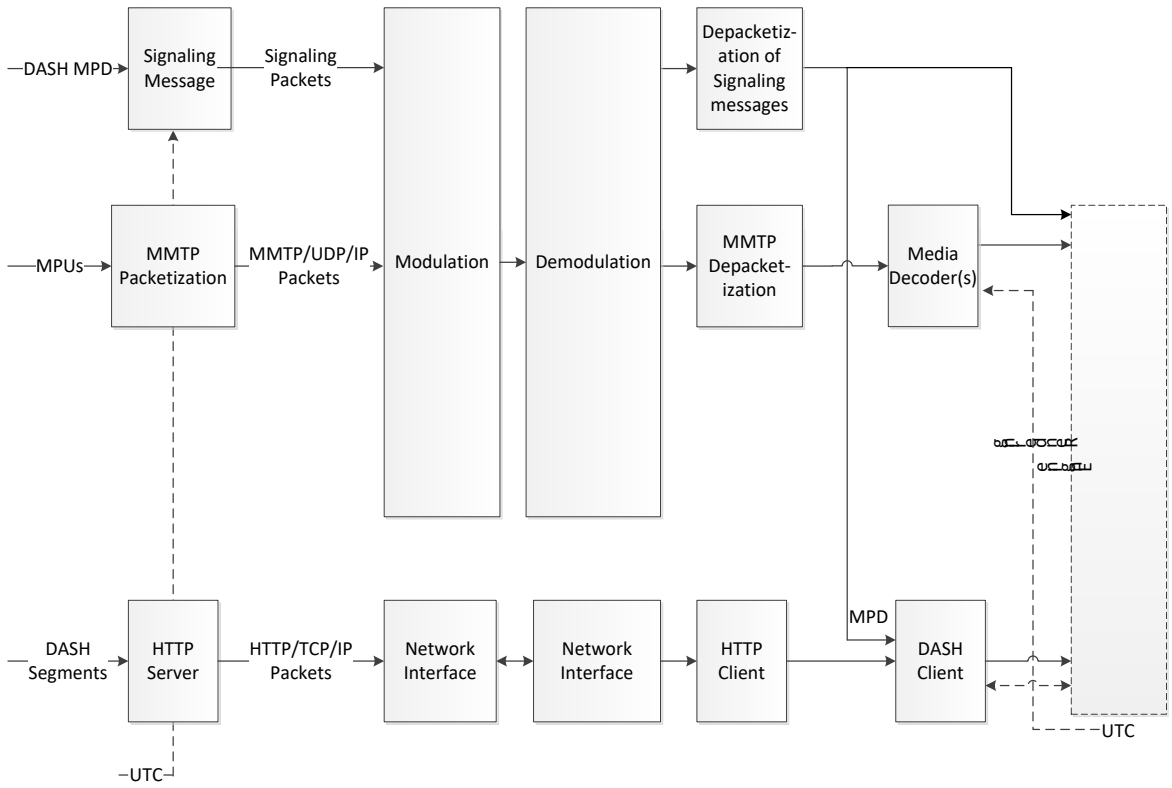


Figure 8.10 Conceptual hybrid service architecture.

8.2.2.2 Constraints on DASH

DASH Segments shall be used as the encapsulation format for broadband streaming media delivery.

8.2.2.3 Synchronization

The synchronization of streaming media shall use timestamps referencing UTC. The MPU_timestamp_descriptor as defined in subclause 10.5.2 of ISO/IEC 23008-1 [37] shall be used to represent the presentation time of the first media sample in presentation order in each MPU.

The presentation timing for DASH Segments delivered through HTTP shall be based on the DASH Media Presentation timeline referencing UTC. The DASH MPD shall provide attributes for calculating the presentation time of the first sample in presentation order.

Frame-level media timing of both MPUs and DASH Segments shall be as per the DASH-IF [12] profile.

8.2.2.4 Acquisition

Acquisition of broadcast MPU streaming shall be as specified as in Section 8.1.2. Acquisition of broadband content accessed through HTTP shall be as specified as in Section 8.2.1.2.

8.2.2.5 Broadband-Only Service (Signaled by Broadcast)

When all content components are delivered by broadband exclusively, media data is encapsulated into DASH Segments. DASH MPDs are delivered via the broadcast network by a signaling message and DASH Segments are delivered via the broadband network by an HTTP session through the network interface by a regular HTTP server.

8.3 File Repair Procedure

As described in Section 7.1.7, the file repair procedure is an HTTP request/response transaction whereby a receiver, unable to acquire the entire object delivered by ROUTE, can request and obtain the missing data via broadband from a file repair server. File repair is typically used to ensure reliable recovery of NRT content (app-based enhancements or data services). NRT content need not be consumed in real-time, and therefore the delay incurred by the request and return of missing data over broadband, after broadcast delivery has ended, is usually acceptable to the application making use of that data.

Under the assumption that the object delivered by ROUTE/broadcast was not completely received, usage of the file repair procedure, if desired by the receiver, will depend on the implementation method of the ROUTE protocol for NRT content delivery, as well as whether the receiver is AL-FEC enabled, for example:

- *Broadcast delivery of the file object does not employ AL-FEC*. This implies that only the ROUTE source protocol is implemented for the object delivery (i.e., the repair protocol is not implemented). In this scenario, the receiver need not be AL-FEC capable to perform file repair. It can request, from the file repair server, byte range(s) corresponding to the byte position(s) of the missing data in the delivery object.
- Broadcast delivery of the file object employs AL-FEC. This implies that the repair protocol is implemented in association with the source protocol for the object delivery. In this scenario, a receiver which is not AL-FEC capable can perform file repair in exactly the same manner as described in scenario (a) above. For AL-FEC capable receivers, it is expected to first employ AL-FEC decoding to determine whether the file can be fully recovered from the FEC symbols sent on the repair protocol. If file recovery is still unsuccessful, the AL-FEC capable receiver may request byte range(s) corresponding to the position(s) of a subset of the missing source symbols in the delivery object. Such subset represents the number of source symbols required by the receiver to ensure guaranteed object recovery, as a property of the AL-FEC scheme in use.

As described in the above examples, the file repair mechanism performed by the receiver is expected to differ depending on whether it is AL-FEC capable. In addition, as indicated in the above scenarios, the implementation of ROUTE should take into account the assumption by the broadcaster on AL-FEC capability of its targeted receiver population. For example, a broadcaster may choose to implement ATSC 3.0 services with the assumption that all targeted receivers are AL-FEC capable.

8.3.1 File Repair for Receivers that are not AL-FEC Capable

Assuming that the delivery object was not completely received during broadcast delivery via the ROUTE source protocol, a high-level summary of the file repair procedure performed by a receiver

that is not AL-FEC capable is provided below. In performing the file repair procedure, the receiver shall utilize the APD fragment (as previously described in Section 7.1.7) as well as the **EFDT** element in the S-TSID fragment (see Annex A, Section A.3.3.2.2), each of which are associated with the source flow carrying the delivery object of concern. The APD and the **EFDT** (corresponding to an Extended FDT) provide information on the location of the file repair server, and temporal parameters that govern when the file repair procedure should begin. The sequence of events are as follows:

1) The receiver determines that broadcast file delivery has ended;
2) It identifies missing data from the broadcast reception as the corresponding byte range(s) in the delivery object;
3) It calculates a random back-off time and selects a repair server from a list provided in the **EFDT**;
4) The receiver sends a HTTP repair request message for the missing data to the selected repair server at the computed time;
5) File repair server returns the requested byte ranges;
6) The file of interest is recovered, and forwarded to a cache or a service application.

The means by which the receiver may select and determine the location of a file repair server, and the deadline by which the receiver can expect the requested data to be returned by the repair server, are described in Section 8.3.3.

8.3.2 File Repair for Receivers that are AL-FEC capable

Here, similar to Section 8.3.1, it is assumed that the delivery object was not completely received during broadcast delivery via the ROUTE protocol, a high-level summary of the file repair procedure performed by a receiver that is AL-FEC capable is provided below. Also similar to Section 8.3.1, in performing the file repair procedure, the receiver shall utilize the APD fragment and the **EFDT** element in the S-TSID fragment to provide information on the location of the file repair server, and temporal parameters that govern when the file repair procedure should begin. The main difference between the scenarios in Sections 8.3.1 and 8.3.2 pertains to steps 2 and 4. Here, because the receiver is AL-FEC capable, it will determine the missing data as FEC symbol(s), and the byte range(s) in the HTTP request represents the mapping from the desired source symbol(s) to its(their) locations in the delivery object. The sequence of events are as follows:

1) The receiver determines that broadcast file delivery has ended;
2) It identifies missing data from the broadcast reception as FEC symbol(s), and converts that to one or more byte ranges corresponding to the positions of the desired source symbol(s) in the delivery object;
3) It calculates a random back-off time and selects a repair server from a list provided in the **EFDT**;
4) The receiver sends a HTTP repair request message for the missing data, as byte range(s) of source symbols(s), to the selected repair server at the computed time;
5) File repair server returns the requested byte ranges;
6) The file of interest is recovered, and forwarded to a cache or a service application.

8.3.3 Repair Server Location and Response Deadline

The location(s) of the one or more file repair servers, in the form of HTTP(s) URLs, which the receiver may contact to request repair data are provided by the **Alternate-Content-Location-1**

and **`Alternate-Content-Location-2`** child elements of the **`FDT-Instance.File`** element in the Extended FDT. These elements are specified in Section 9.3.5 of MBMS [14]. If file repair is available, at least one **`Alternate-Content-Location-1`** element in the Extended FDT shall be present, and if more than one such element is present, the receiver can choose any of them to use. **`Alternate-Content-Location-2`** if present, represents the one or more file repair servers as back-up to the main repair server(s) specified by **`Alternate-Content-Location-1`**. If more than one instance of **`Alternate-Content-Location-2`** is present and necessary for use as back-up repair server, the receiver can choose to use any of them. File repair server URL selection by the receiver, in decreasing priority is:

- Byte-range based repair servers included as **`Alternate-Content-Location-1`**;
- Byte-range based repair servers included as **`Alternate-Content-Location-2`**.

Base URLs against which URIs provided in **`Alternate-Content-Location-1`** and **`Alternate-Content-Location-2`** are resolved are provided in the **`Base-URL-1`** and **`Base-URL-2`** elements, also specified in MBMS [14]. When present, the **`Base-URL-`**`1` and **`Base-URL-2`** elements are present as child elements of the **`FDT-Instance`** and provide base URLs against which to resolve relative references included in **`Alternate-Content-Location-1`** and **`Alternate-Content-Location-2`** element, respectively.

When present, the **`FDT-Instance.File`**`@Availability-Time` attribute (also defined in MBMS [14]) indicates the latest wall-clock time, according to the UTC time standard, that the receiver should expect that, if reachable and functioning, the contacted file repair server will return the requested repair data.

Annex A: ROUTE

A.1 OVERVIEW OF ROUTE

A.1.1 General

The Real-time Object delivery over Unidirectional Transport (ROUTE) protocol is split in two major components:

- The source protocol for delivery of objects or flows/collection of objects (see Section A.3).
- The repair protocol for flexibly protecting delivery objects or bundles of delivery objects that are delivered through the source protocol (see Section A.4). The association between delivery objects delivered in the source protocol and the repair protocol is also provided in Section A.4.

The source protocol is independent of the repair protocol, i.e. the source protocol may be deployed without the ROUTE repair protocol. Repair may be utilized only for certain constrained or specialized deployment scenarios, for example only for mobile reception, only in certain geographical areas, only for certain service, etc.

A.1.2 Source Protocol

The ROUTE source protocol is aligned with FLUTE as defined in RFC 6726 [30] as well as the extensions defined in MBMS [14], but also makes use of some principles of FCAST as defined in RFC 6968 [52], for example, object metadata and the object content may be sent together in a compound object.

In addition to the basic FLUTE protocol, certain optimizations and restrictions are added that enable optimized support for real-time delivery of media data; hence, the name of the protocol. Among others, the source ROUTE protocol enables or enhances the following functionalities:

- Real-time delivery of object-based media data
- Flexible packetization, including enabling media-aware packetization as well as transport aware packetization of delivery objects
- Independence of files and delivery objects, i.e. a delivery object may be a part of a file or may be a group of files

A.1.3 Repair Protocol

The ROUTE repair protocol is FEC based and enabled as an additional layer between the transport layer (e.g., UDP) and the object delivery layer protocol. The FEC reuses concepts of FEC Framework defined in RFC 6363 [50], but in contrast to the FEC Framework in RFC 6363 [50] the ROUTE repair protocol does not protect packets, but instead it protects delivery objects as delivered in the source protocol. In addition, as an extension to FLUTE, it supports the protection of multiple objects in one source block which is in alignment with the FEC Framework as defined in RFC 6363 [50]. Each FEC source block may consist of parts of a delivery object, as a single delivery object (similar to FLUTE) or multiple delivery objects that are bundled prior to FEC

protection. ROUTE FEC makes use of FEC schemes in a similar way as those defined in RFC 5052 [21], and uses the terminology of that document. The FEC scheme defines the FEC encoding and decoding, as well as the protocol fields and procedures used to identify packet payload data in the context of the FEC scheme.

In ROUTE all packets are LCT packets as defined in RFC 5651 [26]. Source and repair packets may be distinguished by:

- Different ROUTE sessions; i.e., they are carried on different IP/UDP port combinations.
- Different LCT channels; i.e., they use different TSI values in the LCT header.
- The PSI bit in the LCT, if carried in the same LCT channel. This mode of operation is mostly suitable for FLUTE-compatible deployments.

A.1.4 Features

ROUTE defines the following features:

- Source protocol including packet formats, sending behavior and receiving behavior
- Repair protocol
- Metadata for transport session establishment
- Metadata for object flow delivery
- Recommendations for DASH configuration and mapping to ROUTE to enable rich and high-quality linear TV broadcast services. In addition, as an extension to FLUTE, it supports the protection of multiple objects in one source block which is in alignment with the FEC Framework as defined in RFC 6363 [50]

A.1.5 System Architecture

The scope of the ROUTE protocol is the reliable transport of delivery objects and associated metadata using LCT packets. This architecture is depicted in Figure A.1.1. In the figure, dotted-line connections illustrate the type of data being conveyed from one functional block to another.

The normative aspects of the ROUTE protocol focus on the following aspects:

- The format of the LCT packets that carry the delivery objects.
- The reliable transport of the delivery object using a repair protocol based on FEC.
- The definition and carriage of object metadata along with the delivery objects to enable the interface between the delivery object cache and the application.
- The ROUTE and LCT channel description to establish the reception of objects along with their metadata.
- The normative aspects (formats, semantics) of auxiliary information to be delivered along with the packets to optimize the performance for specific applications; e.g., real-time delivery. The objects are made available to the application through a Delivery Object Cache.

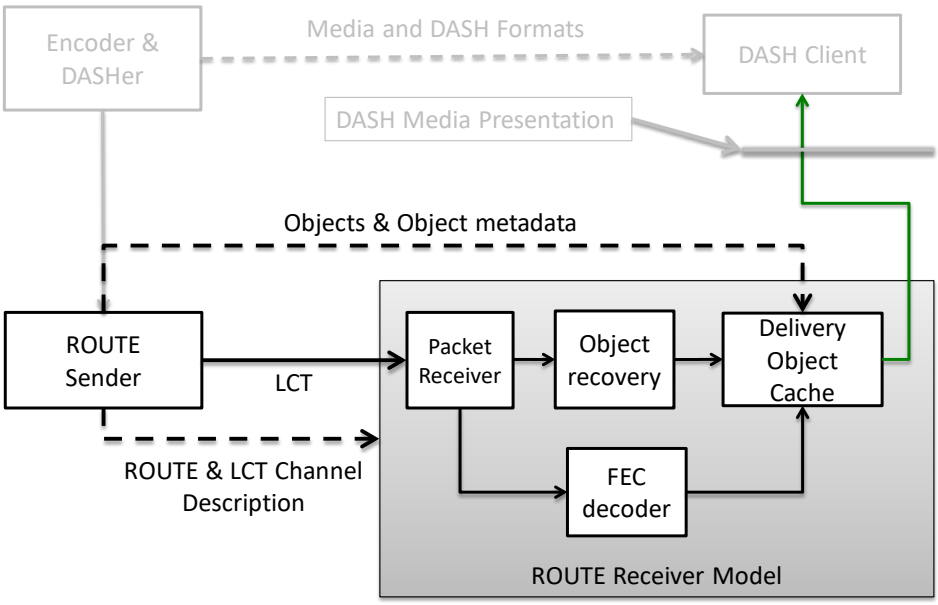


Figure A.1.1 Reference receiver architecture model in ROUTE.

Figure A.1.2 shows the basic sender concept. A ROUTE session is established that delivers LCT packets. These packets may carry source objects or FEC repair data. From a top down approach, a source protocol consists of one or more LCT channels, each carrying delivery objects and optionally, object metadata. The metadata may be statically delivered in signaling metadata or may be dynamically delivered, either as a compound object in the Entity Mode or as LCT extension headers in packet headers. The packets are carried in ALC using a specific FEC scheme that permits flexible fragmentation of the object at arbitrary byte boundaries. In addition, delivery objects may be FEC protected, either individually or in bundles. In either case, the bundled object is encoded and only the repair symbols are delivered, whereas the source symbols are not delivered. By themselves, or in combination with any received source packets, the received repair symbols permit the recovery of delivery object bundles. Note that one or more repair flows may be generated, each with different characteristics, for example to support different latency requirements, different protection requirements, etc.

The flexible FEC concept also permits different protection for different source streams, for example to differentiate the level of protection for the base layer and the enhancement layer when a scalable video codec is in use.

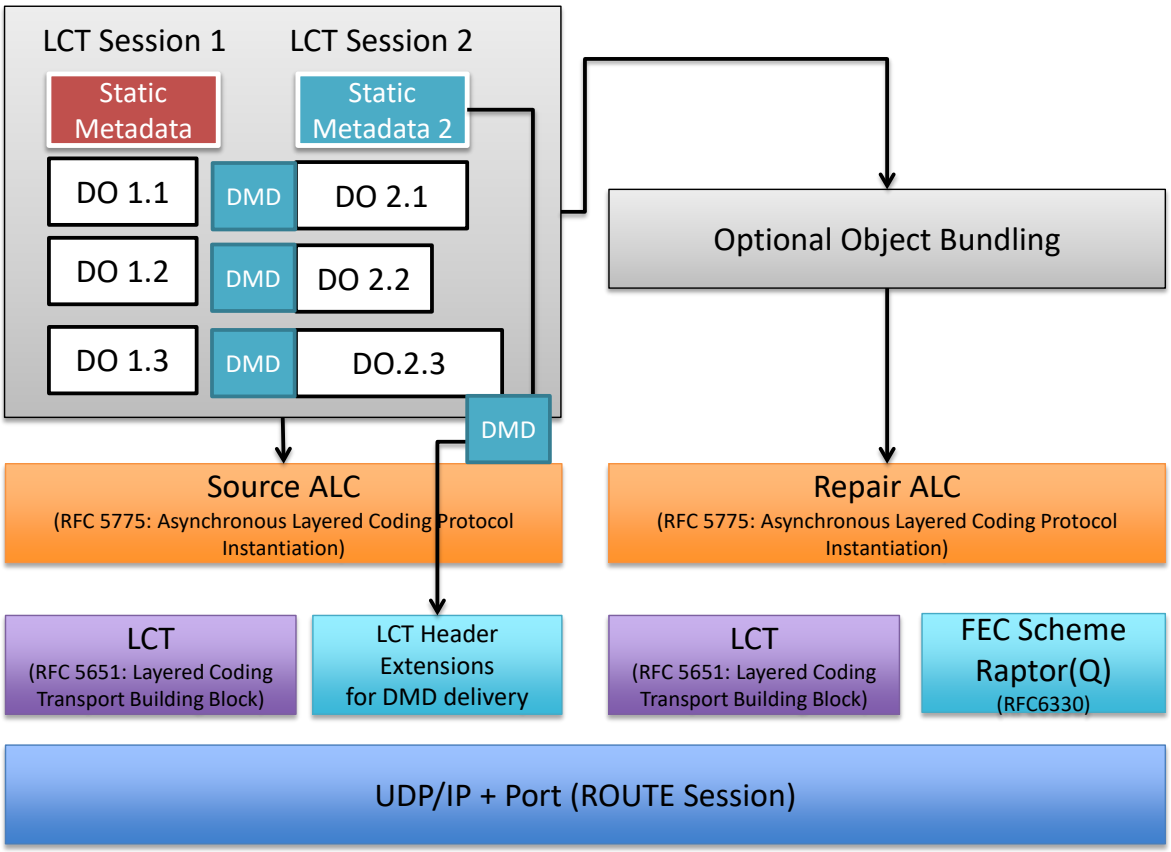


Figure A.1.2 Sender operation of ROUTE protocol.

The architecture supports different protection and delivery schemes of the source data. It also supports all existing use cases for NRT delivery, as it can be deployed in a backwards-compatible mode.

A.2 DATA MODEL AND SESSION INITIATION

A.2.1 Data Model

Each ROUTE session shall be associated with an IP address/port combination. Each ROUTE session shall constitute one or more LCT channels. LCT channels are a subset of a ROUTE session. For media delivery, an LCT channel would typically carry a media component, for example a DASH Representation. From the perspective of broadcast delivery of DASH formats, the ROUTE session can be considered as the multiplex of LCT channels that carry constituent media components of one or more DASH Media Presentations. Within each transport session, one or more objects are carried, typically objects that are related, e.g. DASH Segments associated to one Representation. Along with each object, metadata properties are delivered such that the objects can be used in application services which may include, but are not limited to, DASH Media Presentations, HTML-5 Presentations, or any other object-consuming application.

A.2.2 ROUTE Session

ROUTE sessions may be temporally bounded or unbounded.

A ROUTE session contains one or more LCT channels. Each transport session shall be uniquely identified by a unique Transport Session Identifier (TSI) value in the LCT header. The TSI is scoped by the IP address of the sender, and the IP address of the sender together with the TSI shall uniquely identify the session.

Before a receiver can join a ROUTE session, the receiver needs to obtain a ROUTE session description that contains at least the following information:

- The sender IP address
- The address and port number used for the session
- The indication that the session is a ROUTE session and that all packets are LCT packets
- Other information that is essential to join and consume the session on an IP/UDP level

The session description could also include, but is not limited to:

- The data rates used for the ROUTE session
- Any information on the duration of the ROUTE session

A.2.3 Transport Sessions

Transport sessions are not described in the ROUTE session description, but in signaling external to the ROUTE protocol itself. Transport sessions (i.e., LCT channels) may contain either or both

- Source Flows: In this case source data is carried.
- Repair flows: In this case repair data is carried.

A.3 SOURCE PROTOCOL SPECIFICATION

A.3.1 Overview

The source protocol is the core component of ROUTE; it is used to transmit delivery objects through a unidirectional channel. The source protocol establishes one or more source flows within a ROUTE session, each of which delivers related objects as an object flow. Each object is recovered individually.

The source protocol shall be defined by the description of a source flow as defined in Section A.3.2. A source flow sends delivery objects as defined in Section A.3.3. The usage of ALC [27] and LCT to deliver the objects shall be as defined in Section A.3.4. The packet format shall be as defined in Section A.3.5. The details on how to use LCT are defined in Section A.3.6. Section A.3.7 introduces newly-defined LCT headers. Sender and Receiver operations are provided in Sections A.3.8 and A.3.9, respectively.

A.3.2 Description

The **`SrcFlow`** element describes a source flow. The **`SrcFlow`** element is a child element of **`S-TSID.RS.LS`** (see Section 7.1.4).

A.3.2.1 Source Flow Syntax Description

While the indicated XML schema specifies the normative syntax of the **`SrcFlow`** element, informative Table A.3.1 below describes the structure of the **`SrcFlow`** element in a more illustrative way. The specifications following the table give the semantics of the elements and attributes.

Table A.3.1 XML Format of `SrcFlow` Element

Element or Attribute Name	Use	Data Type	Description
`SrcFlow`		srcFlowType	Source flow carried in the LCT channel.
`@rt`	0..1	boolean	Indication of whether the source flow conveys real-time content.
`@minBuffSize`	0..1	unsignedInt	The minimum number of kilobytes required in the receiver transport buffer for the LCT channel.
`EFDT`	0..1		The extended FDT instance.
`FDT-Instance`	1	fdt: FDT-InstanceType	A FLUTE FDT per RFC 6726 [30] with ATSC- and 3GPP-defined extensions.
`ContentInfo`	0..1		Additional information that can be mapped to the application service that is carried in this transport session.
`<choice>`			
`MediaInfo`	0..1		DASH Representation
`@startUp`	0..1	boolean	A Boolean flag, used for default “MPD-less startup” operation, that provides indication on whether the DASH resource carried by this LCT channel should be delivered to the media rendering application for decoding and rendering.
`@lang`	0..1	lang	The audio language of the DASH resource delivered by this LCT channel.
`@contentType`	0..1	contentType	The media type of the DASH resource delivered by this LCT channel.
`@repId`	1	StringNoWhitespace	Representation ID of the DASH Representation delivered by this LCT channel.
`ContentRating`	0..N		Content rating information associated with the program to which the media content of this source flow pertains
`@schemeIdUri`	0..1	anyURI	Content advisory rating scheme associated with the program to which the media content of this source flow pertains
`@value`	1	string	Content advisory rating value associated with the program to which the media content of this source flow pertains.
`AEAMedia`	0..1		Container of identifiers of AEA messages
`AEAId`	1..N	string	Identifier of an AEA message to which the AEA media files carried in this LCT channel are associated.
`Payload`	0..N		Information on the payload of ROUTE packets carrying the objects of the source flow
`@codePoint`	1	unsignedByte	A numerical representation of the combination of values specified for the attributes of this **`Payload`** element.
`@formatId`	1	unsignedByte	The payload format of the delivery object.
`@frag`	0..1	unsignedByte	Indication of how the payload of ROUTE packets carrying the objects of the source flow are fragmented for delivery.
`@order`	0..1	boolean	Indication of how the payload of ROUTE packets carrying the objects, or a portion thereof, of the source flow as DASH Segments are delivered relative to the order of their generation by the DASH encoder.
`@srcFecPayloadId`	0..1	unsignedByte	The implied meaning and representation of the ROUTE packet header “FEC Payload ID” in the packets carried by this source flow.

A.3.2.2 Source Flow Semantics

The following text specifies the semantics of the portion of elements and attributes in the **`SrcFlow`** fragment represented in Table A.3.1.

`SrcFlow` – A complex element whose subordinate elements and attributes shall contain information about the source flow (as defined in Annex A, Section A.3), if present, within its parent LCT channel.

`@rt` – A Boolean flag which shall indicate whether the content component of the ATSC 3.0 service carried by this source flow corresponds to real-time streaming media, or non-real-time content. When set to `"true"`, it shall be an indication of real-time content, and if absent, or set to `"false"`, it shall be an indication of non-real-time content.

`@minBufferSize` – A 32-bit unsigned integer which shall represent, in kilobytes, the minimum required storage size of the receiver transport buffer, for the parent LCT channel of this source flow. This attribute shall only be applicable when `@rt = "true"`. If `@rt "true"` and this attribute is absent, the minimum receiver transport buffer size is unknown.

`EFDT` – An optional element that, when present, shall contain a single instance of an **`FDT-Instance`** element per RFC 6726 [30] FLUTE, which may contain FDT extensions as defined in Sections A.3.3.2.3, A.3.3.2.4 and A.3.3.2.5. The **`EFDT`** element shall be present if this source flow transports streaming media segments. The presence of the **`SrcFlow.EFDT`** element and an EFDT transported in TOI=0 of the same LCT channel carrying this source flow shall be mutually exclusive.

`ContentInfo` – A complex element whose child elements and attributes shall provide additional information for the content carried by this source flow. An LCT channel should be restricted to contain only a single type of content (i.e., either media content of a DASH-formatted streaming service, or emergency alert media files), and if the parent LCT channel contains any Segments of a DASH Representation, it should contain all the Segments of that Representation and no Segments of any other Representation. Indication of the type of content present in ContentInfo is implemented by the <**`choice`**> element in XML schema. If the **`ContentInfo`** element is absent, it means that no additional information is available on the content component of this source flow.

`MediaInfo` – A complex element whose subordinate attributes shall contain information about the DASH resource delivered by the parent LCT channel and source flow.

`@startUp` – This Boolean attribute shall indicate whether the DASH resource carried by the parent LCT channel and source flow should be delivered to the media rendering application for decoding and rendering in a “MPD-less start-up” mode of operation, whereby reception of the MPD is not required to initiate service playout in support of fast start-up upon a channel change using MDE delivery mode. A value of `"true"` shall indicate that MPD-less start and playback of the media content delivered by this source flow is permitted. A value of `"false"` shall indicate that MPD-less start-up is not allowed. The default value of this attribute shall be `"false"`. It is recommended that `@startup` should be set to `"true"` if and only if the presentation times of the first sample, in presentation order, of Media Segments whose Representations are associated with `@startup="true"` are exactly identical, to ensure synchronized play-out. It is expected that the ROUTE receiver, upon acquiring the MPD, will immediately forward the MPD to the DASH client. Such behavior, in the event that media playout has occurred as result of `@startup="true"` and before MPD reception, enables normal

function of the DASH client in MPD processing to support advanced media selection and playback features.

`@lang` – This xs:lang attribute shall indicate the audio language of the DASH resource carried by the parent LCT channel and source flow.

`@contentType` – This xs:string attribute shall indicate the media type of the content associated with the DASH resource carried by the parent LCT channel and source flow. If present, one of the following three content types shall be specified:

- audio
- video
- subtitles

`@repId` – This xs:string attribute shall correspond to the Representation ID of the DASH Representation being delivered in this LCT channel and source flow. Its value shall be passed by the DASH player to the ROUTE receiver upon the DASH player having selected the Representation(s) to be rendered. In turn, the ROUTE client will be able to download only those Representations for which the DASH client has forwarded the `@repId` attribute, instead of downloading all Representations available to the receiver.

`ContentRating` – This element shall provide content rating information associated with the program to which the media content of this source flow pertains. Absence of this element shall imply that the program is not rated.

`@schemeIdUri` – An attribute which shall identify the content advisory rating scheme associated with the program to which the media content of this source flow pertains. This string, when present shall be set equal to `"tag:atsc.org,2016:carating:1"` to indicate RRT-based ratings. Other values are undefined. The default value of this attribute shall be `"tag:atsc.org,2016:carating:1"`.

`@value` – An attribute which shall identify content advisory rating value associated with the program to which the media content of this source flow pertains. When the value of `@schemeIdUri` is `"tag:atsc.org,2016:carating:1"`, this string shall be set equal to the content advisory ratings string specified in Section 7.3.1.

`AEAMedia` – A complex element which shall contain a list of identifiers for AEA messages to which the AEA media files carried in the parent LCT channel and source flow are associated.

`AEAId` – This xs:string element shall represent the identifier of an individual AEA message. Its value may be used by the receiver to be linked to the **`Media`** element, associated with the AEA message, whose referenced multimedia resources are delivered in the parent LCT channel of this **`ContentInfo`** instance. Based on the filter parameters tagged to each AEA message (`@priority` and **`Location`**) for use by the receiver to determine whether the AEA message should be displayed to the user, the receiver may decide whether or not to download the multimedia resource file(s) associated with that AEA message.

`Payload` – A complex element whose attributes shall contain information about each type of payload as mapped by Codepoint (CP) value to the corresponding delivery object carried by ROUTE packets of this source flow. A different Payload element shall be present for each value of the Codepoint (CP) field in the LCT headers in use for this source flow, for CP values 128 or above. For CP values below 128, the Payload elment is not required because the attributes of the associated packets are as given in Table A.3.6.

`@codePoint` – This required 8-bit unsigned byte value shall be in the range 128 to 255, inclusive. The value represents a numerical “shorthand” or “alias” for the specific combination of values

specified for the other attributes of this **`Payload`** element as given in Table A.3.6. The value of this attribute shall reference the Codepoint (CP) field in the LCT header. For example, if **`Payload`**`@codePoint`=130, LCT packets associated with this source flow with CP=130 have the format given in the `@formatId`, `@frag`, and `@order` attributes given in this same **`Payload`** element.

`@formatId` – An 8-bit unsigned byte value which shall represent the payload format of the delivery object, and which shall be encoded according to the delivery object format ID values as specified in Table A.3.2.

`@frag` – An 8-bit unsigned byte value which shall indicate how the payload of ROUTE packets carrying the objects of the source flow are fragmented for delivery. The following values are currently defined:

0: arbitrary. This value shall mean that the payloads of ROUTE packets with CP value equal to **`Payload`**`@codePoint` carry a contiguous portion of the delivery object whose fragmentation occurs at arbitrary byte boundaries.

1: application specific (sample based). This value shall mean that the payloads of ROUTE packets with CP value equal to **`Payload`**`@codePoint` carry media data in the form of one or more complete samples, where the term “sample” is as defined in ISO/IEC 14496-12 [34]. Its usage shall pertain to the MDE mode as described in Section 8.1.1.5.2, whereby the packet strictly carries an MDE data block comprising samples stored in the ‘mdat’ box.

2: application specific (a collection of boxes). This value shall mean that the payloads of ROUTE packets with CP value equal to **`Payload`**`@codePoint` contain the entire data content of one or more boxes, where term “box” is as defined in ISO/IEC 14496-12 [34]. Its usage shall pertain to the MDE mode as described in Section 8.1.1.5.2, whereby each packet carries the portion of an MDE data block starting with RAP, and strictly comprising boxes which contain metadata (e.g. styp, sidx, moof and their contained (subordinate) boxes).

3-255: ATSC reserved

The default value of this attribute is "0".

`@order` – This Boolean attribute shall indicate whether the payload of ROUTE packets with CP value equal to **`Payload`**`@codePoint` carrying the objects, or a portion thereof, of the source flow as DASH Segments are delivered in the order of their generation by the DASH encoder. When the value is `"true"`, in order delivery is indicated. When the value is `"false"`, arbitrary order of delivery is indicated. When this attribute is absent, the default value shall be `"true"`.

`@srcFecPayloadId` – The value of this xs:unsignedByte attribute shall convey the meaning and, if applicable, the data format of the FEC Payload ID field in the header of ROUTE packets with CP value equal to **`Payload`**`@codePoint` which carry the delivery object in this source flow, as follows:

0: the FEC Payload ID field is a 32-bit unsigned integer value that expresses the start offset in the object. Start offset is defined in Section A.3.5.

1-255: ATSC reserved.

The default value of this attribute is "0".

A.3.3 Delivery Objects

A.3.3.1 Overview

The ROUTE protocol enables delivery of delivery objects to receivers which recover the delivery objects and pass them to the application.

A delivery object is self-contained for the application, typically associated with certain properties, metadata and timing-related information that are of relevance for the application. In some cases the properties are provided in-band along with the object, in other cases the data needs to be delivered out-of-band in a static or dynamic fashion.

This protocol enables delivery of at least the following delivery objects:

1) Complete or partial files described and accompanied by an Extended FDT Instance.
2) HTTP Entities (HTTP Entity Header and HTTP Entity Body).
3) Packages of delivery objects.

In particular, a type 1 or 2 delivery object may be further differentiated by:

- Whether it corresponds to a full file or a byte range of a file, along with an Extended FDT Instance.
- Whether it represents timed or non-timed delivery. If timed, certain real-time and buffer restrictions apply and specific extension headers may be used.
- Usage of dynamic and/or static metadata to describe delivery object properties.
- Delivery of specific data structures, especially ISO BMFF files. In this case a media-aware packetization or a general packetization may be applied.

The delivery format ID specifies the payload format used in order to provide information to the applications. Values of delivery object format ID shall be as specified in Table A.3.2.

Table A.3.2 Meaning of the Delivery Object Format ID Values

Delivery Object Format ID Value	Meaning
0	ATSC Reserved
1	File Mode as defined in Section A.3.3.1.1
2	Entity Mode as defined in Section A.3.3.3
3	Unsigned Package Mode as defined in Section A.3.3.4
4	Signed Package Mode as defined in Section A.3.3.5
≥ 5	ATSC Reserved

A.3.3.1.1 Content Encoding

Content encoding may be used for delivery of any file described by an **`FDT-Instance.File`** element in the EFDT. The content encoding defined in the present standard is gzip [16]. When content encoding is used, the **`File`**`@Content-Encoding` and **`File`**`@Content-Length` attributes shall be present in the EFDT.

A.3.3.2 File Mode

A.3.3.2.1 Background

In the file mode, the delivery object represents a file or a byte range of the file. This mode replicates FLUTE as defined in RFC 6726 [30], but with the ability to send static and pre-known file metadata and the associated delivery object via separate LCT channels, as shown in Figure A.3.1.

In FLUTE, FDT Instances are delivered in-band and need to be generated and delivered in real-time if objects are generated in real-time at the sender. In contrast to the FDT in RFC 6726 [30], Section 3.4.2 and MBMS [14], Section 7.2.10, besides separated delivery of file metadata from the delivery object it describes, the FDT functionality in ROUTE may be extended by additional file metadata and rules that enable the receiver to generate the `Content-Location` attribute of the File element of the FDT, on-the-fly, by using information in both the extensions to the FDT and the LCT header. The combination of pre-delivery of static file metadata and receiver self-generation of dynamic file metadata avoids the necessity of continuously sending the FDT Instances for real-time objects. Such modified FDT functionality in ROUTE is referred to as the Extended FDT.

Figure A.3.1 below illustrates the functional difference between FLUTE and ROUTE regarding FDT/ Extended FDT Instance delivery and recovery. It is not intended to prescribe any particular means of implementation, but is only providing a reference model.

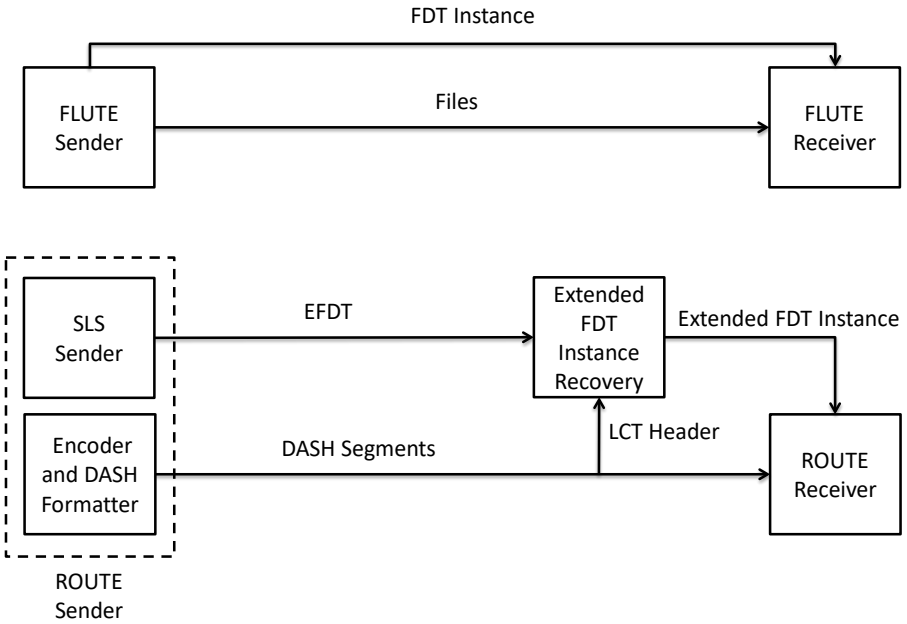


Figure A.3.1 ROUTE distribution in file mode compared to FLUTE distribution.

The following methods can be used for delivering the Extended FDT Instance:

- As the **`EFDT`** element embedded in the **`SrcFlow`** element;
- As the **`FDT-Instance`** element, as a separate delivery object referenced by the signaling metadata and which is in turn carried in the same ROUTE session and constituent LCT channel that carries the delivery object described by this Extended FDT.

When the Extended FDT Instance (**`FDT-Instance`** element) is delivered as a separate delivery object in the same LCT channel that carries the delivery object described by this Extended FDT Instance, ROUTE in File mode operates in a delivery mode compatible to legacy FLUTE.

The Extended FDT represents file description entries for the one or more delivery objects carried in the parent source flow. The Extended FDT contains the descriptions of the affiliated delivery objects, including i) nominal FLUTE FDT parameters as defined in RFC 6726 [30], ii) certain extensions to the FLUTE FDT as defined by 3GPP in MBMS [14], and iii) ATSC-defined extensions to the FLUTE FDT as specified in this section. Presence of the **`EFDT`** child element of

the **SrcFlow** element shall be the indication that the Extended FDT Instance is embedded in the S-TSID. Absence of this element shall be an indication that an Extended FDT Instance (an **FDT-Instance** element) is carried as a separate delivery object, with TOI = '0', in the LCT channel of the associated source flow. In the latter (referencing) case, the Extended FDT Instance may be updated independently of the signaling metadata.

The elements and attributes defined in this section are extensions to the standard RFC 6726 [30] FDT and may be added at permissible locations as specified below.

A.3.3.2.2 Semantics Overview

The FDT extensions shall be represented as an XML data structure as elements and attributes that conform to the definitions in the XML schema that has namespace:

```
tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/ATSC-FDT/1.0/
```

The definition of this schema is in an XML schema file, ATSC-FDT-1.0-20170920.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"afdt"`.

The indicated XML schema specifies the normative syntax of the FDT extensions.

A.3.3.2.3 ATSC Extensions to the `FDT-Instance` Element

Informative Table A.3.3 below lists the ATSC-defined extensions to the **FDT-Instance** element. The semantics of the attributes described in Table A.3.3 shall be as given below the table.

Table A.3.3 ATSC-Defined Extensions to **FDT-Instance** Element

Element or Attribute Name	Use	Data Type	Description
`@efdtVersion`	0..1	unsignedByte	The version of this Extended FDT instance descriptor.
`@maxExpiresDelta`	0..1	unsignedInt	Time interval for use in deriving the expiration time of the associated EFDT.
`@maxTransportSize`	0..1	unsignedInt	The maximum transport size of any object described by this EFDT.
`@appContextIdList`	0..1	afdt:uriList	A space-separated list of URIs representing one or more unique Application Context Identifiers.
`@fileTemplate`	0..1	string	Describes the means to generate the file URL, i.e. `Content-Location` attribute of the FDT.
`@filterCodes`	0..1	afdt:listOfUnsignedInt	Filter Codes applying to all files in the Source flow.

`@efdtVersion` – An 8-bit unsigned integer value that shall represent the version of this **EFDT** element. The version shall be increased by one modulo 256 each time the **EFDT** element is updated. `@version` is an ATSC-defined extension of the FLUTE FDT as specified in RFC 6726 [30].

`@maxExpiresDelta` – A 32-bit unsigned integer value that shall represent a time interval in number of seconds, which when added to the wall-clock time at the receiver when it acquires the first ROUTE packet carrying the object described by this EFDT, shall represent the expiration time of the associated **EFDT**. If `@maxExpiresDelta` is not present, the expiration time of the **EFDT** shall be given by the sum of a) the value of the `ERT` field in the `EXT_TIME` LCT header extension in the ROUTE packet and b) the current receiver time when parsing the packet header. See

Section A.3.3.2.7 on additional rules for deriving the **EFDT** expiration time. `@maxExpiresDelta` is an ATSC-defined extension of the FLUTE FDT as specified in RFC 6726 [30].

`@maxTransportSize` – A 32-bit unsigned integer value that shall represent the maximum transport size in bytes of any delivery object described by this **EFDT**. This attribute shall be present if 1) the **FDT-Instance**`@fileTemplate` is present; or 2) one or more **File** elements in this EFDT do not include the `@Transfer-Length` attribute. When `@maxTransportSize` is not present, the maximum transport size is not signaled.

`@appContextIdList` – A list of unique URI values, separated by spaces, which when present, define one or more Application Context Identifiers associated with the containing elements. Files may be associated with multiple application contexts. The Application Context Identifier provides a mechanism of grouping files for use with Broadcaster Applications as defined in A/344, Interactive Content [45].

Any Application Context Identifiers contained as an attribute of the Extended FDT element, **FDT-Instance**`@appContextIdList`, are associated with all files defined as children of the **FDT-Instance** data structure. Any Application Context Identifiers defined in the list attribute of the File element, **File**`@appContextIdList`, are associated with that file only. Thus, the total collection of Application Context Identifiers associated with a specific file consists of the union of the two lists: **File**`@appContextIdList` and **FDT-Instance**`@appContextIdList`. For the complete definition and usage, please see Section A.3.3.2.3 above.

Each Application Context Identifier shall be a globally-unique URI that allows files to be grouped together. Receivers use the Application Context Identifier to provide access to the associated files such that the group of files from one Application Context Identifier is completely separate from the group of files assigned to another Application Context Identifier. A file may be associated with multiple Application Context Identifiers, each resulting in a different URL for accessing the file. In this way, files may be shared among applications.

For additional discussion and details of the Application Context Identifier concept, refer to A/337, Application Signaling [7] and A/344, Interactive Content [45].

`@fileTemplate` – A string value, which when present and in conjunction with parameter substitution, shall be used in deriving the file URL, i.e., the `Content-Location` attribute, for the delivery object described by this **EFDT**. The mechanism, as further described in Section A.3.3.2.7, shall involve substituting the TOI value of the delivery object for the pattern `'$TOI$'`, in the string representation of `@fileTemplate`. The derived `Content-Location` shall be a relative URI conforming to the provisions of Section 8.1.1.2 for broadcast-delivered resources. By this means, a one-to-one mapping is created between the TOI and the file URL. It also implies that in the event that each delivery object of the source flow is a DASH Segment, the Segment number will be equal to the TOI value of the object. The `@fileTemplate` attribute shall be present in the **EFDT** when the LCT Channel carries a sequence of DASH Media Segment files.

`@filterCodes` – A space-separated list of 32-bit unsigned integers that shall represent Filter Codes that are associated with all files identified in this EFDT except for those files in which a **File**`@filterCodes` attribute is present, in which case the Filter Codes in the **File**`@filterCodes` attribute shall take precendence.

A.3.3.2.3.1 ATSC Extensions to the **FDT-Instance.File** Element

Informative Table A.3.4 below lists the ATSC-defined extensions to the **FDT-Instance.File** element. The semantics of the attributes described in Table A.3.4 shall be as given below the table.

Table A.3.4 ATSC-Defined Extensions to **`FDT-Instance.File`** Element

Element or Attribute Name	Use	Data Type	Description
`@appContextIdList`	0..1	afdt:uriList	A space-separated list of URIs representing one or more unique Application Context Identifiers.
`@filterCodes`	0..1	afdt:listOfUnsignedInt	A space-separated list of 32-bit unsigned integers representing Filter Codes that applies to this file.

`@appContextIdList` – A list of unique URI values, separated by spaces, which when present, define one or more Application Context Identifiers associated with the containing instance of the **`File`** element. If **`FDT-Instance`**`@appContextIdList` is present, the total collection of Application Context Identifiers associated with the file shall consist of the union of the two lists: **`File`**`@appContextIdList` and **`FDT-Instance`**`@appContextIdList`. If all instances of the **`File`** elements were to contain the same `@appContextIdList` value, then the **`FDT-Instance`**`@appContextIdList` attribute is expected to be present with that value, and the `@appContextIdList` attribute should be omitted under all instances of the **`FDT-Instance.File`** element. When this attribute is not present, there is no default value.

`@filterCodes` – A space-separated list of 32-bit unsigned integers that represent Filter Codes that are associated with the parent **`FDT-Instance.File`** element. This attribute should not be present when the `@appContextIdList` attribute is not present. When this attribute is not present, there is no default value.

A.3.3.2.4 3GPP Extensions to the **`FDT-Instance`** Element

The Extended FDT may include one or more of the following elements defined by MBMS [14] in the the **`FDT-Instance`** element following all **`File`** elements. This is a brief summary. See MBMS [14] for more information about the use of these elements:

`Base-URL-1` – Optional base URL against which to resolve a relative reference included in any **`Alternate-Content-Location-1.Alternate-Content-Location`** element.

`Base-URL-2` – Optional base URL against which to resolve a relative reference included in any **`Alternate-Content-Location-2.Alternate-Content-Location`** element.

A.3.3.2.5 3GPP Extensions to the **`FDT-Instance.File`** Element

The Extended FDT may include one or more of the following elements and attributes defined by MBMS [14] in the the **`FDT-Instance.File`** element. This is a brief summary. See MBMS [14] for more information about the use of these elements:

`Cache-Control` – An optional element specified in the `mbms2007` namespace. Its use in the FDT shall be per [14]. The **`Cache-Control`** element may appear at most one time.

`Alternate-Content-Location-1` – Container for the location(s) of primary file repair server(s) and deadline for returning file repair data to the receiver. Zero or more instances may appear.

`Alternate-Content-Location-1``@Availability-Time` – An optional attribute indicating the latest wall-clock time that the receiver should expect that, if reachable and functioning, the contacted file repair server will return the requested repair data.

`Alternate-Content-Location-1.Alternate-Content-Location` – An child element of **`Alternate-Content-Location-1`** that indicates the source location(s) of primary file repair server(s) for this file.

`Alternate-Content-Location-2` – Container for the location(s) of back-up file repair server(s) and deadline for returning file repair data to the receiver

`Alternate-Content-Location-2``@Availability-Time` – An optional attribute indicating the latest wall-clock time that the receiver should expect that, if reachable and functioning, the contacted file repair server will return the requested repair data

`Alternate-Content-Location-2.Alternate-Content-Location` – Resource location(s) of back-up file repair server(s) for this file.

A.3.3.2.6 Extended FDT Instance Semantics

The Extended FDT Instance shall conform to an FDT Instance according to RFC 6726 [30]. This means that:

- At least one **`File`** element must be present, and
- The `@Expires` attribute must be present.

If a `@fileTemplate` attribute is present, then the sender shall operate as follows:

- The TOI shall be set such that `Content-Location` can be derived according to Section A.3.3.2.7.
- After sending the first packet of a TOI, none of the packets pertaining to this TOI shall be sent later than as indicated by `@maxExpiresDelta`. In addition, the `EXT_TIME` header with Expected Residual Time (ERT) may be used in order to convey more accurate expiry time, if considered useful. If `@maxExpiresDelta` is not present, then the `EXT_TIME` header with Expected Residual Time (ERT) shall be used to signal the value of `@Expires`, according to the procedure described below in Section A.3.3.2.7.

If a `@fileTemplate` attribute is present, an Extended FDT Instance is produced at the receiver as follows:

- Any data that is contained in the **`EFDT`** may be used as is in generating an Extended FDT Instance.
- The data in the `@fileTemplate` attribute is used to generate the file URI (equivalent to the **`File`**`@Content-Location` in the FDT) as documented in Section A.3.3.2.7 with the reception of an LCT packet with a specific TOI value.

A.3.3.2.7 File Template

If an LCT packet with a new TOI is received for this transport session, then an Extended FDT Instance is generated with a new **`File`** entry as follows:

- The **`TOI`** is used to generate **`File`**`@Content-Location` using the mechanism defined in Section A.3.3.2.8.
- All other attributes that are present in the **`EFDT.FDT-Instance`** element are applicable to the **`File`**.
- Either the `EXT_FTI` header (per RFC 5775 [27]) or the `EXT_TOL` header (per Section A.4.2.6.1), when present, shall be used to signal the Transport Object Length (TOL) of the **`File`**. If the **`File`**`@Transfer-Length` parameter in the Extended FDT Instance is not present, then the `EXT_TOL` header or the or `EXT_FTI` header shall be present. Note that a header containing the transport object length (`EXT_TOL` or `EXT_FTI`) need not be present in each packet header. If the broadcaster does not know the length of the transport object at

the beginning of the transfer, an `EXT_TOL` or `EXT_FTI` header shall be included in at least the last packet of the file and should be included in the last few packets of the transfer.

- If present, the `@maxExpiresDelta` shall be used to generate the value of the `@Expires` attribute. The receiver is expected to add this value to the current time to obtain the value for `@Expires`. If not present, the `EXT_TIME` header with Expected Residual Time (ERT) shall be used to extract the expiry time of the Extended FDT Instance. If both are present, the smaller of the two values should be used as the incremental time interval to be added to the receiver's current time to generate the effective value for the `@Expires`. If neither `@maxExpiresDelta` nor the ERT field of the `EXT_TIME` is present, then the expiration time of the Extended FDT is given by the Extended FDT Instance's `@Expires` attribute.

A.3.3.2.8 Substitution

The `@fileTemplate` attribute, when present, shall include the "`$TOI$`" identifier. After parameter substitution using the TOI number in this transport session, the `@fileTemplate` shall be a valid URL corresponding to the `Content-Location` attribute of the associated file. Excluding the TOI values associated with any files listed in **`FDT-Instance.File`** elements, the `@fileTemplate` attribute generates a one-to-one mapping between the TOI and the `Content-Location` value. When the `@fileTemplate` is used to identify a sequence of DASH Media Segments, the Segment number is equal to the TOI value

In each URI, the identifiers from Table A.3.5 shall be replaced by the substitution parameter defined in Table A.3.5. Identifier matching is case-sensitive. If the URI contains unescaped `$` symbols which do not enclose a valid identifier then the result of URI formation is undefined. The format of the identifier is also specified in Table A.3.5.

Each identifier may be suffixed, within the enclosing '$' characters following this prototype: `%0[width]d`

The `width` parameter is an unsigned integer that provides the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result shall be padded with zeroes. The value is not truncated even if the result is larger.

An example `@fileTemplate` using a width of 5 is: `fileTemplate="myVideo$TOI%05d$.mps"`, resulting in file names with exactly five digits in the number portion. The Media Segment file name for TOI=33 using this template is `myVideo00033.mps`.

The `@fileTemplate` shall be authored such that the application of the substitution process results in valid URIs.

Strings outside identifiers shall only contain characters that are permitted within URIs according to RFC 3986 [19].

Table A.3.5 Identifiers for File Templates

$<Identifier>$	Substitution Parameter	Format
$$	Is an escape sequence, i.e. "$$" is non-recursively replaced with a single "$"	not applicable
TOI	This identifier is substituted with the TOI.	The format tag may be present. If no format tag is present, a default format tag with width=1 shall be used.

A.3.3.3 Entity Mode

In the Entity Mode, the delivery object is analogous to a representation in HTTP/1.1 and whose data is enclosed in the payload body of a HTTP message. The delivery object is described by metadata in the form of “entity headers” which correspond to one or more of the representation header fields, payload header fields and response header fields as defined in Sections 3.1, 3.3 and 7, respectively, of RFC 7231 [31].

This mode reuses the major concepts of FCAST as defined in RFC 6968 [52], but permits delivery of any metadata suitable for the delivery object via one or more of the aforementioned header field types including extension headers.

The entity headers sent along with the file provides all information for the contained complete or partial file. Note that if the headers include a `Content-Range` header, then the delivery object represents a portion of the target file, in the form of a byte range. In addition, this mode also allows for chunked delivery in case the headers contain the related signaling.

A.3.3.4 Unsigned Package Mode

In this delivery mode, the delivery object consists of a group of files that are packaged for delivery only. If applied, the client is expected to unpack the package and provide each file as an independent object to the application. Packaging is supported by Multipart MIME [18], where objects are packaged into one document for transport. Files delivered in Unsigned Package Mode shall be delivered using **`SrcFlow.Payload`**`@formatId = "3"` with `Content-Type` set to `multipart/related`. When binary files are included in the package, `Content-Transfer-Encoding` of "binary" should be used for those files. Note: `Content-Transfer-Encoding` of "binary" may be uncommon in the typical use of Multipart MIME in email.

A.3.3.5 Signed Package Mode

In Signed Package Mode delivery, the delivery object consists of a group of files that are packaged for delivery, and the package includes one or more signatures for validation. Files delivered in Signed Package Mode shall be delivered using **`SrcFlow.Payload`**`@formatId = "4"` with `Content-Type` set to `multipart/signed`. Signed packaging is supported by RFC 5751 Secure MIME (S/MIME) [22], where objects are packaged into one document for transport and the package includes objects necessary for validation of the package. [10]. When binary files are included in the package, `Content-Transfer-Encoding` of "binary" should be used for those files. Note: `Content-Transfer-Encoding` of "binary" may be uncommon in the typical use of Multipart MIME in email.

A.3.4 Usage of ALC and LCT

The ROUTE source protocol is based on ALC as defined in RFC 5775 [27] with the following details:

- The Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [26] is used with the following constraints:
  - The Congestion Control Information (CCI) field in the LCT header may be set to 0.
  - The TSI in the LCT header shall be set equal to the value of the **`S-TSID.RS.LS`**`@tsi` attribute.
  - The Codepoint (CP) in the LCT header shall be used to signal the applied formatting as defined in the signaling metadata.

  - The MSB of the PSI shall be set to 1 to indicate a source packet.
  - In accordance to ALC, a source `FEC Payload ID` header is used to identify, for FEC purposes, the encoding symbols of the delivery object, or a portion thereof, carried by the associated ROUTE packet. This information may be sent in several ways:
    - As a simple new null FEC scheme with the following usage:
      - The value of the source `FEC Payload ID` header shall be set to 0, in case the ROUTE packet contains the entire delivery object, or
      - The value of the source `FEC Payload ID` header shall be set as a direct address (start offset) corresponding to the starting byte position of the portion of the object carried in this packet using a 32-bit field.
    - In a compatible manner to RFC 6330 [28] where the SBN and ESI defines the start offset together with the symbol size T.
    - The signaling metadata provides the appropriate parameters to indicate any of the above modes using the `@srcFecPayloadId` attribute.
- The LCT Header `EXT_TIME` extension as defined in RFC 5651 [26] may be used by the sender in the following manner:
  - The Sender Current Time (SCT), depending on the application, may be used to occasionally or frequently to signal the sender current time.
  - The Expected Residual Time (ERT) may be used to indicate the expected remaining time for transmission of the current object.
  - The Sender Last Changed (SLC) flag is typically not utilized, but may be used to indicate addition/removal of Segments.
- Additional extension headers may be used to support real-time delivery. Such extension headers are defined in Section A.3.7.
- The session description information for the ROUTE session and child(ren) LCT channel(s) is provided by the S-TSID fragment.

A.3.5 Packet Format

The packet format used by ROUTE follows the ALC packet format per RFC 5775 [27], i.e. the UDP header followed by the LCT header and the source FEC Payload ID followed by the packet payload. The LCT header shall be as defined in the LCT building block in RFC 5651 [26]. The source FEC Payload ID in ROUTE is represented by the `start_offset` field, provided either directly or provided by any FEC scheme. The `start_offset` field value represents the starting byte position, relative to the first byte of the delivery object, of the subsequent and contiguous portion of the delivery object carried in the present ROUTE packet.

The Congestion Control Information (CCI) field in the LCT header contains the required congestion control information. The packet payload contains data bytes belonging to the delivery object. If more than one object is carried in the session, then the TOI within the LCT header shall be used to identify the object to which the packet payload data pertains.

The version number field of the LCT header shall be interpreted as the ROUTE version number field. This version of ROUTE implicitly makes use of version 1 of the LCT building block defined in RFC 5651 [26].

The overall ROUTE packet format shall be as depicted in Figure A.3.2. The packet is an IP packet, either IPv4 or IPv6, and the IP header precedes the UDP header. The ROUTE packet format has no dependencies on the IP version number.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          UDP Header                           |
|                                                               |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                          LCT Header                           |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        FEC Payload ID                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Payload Data                          |
|                            ...                                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.3.2 Overall ROUTE packet format.

In some special cases a ROUTE sender may need to produce ROUTE packets that do not contain any payload. This may be required, for example, to signal the end of a session or to convey congestion control information. These data-less packets do not contain FEC Payload ID or payload data, but only the LCT header fields. The total datagram length, conveyed by outer protocol headers (e.g., the IP or UDP header), enables receivers to detect the absence of the LCT header, FEC Payload ID and payload data.

A.3.5.1 FEC Payload ID for Source Flows

The syntax of the FEC Payload ID for the Compact No-Code FEC Scheme used in ROUTE source flows shall be a 32-bit unsigned integer value that expresses the `start_offset` of the fragment. Figure A.3.3 diagrams the 32-bit `start_offset` field.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         start_offset                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.3.3 FEC Payload ID for Source Flows

A.3.5.2 FEC Payload ID for Repair Flows

In accordance with RFC 6330 [28] Section 3.2, the FEC Payload ID for the RaptorQ FEC Scheme used for repair flows is composed of a Source Block Number (SBN) and an Encoding Symbol ID, formatted as shown in Figure A.3.4.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     SBN       |               Encoding Symbol ID              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.3.4 FEC Payload ID for Repair Flows

A.3.6 LCT Building Block

The LCT packet header fields shall be used as defined by the LCT building block in RFC 5651 [26]. The semantics and usage of the following LCT header fields shall be further constrained in ROUTE as follows:

Version number (`V`) – This 4-bit field indicates the protocol version number. The version number for this specification is ‘0001’.

Congestion Control flag (`C`) field – This 2-bit field, as defined in RFC 5651 [26], shall be set to ‘00’.

Protocol-Specific Indication (`PSI`) – This 2-bit field indicates whether the current packet is a source packet or an FEC repair packet. As the ROUTE source protocol only delivers source packets, this field shall be set to ‘10’.

Transport Session Identifier flag (`S`) – This 1-bit field shall be set to ‘1’ to indicate a 32-bit word in the `TSI` field.

Transport Object Identifier flag (`O`) – This 2-bit field shall be set to ‘01’ to indicate the number of full 32-bit words in the `TOI` field.

Half-word flag (`H`) – This 1-bit field shall be set to ‘0’ to indicate that no half-word field sizes are used.

`Codepoint` (`CP`) – This 8-bit field is used to indicate the type of the payload that is carried by this packet, and for ROUTE, is defined as shown below in Table A.3.6 to indicate the type of delivery object carried in the payload of the associated ROUTE packet. Depending on the type of the payload, additional payload header(s) may be added to prefix the payload data.

Transport Session Identifier (`TSI`) – This 32-bit field shall identify the Transport Session in ROUTE. The context of the Transport Session is provided by signaling metadata. The `TSI` field is constrained to a length of 32 bits because the Transport Session Identifier flag (`S`) must be set to ‘1’ and the Half-word flag (`H`) must be set to ‘0’.

Transport Object Identifier (`TOI`) – This 32-bit field shall identify the object within this session to which the payload of the current packet belongs. The mapping of the TOI field to the object is provided by the Extended FDT. The `TOI` field is constrained to a length of 32 bits because the Transport Object Identifier flag (`O`) must be set to ‘01’ and the Half-word flag (`H`) must be set to ‘0’.

The main changes that ROUTE introduces to the usage of the LCT building block are the following:

- ROUTE limits the usage of the LCT building block to a single channel per session. Congestion control is thus sender-driven in ROUTE.

The functionality of receiver-driven layered multicast may still be offered by the application, allowing the receiver application to select the appropriate delivery session based on the bandwidth requirement of that session.

Table A.3.6 Defined Values of `Codepoint` Field of LCT Header

Codepoint value (CP)	Semantics	@formatId	@frag	@order
0	ATSC Reserved (not used)			
1	NRT- File Mode	1 (File Mode)	0 (arbitrary)	true
2	NRT – Entity Mode	2 (Entity Mode)	0	true
3	NRT – Unsigned Package Mode	3 (Unsigned Package Mode)	0	true
4	NRT – Signed Package Mode	4 (Signed Package Mode)	0	true
5	New IS, timeline changed	1 (File Mode)	0	true
6	New IS, timeline continued	1	0	true
7	Redundant IS	1	0	true
8	Media Segment, File Mode	1	1 (sample)	true
9	Media Segment, Entity Mode	2 (Entity Mode)	1	true
10 – 127	ATSC Reserved			
128 – 255	Attributes of this type of packet are signalled by attributes given in the SrcFlow.Payload element associated with the CodePoint vlaue	Per Payload element	Per Payload element	Per Payload element

Detailed semantics for each of the defined values of the `Codepoint` (CP) field, which represents the type of delivery object carried in the associated ROUTE packet, shall be as follows:

CP=1: The delivery object is an NRT file or a byte-range portion of such file delivered in ROUTE File Mode, as described by the EFDT element of the source flow delivering this object.

CP=2: The delivery object is an NRT file or a byte-range portion of such file delivered in ROUTE Entity Mode.

CP=3: The delivery object is a file of type `multipart/related` or a byte-range portion of such file delivered in ROUTE Unsigned Package Mode.

CP=4: The delivery object is a file of type `multipart/signed` or a byte-range portion of such file delivered in ROUTE Signed Package Mode.

CP=5: The delivery object is a DASH Initialization Segment (IS) delivered in ROUTE File Mode which differs from the previously delivered IS and also indicates a new presentation timeline.

CP=6: The delivery object is a DASH Initialization Segment delivered in ROUTE File Mode which differs from the previously delivered IS, but maintains the same presentation timeline.

CP=7: The delivery object is a DASH Initialization Segment (IS) delivered in ROUTE File Mode that is identical to the previously delivered IS.

CP=8: The delivery object is a DASH Media Segment delivered in ROUTE File Mode as described by the EFDT of the source flow delivering its parent object flow.

CP=9: The delivery object is a DASH Media Segment delivered in ROUTE Entity Mode as described by the EFDT of the source flow delivering its parent object flow.

CP values from 10 to 127: These CP values are reserved for future ATSC use.

CP values from 128 to 255: These CP values are used for packets in which the corresponding Payload element in the SrcFlow specifies the attributes `@formatId`, `@frag`, and `@order`. For example, a **`SrcFlow.Payload`** element in which `@codePoint` = 128 will specify the `@formatId`, `@frag`, and `@order` attributes of LCT packets with CP=128.

A.3.7 Extension Headers for MDE Mode Operation

A new LCT extension header, `EXT_ROUTE_PRESENTATION_TIME`, is described in Section A.3.7.1. In addition, the `EXT_TIME` header as defined in RFC 5651 [26], and with the SCT-High and SCT-Low flags set is present when the `EXT_ROUTE_PRESENTATION_TIME` header is present.

A.3.7.1 EXT_ROUTE_PRESENTATION_TIME Header

A ROUTE presentation time header may be added to an LCT packet. The `EXT_ROUTE_PRESENTATION_TIME` LCT extension header shall be present in the first LCT packet containing an MDE data block with Random Access Point (RAP). No other types of LCT packets shall include the `EXT_ROUTE_PRESENTATION_TIME` LCT extension header. Thus the presence of an `EXT_ROUTE_PRESENTATION_TIME` header indicates that MDE mode is supported in the delivery of the media stream by the associated ROUTE packets.

The format of the `EXT_ROUTE_PRESENTATION_TIME` header shall be as shown in Figure A.3.5. Header Extension Type (HET) value 66 has been registered with IANA for this header. If present, the `EXT_ROUTE_PRESENTATION_TIME` Header expresses the full 64-bit NTP timestamp value. The value must always be greater than the SCT.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET = 66     |      HEL      |       reserved                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          NTP timestamp, most significant word                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          NTP timestamp, least significant word                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.3.5 12-byte `EXT_ROUTE_PRESENTATION_TIME` header.

A.3.7.2 `EXT_TIME` Header

LCT packets containing the `EXT_ROUTE_PRESENTATION_TIME` header shall also include the `EXT_TIME` header as defined in RFC 5651 [26]. The `SCT-High` and `SCT-Low` flags shall both be set and the corresponding time value shall denote the SCT value in the "wait time" associated with earliest MDE ROUTE presentation time, i.e., [`EXT_ROUTE_PRESENTATION_TIME` - SCT], as described in Section 8.1.1.3.

A.3.8 Basic ROUTE Sender Operation

The following description of the ROUTE sender operation on mapping of the delivery object to the ROUTE packet payloads logistics represents an extension of RFC 5445 [25], which in turn inherits the context, language, declarations and restrictions of the FEC building block in RFC 5052 [21]. The data carried in the payload of a given ROUTE packet constitute a contiguous portion of the delivery object. ROUTE source delivery can be considered as a special case of the use of the Compact No-Code Scheme associated with FEC Encoding ID = 0 according to Sections 3.4.1 and 3.4.2 of RFC 5445 [25], in which the encoding symbol size is exactly one byte. For ROUTE source flows, the FEC Payload ID shall deliver the 32-bit `start_offset` as specified in Section A.3.5.1 above. All receivers are expected to support, at minimum, operation with this special case of the Compact No-Code FEC.

Note that in the event the source object size is greater than 2^32 bytes (approximately 4.3 GB), the applications (in the broadcaster server and the receiver) are expected to perform segmentation/re-assembly using methods beyond the scope of this transport standard.

In the basic operation, it is assumed that:

- A delivery object is fully available at the ROUTE sender
- T > 0 represents the `Transfer-Length` of the object in bytes
- The source FEC Payload ID comprises the `start_offset` field

The maximum transfer unit of the first data packet is known as Y. The transfer unit is determined either by knowledge of underlying transport block sizes or by other constraints. In case Y is greater than or equal to T, this is the only packet for this delivery object.

If Y is smaller than T, then the data carried in the payload of a packet consists of a contiguous portion of the object. For any arbitrary X and any arbitrary Y > 0 as long as X + Y is at most T, a ROUTE packet may be generated. In this case the following shall hold:

1) The data carried in the payload of a packet shall consist of a contiguous portion of the object starting from the beginning of byte X through the beginning of byte X + Y.
2) The `start_offset` field in the ROUTE packet header shall be set to X and the payload data shall be added into the packet to send.

The order of packet delivery is arbitrary, but in the absence of other constraints delivery with increasing `start_offset` value is recommended. Certain applications may require a strict sending order with increasing `start_offset` value for each subsequent ROUTE packet.

In other cases, the transfer length may be unknown prior to sending earlier pieces of the data, and only a maximum transfer length is known as signaled in the Extended FDT parameter `@maxTransportSize`. In this case, T may be determined later. However, this does not affect the sending process above. Additional packets may be sent following the above rules 1 to 3.

A.3.9 Basic ROUTE Receiver Operation

A.3.9.1 Overview

Figure A.3.6 illustrates the basic receiver operation. The receiver receives packets and filters these packets accordingly. From the ROUTE session and each contained LCT channel it regenerates delivery objects from the ROUTE session and each contained LCT channel. The delivery objects are in turn passed to the appropriate handler for further data processing.

The basic receiver information is provided below. In addition, the treatment in case of different received objects is described in this section.

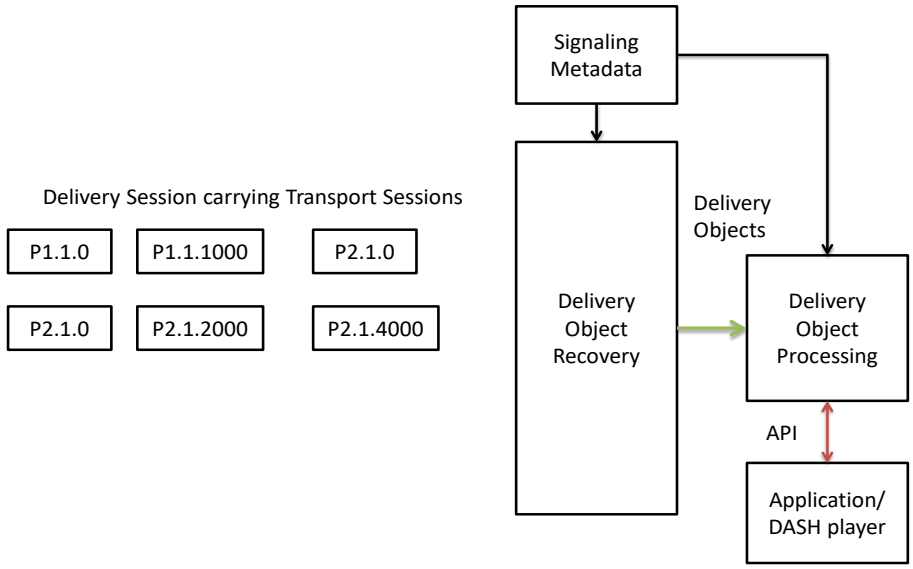


Figure A.3.6 Receiver operation.

A.3.9.2 Basic Delivery Object Recovery

Signaling metadata contains information that describes the carried object flows. Upon receipt of each payload, the receiver proceeds with the following steps in the order listed.

1) The ROUTE receiver is expected to parse the LCT and FEC Payload ID to verify that it is a valid header. If it is not valid, then the payload is discarded without further processing.
2) The ROUTE receiver is expected to assert that the TSI and the `CodePoint` represent valid operation points in the signaling metadata, i.e. the signaling contains a matching entry to the TSI value provided in the packet header, as well as for this TSI, the signaling contains the attribute **`PayloadFormat`**`@codePoint` whose value is identical to the `CodePoint` field in the LCT header.
3) The ROUTE receiver should process the remainder of the payload, including the appropriate interpretation of the other payload header fields, and using the source FEC Payload ID (to determine the `start_offset`) and the payload data to reconstruct the corresponding object as follows:
   a) The ROUTE receiver can determine the object associated with the received ROUTE packet payload via the signaling metadata and the TOI carried in the LCT header.
   b) Upon receipt of the first ROUTE packet payload for an object, the ROUTE receiver uses the **`File`**`@Transfer-Length` attribute of the associated Extended FDT Instance, if present, to determine the length T of the object. If the **`File`**`@Transfer-Length` attribute is not present in the EFDT, the receiver uses the `@maxTransportSize` attribute of the associated EFDT Instance to determine the maximum length T' of the object.
   c) The ROUTE receiver allocates buffer space for the T or T' bytes that the object will or may occupy.
   d) The ROUTE receiver also computes the length of the payload, Y, by subtracting the payload header length from the total length of the received payload.
   e) The ROUTE receiver allocates a Boolean array RECEIVED[0..T-1] or RECEIVED[0..T'-1], as appropriate, with all entries initialized to false to track

received object symbols. The ROUTE receiver continuously acquires packet payloads for the object as long as all of the following conditions are satisfied: i) there is at least one entry in RECEIVED still set to false; ii) the object has not yet expired; and iii) the application has not given up on reception of this object. More details are provided below.

f) For each received ROUTE packet payload for the object (including the first payload), the steps to be taken to help recover the object are as follows:
   i. If the packet includes an `EXT_TOL` or `EXT_FTI` header, modify the Boolean array RECEIVED[0..T'-1] to become RECEIVED[0..T-1].
   ii. Let X be the value of the `start_offset` field in in the ROUTE packet header and let Y be the length of the payload, Y, computed by subtracting the LCT header size and the FEC Payload ID size from the total length of the received packet.
   iii. The ROUTE receiver copies the data into the appropriate place within the space reserved for the object and sets RECEIVED[X ... X+Y-1] = true.
   iv. If all T entries of RECEIVED are true, then the receiver has recovered the entire object.

Upon recovery of both the complete set of packet payloads for the delivery object associated with a given TOI value, and the metadata for that delivery object, the object is handed to the application.

Metadata recovery depends on the applied delivery mode.

A.3.9.3 General Metadata Recovery

Typically, delivery objects are only handed up to the application if they are complete and intact.

However, in certain circumstances a partially received object may be handed up, if the application API permits this and assuming that sufficient metadata is available to enable the application to handle the partial object. One defined mechanism for this is described in Section 7.2.3 of ETSI TR 126.946 [46].

If an object is received in the file mode, then the Extended FDT is used to recover all object metadata. For details on how to use this, refer to Section A.3.3.2.6.

If an object is received in the entity mode, then the entity header and the entity body are treated according to RFC 7231 [31].

A.3.9.4 Packaged Mode Reception

If a delivery object is received in packaged mode, then the receiver is expected to unpack the files prior to handing them to the application along with the metadata that is included in the package.

A.4 REPAIR PROTOCOL SPECIFICATION

A.4.1 Introduction

The delivery objects and bundles of delivery objects delivered by the source protocol as described in Section A.3 may be protected with FEC. The base protocol as defined in Section A.3 avoids including any FEC-specific signaling.

In this section an FEC framework is defined that enables FEC protection of individual or bundles of delivery objects associated with the source protocol as defined in Section A.3.

The FEC framework uses concepts of the FECFRAME work as defined in RFC 6363 [50], as well as the FEC building block, RFC 5052 [21], which is adopted in the existing FLUTE/ALC/LCT specifications.

The FEC design adheres to the following principles:

- FEC-related information is provided only where needed.
- Receivers not capable of this framework can ignore repair packets.
- The FEC is symbol-based with fixed symbol size per protected repair flow. The ALC protocol and existing FEC schemes are reused.
- A FEC repair flow provides protection of delivery objects from one or more source flows.

The FEC-specific components of the FEC framework are:

- FEC repair flow declaration including all FEC specific information.
- FEC transport object that is the concatenation of a delivery object, padding octets and size information in order to form a symbol-aligned chunk of data.
- FEC super-object that is the concatenation of one or more FEC transport objects in order to bundle FEC transport objects for FEC protection.
- FEC protocol and packet structure.

A receiver needs to be able to recover delivery objects from repair packets based on available FEC information.

A.4.2 FEC Protocol

A.4.2.1 Introduction

This section specifies the protocol elements for the FEC Framework. Four components of the protocol are defined in this document and are described in the following sections:

1) FEC transport object construction
2) FEC super-object construction as the concatenation of FEC transport objects
3) TOI mapping
4) Repair packet generation

The operation of the FEC Framework is governed by repair flow definition as defined in Section A.4.3.

Figure A.4.1 shows the FEC packet generation based on two delivery objects.

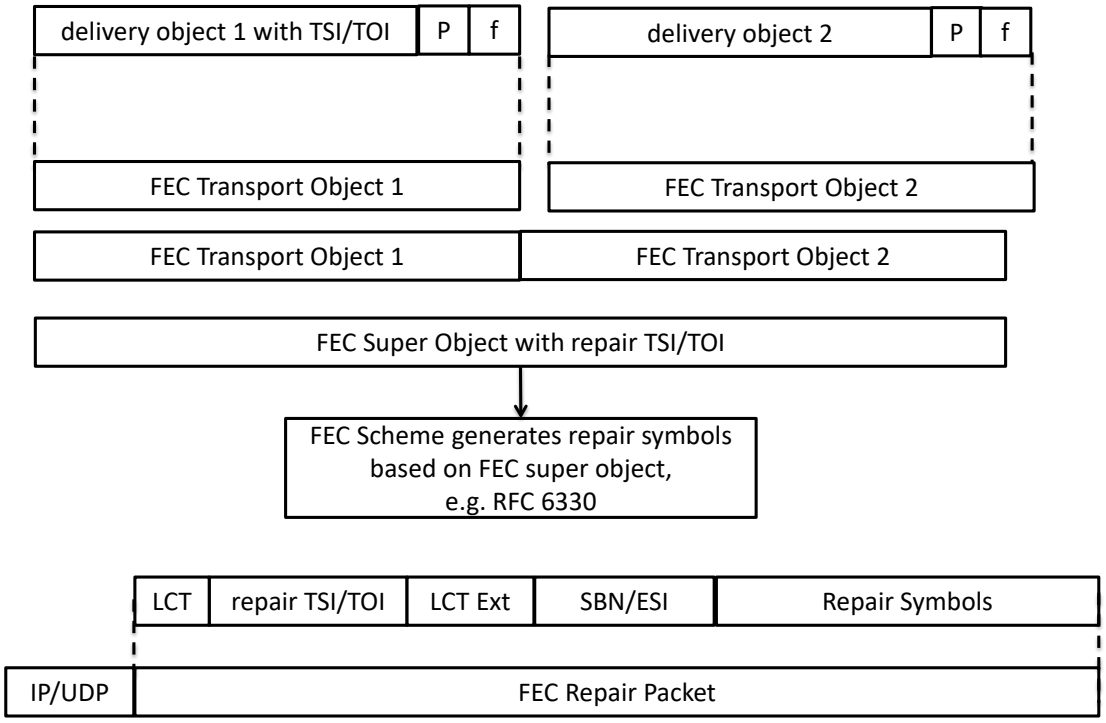


Figure A.4.1 FEC packet generation.

A.4.2.2 FEC Transport Object Construction

In order to identify a delivery object in the context of the Repair protocol, the following information is needed:

- TSI and TOI of the delivery object. In this case, the FEC object corresponds to the (entire) delivery object.
- Octet range of the delivery object, i.e. start offset within the delivery object and number of subsequent and contiguous octets of delivery object that constitutes the FEC object (i.e., the FEC-protected portion of the source object). In this case, the FEC object corresponds to a contiguous byte range portion of the delivery object.

Typically, the first mapping is applied; i.e., the delivery object is an FEC object.

Assuming that the FEC object is the delivery object, for each delivery object, the associated FEC transport object is comprised of the concatenation of the delivery object, padding octets (P) and the FEC object size (F) in octets, where F is carried in a 4-octet field. The FEC transport object size S, in FEC encoding symbols, shall be an integer multiple of the symbol size Y.

S is determined from the session information and/or the repair packet headers.

F is carried in the last 4 octets of the FEC transport object.

Specifically, let

- F be the size of the delivery object in octets,
- **F** be the F octets of data of the delivery object,
- **f** denotes the four octets of data carrying the value of F in network octet order (high order octet first),

- S be the size of the FEC transport object with S=ceil((F+4)/Y), where the ceil() function rounds the result upward to its nearest integer,
- **P** be S*Y-4-F octets of data, i.e. padding placed between the delivery object and the 4-byte field conveying the value of F and located at the end of the FEC transport object, and
- **O** be the concatenation of **F**, **P** and **f** .

O then constitutes the FEC transport object of size S*Y octets. Note that padding octets and the object size F are NOT sent in source packets of the delivery object, but are only part of an FEC transport object that FEC decoding recovers in order to extract the FEC object and thus the delivery object or portion of the delivery object that constitutes the FEC object. In the above context, the FEC transport object size in symbols is S.

Figure A.4.2 shows an example with S=3.

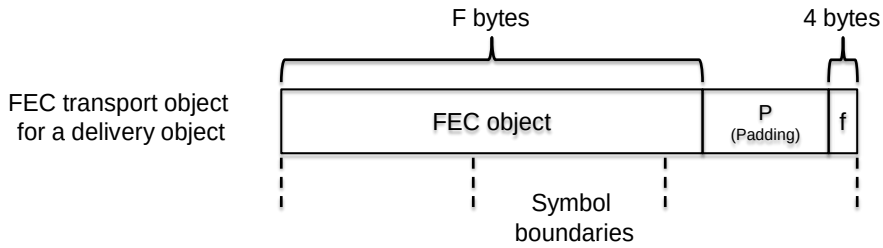


Figure A.4.2 FEC transport object construction (example with S = 3).

The general information about an FEC transport object that is conveyed to an FEC enabled receiver is the source TSI, source TOI and the associated octet range within the delivery object comprising the associated FEC object. However, as the size in octets of the FEC object is provided in the appended field within the FEC transport object, the remaining information can be conveyed as:

- TSI and TOI of the delivery object from which the FEC object associated with the FEC transport object is generated
- Start octet within delivery object for the associated FEC object
- Size in symbols of the FEC transport object

A.4.2.3 Super-Object and FEC repair for delivery objects

From the FEC repair flow declaration, the construction of an FEC super-object as the concatenation of one or more FEC transport objects can be determined. The FEC super-object includes the general information about the FEC transport objects as described in the previous subsection, as well as the placement order of FEC transport objects within the FEC super-object.

Let

- N be the total number of FEC transport objects for the FEC super-object construction.
- For i = 0,…,N-1, let S[i] be the size in symbols of FEC transport object i.
- **B** be the FEC super-object which is the concatenation of the FEC transport objects in numerical order, comprised of K = $\sum_{i=0}^{N-1} S[i]$ source symbols.

For each FEC super-object, the remaining general information that needs to be conveyed to an FEC enabled receiver, beyond what is already carried in the FEC transport objects that constitute the FEC super-object, comprises:

- The total number of FEC transport objects N.
- For each FEC transport object, the
  - TSI and TOI of the delivery object from which the FEC object associated with the FEC transport object is generated,
  - start octet within delivery object for the associated FEC object, and
  - size in symbols of the FEC transport object.

The carriage of the FEC repair information is discussed below.

A.4.2.4 Repair Packet Structure

The repair protocol is based on Asynchronous Layered Coding (ALC) as defined RFC 5775 [27] and the Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [26] with the following details:

- The Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [26] is used as defined in Asynchronous Layered Coding (ALC), Section 2.1. In addition, the following constraints apply:
  - The TSI in the LCT header shall identify the repair flow to which this packet applies, by the matching value of the `@tsi` attribute in the signaling metadata among the LCT channels carrying repair flows.
  - The first bit of the Protocol Specific Indication (`PSI` bit `X`), the Source Packet Indicator (`SPI`), shall be set to ‘0’ to indicate a repair packet.
  - FEC configuration information may be carried in LCT extension headers.
- The FEC building block is used according to RFC 6330 [28], but only repair packets are delivered.
  - Each repair packet within the scope of the repair flow (as indicated by the TSI field in the LCT header) shall carry the appropriate repair object/super-object TOI. The repair object/super-object TOI shall be unique for each FEC super-object that is created within the scope of the TSI.

A.4.2.5 Summary FEC Information

For each super-object (identified by a unique TOI within a repair flow that is in turn identified by the TSI in the LCT header) that is generated, the following information needs to be communicated to the receiver:

- The FEC configuration consisting of
  - FEC Object Transmission Information (OTI) per RFC 5052 [21].
  - Additional FEC information (see Table A.4.1).
- The total number of FEC objects included in the FEC super-object.
- For each FEC transport object,
  - TSI and TOI of the delivery object used to generate the FEC object associated with the FEC transport object,
  - Start octet within the delivery object of the associated FEC object, if applicable, and
  - The size in symbols of the FEC transport object.

The above information may be delivered:

- Statically in the `RepairFlow` declaration as defined in Section A.4.3,
- Dynamically in an LCT extension header, or
- Via a combination of the above two methods.

A.4.2.6 Extension Headers

The ROUTE receiver uses either the `EXT_FTI` per RFC 5775 [27] or the `EXT_TOL` header specified below to extract the transfer length of the object.

A.4.2.6.1 `EXT_TOL` Header

The `EXT_TOL` header is used to signal the Transport Object Length (TOL) of the File. The `EXT_TOL24` and the `EXT_TOL48` header formats are shown in Figure A.4.3. `EXT_TOL` is an LCT extension header that denotes the transport object length as either a 24-bit or 48-bit unsigned integer value.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET = 194    |             Transfer Length                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  HET = 67     |      HEL      |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               +
|                       Transfer Length                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.4.3 `EXT_TOL` Header (24-bit version shown on top and 48-bit version shown on bottom).

A.4.3 Repair Flow Declaration

A.4.3.1 General

In the delivery of ATSC 3.0 services, a repair flow declaration may be included.

As part of the repair flow declaration, a repair flow identifier is provided for the repair flow in the `@tsi` attribute. The combination of the IP address, the port number and the repair flow identifier shall provide a unique identifier among all flows within a User Service. Note that a destination IP address/port number combination may carry multiple FEC repair flows as well as source flows, in which case each flow is identified by a unique TSI value in the LCT header.

The repair flow declaration indicates the pattern of the delivery objects (or octet ranges of delivery object) from source flows that are to be protected by the repair flow.

A.4.3.2 Semantics

The **`RepairFlow`** element is a child element of **`S-TSID.RS.LS`** (see Section 7.1.4).

While the XML schema specifies the normative syntax of the **`RepairFlow`** element, informative Table A.4.1 below describes the structure of the **`RepairFlow`** element in a more illustrative way.

The semantics of the elements and attributes of **`RepairFlow`** shall be as given in Table A.4.1.

Table A.4.1 Semantics of `RepairFlow` Element

Element or Attribute Name	Use	Data type	Description
`RepairFlow`		stsid:rprFlowType	Repair flow carried in the LCT channel.
`FECParameters`	0..1		FEC Parameters corresponding to the repair flow
`@maximumDelay`	0..1	unsignedInt	Maximum delivery delay between any source packet in the source flow and the repair flow.
`@overhead`	0..1	unsignedShort (percentage)	FEC overhead in a ROUTE packet represented as a percentage
`@minBuffSize`	0..1	unsignedInt	Required buffer size to handle all associated objects that are assigned to a super-object.
`@fecOTI`	1	hexBinary	FEC related information associated with an object as well as FEC information associated with the encoding symbols of the object and is to be included within this declaration and applies to all repair packets with the repair flow.
`ProtectedObject`	0..N		List of the source flow(s) protected by this repair flow and the details on how the protection is done. It also defines how certain delivery objects of a collection of objects are included in the repair flow.
`@sessionDescription`	0..1	string	Specifies the session description information for the source flow.
`@tsi`	1	unsignedInt	The Transport Session Identifier (TSI) for the source flow to be protected.
`SourceTOI`	1..N		The Transport Object Identifier (TOI) value(s) of the delivery object(s) corresponding to the TOI of a given FEC (super-)object delivered by the repair flow.
`@x`	0..1	unsignedShort	Value of the constant X for use in deriving the TOI of the delivery object from the TOI of the FEC (super-)object.
`@y`	0..1	unsignedShort	Value of the constant Y for use in deriving the TOI of the delivery object from the TOI of the FEC (super-)object.

`RepairFlow` – A complex element whose subordinate elements and attributes shall contain information about each of the one or more repair flows carried in the LCT channel, which may be associated with Source Flows, and referenced by other service signaling metadata.

`FECParameters` – A complex element whose subordinate elements and attributes contain information pertaining to AL-FEC parameters that apply to this repair flow.

`@maximumDelay` – A 32-bit unsigned integer whose value, if present, shall represent the maximum delivery delay, in milliseconds, between any source packet in the source flow and the repair packet, associated with that source packet, in the repair flow. Default semantics of this attribute, when absent, is not defined.

`@overhead` – A 16-bit unsigned integer whose value shall represent the sum of the AL-FEC related fields in the ROUTE repair packet relative to the size of the repair packet size as a percentage. The AL-FEC related parameters shall comprise the attributes `@maximumDelay`, `@overhead`, and `@minBufferSize`, and the `@fecOTI` attribute under the ele **`RepairFlow.FECParameters`**. Absence of this attribute means that the AL-FEC related overhead information is not provided.

`@minBuffSize` – A 32-bit unsigned integer whose value shall represent a required size of the receiver transport buffer for AL-FEC decoding processing. If present, then this attribute shall

indicate the minimum buffer size that is required to handle all associated objects that are assigned to a super-object i.e. a delivery object formed by the concatenation of multiple FEC transport objects in order to bundle these FEC transport objects for AL-FEC protection. Absence of this attribute means that the required minimum buffer size is unknown.

`@fecOTI` – A parameter formatted as 12 octets in xs:hexBinary format consisting of the concatenation of Common and Scheme-Specific FEC Object Transmission Information (FEC OTI) as defined in Sections 3.3.2 and 3.3.3 of RFC 6330 [28], and which correspond to the delivery objects carried in the source flow to which this repair flow is associated, with the following qualification. The 40-bit `Transfer Length (F)` field may either represent the actual size of the object, or it is encoded as all zeroes. In the latter case, it means that the FEC transport object size is either unknown, or cannot be represented by this attribute. In other words, for the all-zeroes format, the delivery objects in the Repair flow correspond to streaming content – either a live service whereby content encoding has not yet occurred at the time this S-TSID instance was generated, or pre-recorded streaming content whose delivery object sizes, albeit known at the time of S-TSID generation, are variable and cannot be represented as a single value by the `@fecOTI` attribute.

`ProtectedObject` – A complex element whose attributes contain information about the source flow(s) protected by this repair flow, and the details on how the protection is performed. It also defines how certain delivery objects of a collection of objects are included in the repair flow.

`@sessionDescription` – A string whose value, in the form of comma-separated attribute-value pairs, shall represent session description information for the source flow, if applicable, protected by the repair flow. The set of attribute/value pairs contained in this attribute shall be identical to the corresponding session description information as specified in the S-TSID fragment. These might include, for example, **`S-TSID.RS`**`@sIpAddr`, **`S-TSID.RS`**`@dIpAddr`, **`S-TSID.RS`**`@dPort`, **`S-TSID.RS.LS`**`@tsi`, **`S-TSID.RS.LS`**`@bw`, **`S-TSID.RS.LS`**`@startTime`, **`S-TSID.RS.LS`**`@endTime` (list is not intended to be comprehensive). If `@sessionDescription` is not present, it shall imply that the source flow is carried in the same ROUTE session/LCT channel as the repair flow**.**

`@tsi` – A 32-bit unsigned integer whose value shall represent the Transport Session Identifier (TSI) of the source flow protected by this repair flow.

`SourceTOI` – This element shall convey the TOI value(s) of the one or more source objects protected by a given FEC transport object or FEC super-object. Its attributes `@x` and `@y`, if present, shall represent the constants X and Y, respectively, for use in deriving each constituent **`SourceTOI`** value from the TOI of a corresponding FEC (super-)object according to the equation as shown in Section A.4.3.3, i.e., [`sourceTOI = X*TOI + Y`].

`@x` – The value of the constant X, if present, for use in deriving the parent **`SourceTOI`** value from the above equation. The default value shall be `"1"`.

`@y` – The value(s) of the constant Y, if present, for use in deriving the parent **`SourceTOI`** value from the above equation. The default value shall be `"0"`.

Note that if neither `@x` nor `@y` is present there is a 1:1 relationship between each delivery object carried in the source flow as identified by **`ProtectedObject`**`@tsi` to an FEC object carried in this repair flow. In this case the TOI of each of those delivery objects shall be identical to the TOI of the corresponding FEC object.

Note that the TSI of the LCT channel carrying the repair flow is not included as an attribute of **`RepairFlow`** element because that parameter is already provided by the **`S-TSID.RS.LS`**`@tsi` associated with the repair flow.

A.4.3.3 TOI Mapping

If the repair flow declaration contains a **`ProtectedObject`** element, then the `@sourceTOI` attribute shall specify a mapping of the repair TOI value contained in a repair packet to a source TOI of a delivery object that the repair packet protects.

The mapping shall be described through an equation in C Language format, in the form [`sourceTOI = X*TOI + Y`]; whereby the TOI field of the repair flow is specified as the variable `TOI`, `X` and `Y` represent 16-bit unsigned integer values, and the result of the equation determines the source TOI value. The mapping shall be represented by a proper C equation with at most one variable TOI and without the addition of the ";" sign. If no `@sourceTOI` attribute is provided in the **`ProtectedObject`** then the default equation `sourceTOI="TOI"` shall be applied to derive the source TOI from the repair TOI.

A.4.3.4 Examples for **`RepairFlow`** Declarations

A.4.3.4.1 Example 1

In Example 1, a repair packet with TSI value 10 and TOI 20 protects the delivery object from the source flow with TSI 1 and TOI 20. The FEC transport object size is defined as 892 symbols. The repair flow declaration is provided in Figure A.4.4.

```
<RepairFlow>
    <FECParameters maximumDelay="5000" overhead="50" minBuffSize="20000000"
fecOTI="f0f1f2f3f4f5f6f7f8f9fafb">
    <ProtectedObject
sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.54,dport=1234,bw=3500000,
startTime=32159980800,endTime=32162572800,srcFecPayloadId=177" tsi="1" sourceTOI="20"
fecTransportObjectSize="892"/>
    </FECParameters>
</RepairFlow>
```

Figure A.4.4 `RepairFlow` example #1.

A.4.3.4.2 Example 2

In Example 2, since no `@sourceTOI` attribute is provided in the **`ProtectedObject`** then the default equation `sourceTOI="TOI"` is used to derive the source TOI from the TOI. (See Figure A.4.5.) Thus, the super-object that is protected by a repair packet with TSI 11 and TOI 20 is the concatenation of the FEC transport object for the delivery object with TSI 2 and TOI 20 and the FEC transport object for the delivery object with TSI 3 and TOI 20. The FEC transport object size in both cases is defined and the FEC super-object has a total size of 2232 symbols.

All three flows are delivered in the same session.

```
<RepairFlow>
    <FECParameters maximumDelay="5000" overhead="50" minBuffSize="20000000"
fecOTI="f0f1f2f3f4f5f6f7f8f9fafb">
    <ProtectedObject sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.54,dport=1234,
bw=3500000,startTime=32159980800,endTime=32162572800,srcFecPayloadId=177"
tsi="2" fecTransportObjectSize="892"/>
    <ProtectedObject sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.55,dport=1234,
        bw=128000,startTime=32159980800,endTime=32162572800,srcFecPayloadId=39"
        tsi="3" fecTransportObjectSize="1340"/>
    </FECParameters>
</RepairFlow>
```

Figure A.4.5 `RepairFlow` example #2.

A.4.3.4.3 Example 3

In Example 3, the FEC super-object that is protected by a repair packet with TSI 12 and TOI 20 is the concatenation of the FEC transport object for the delivery object with TSI 4 and TOI 40, the FEC transport object for the delivery object with TSI 5 and TOI 20 and the FEC transport object for the delivery object with TSI 4 and TOI 41. (See Figure A.4.6.) In this example, the FEC transport object sizes of the delivery objects are not included in the repair flow declaration.

Note that the sequence of the `ProtectedObject` declarations within a `RepairFlow` declaration determines the order of the FEC transport objects for the delivery objects in the super-object.

```
<RepairFlow>
    <FECParameters maximumDelay="5000" overhead="50" minBuffSize="20000000"
fecOTI="f0f1f2f3f4f5f6f7f8f9fafb">
    <ProtectedObject sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.54,dport=1234,
        bw=3500000,startTime=32159980800,endTime=32162572800,srcFecPayloadId=177" tsi="4"
sourceTOI="40"/>
    <ProtectedObject sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.55,dport=1234,
        bw=128000,startTime=32159980800, endTime=32162572800,srcFecPayloadId=39" tsi="5"
sourceTOI="20"/>
    <ProtectedObject sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.56,dport=1234,
        bw=1600,startTime=32159980800,endTime=32162572800,srcFecPayloadId=18" tsi="4"
sourceTOI="41"/>
    </FECParameters>
</RepairFlow>
```

Figure A.4.6 `RepairFlow` example #3.

A.4.3.4.4 Example 4

In Example 4, a repair packet with TSI 13 and TOI 20 protects the delivery object with TSI 6 and TOI 21. (See Figure A.4.7.)

```
<RepairFlow>
    <FECParameters maximumDelay="5000" overhead="50" minBuffSize="20000000"
fecOTI="f0f1f2f3f4f5f6f7f8f9fafb">
    <ProtectedObject sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.54,dport=1234,
        bw=3500000,startTime=32159980800,endTime=32162572800,srcFecPayloadId=177" tsi="6"
    sourceTOI="21" fecTransportObjectSize="892"/>
    </FECParameters>
</RepairFlow>
```

Figure A.4.7 `RepairFlow` example #4.

A.4.3.4.5 Example 5

In Example 5, a repair packet with TSI 14 and TOI 10 protects the concatenation of the FEC transport objects for two delivery objects: the delivery object with TSI 7 and TOI 20 and the delivery object with TSI 7 and TOI 21. (See Figure A.4.8.) The following repair flow declaration with TSI 14 provides the static FDD.

```
<RepairFlow>
    <FECParameters maximumDelay="5000" overhead="50" minBuffSize="20000000"
fecOTI="f0f1f2f3f4f5f6f7f8f9fafb">
        <ProtectedObject sessionDescription="http://example.com/session7.xml" tsi="7"
sourceTOI="20"/>
        <ProtectedObject sessionDescription="http://example.com/session7.xml" tsi="7"
sourceTOI="21"/>
    </FECParameters>
</RepairFlow>
```

Figure A.4.8 `RepairFlow` example #5.

The ordering of the FEC transport objects in the FEC super-object is the same as the order in which the `ProtectedObject` declarations are made in the `RepairFlow` declaration. Thus, in example 5, the FEC super-object corresponding to a repair packet with TSI 14 and TOI 10 is the concatenation of the FEC transport object for the delivery object with TSI 7 and TOI 20 and the FEC transport object for the delivery object with TSI 7 and TOI 21.

A.4.4 Receiver Operation

A.4.4.1 Introduction

For robust reception, FEC may be applied. In this case, one or more repair flows are sent along with the source packet streams. The repair flow description is provided in Section A.4.3.2.

The crucial aspect of receiver operation in a ROUTE environment is the recovery of the delivery objects from incoming LCT packets along with additional information provided in the signaling that describes the source and repair packet flows. The specific procedure depends on the implementation option, for example:

- No FEC is employed and the delivery object is to be recovered directly from the source packets.

- FEC is provided individually for each delivery object, i.e. no use of FEC super-objects as described in Section A.4.2.3.
- Single FEC scheme is used to protect a multitude of delivery objects, i.e. use of FEC super-objects.
- Multiple FEC schemes are used to protect different sets of delivery objects.

In addition, it may be up to the receiver to decide to use zero, one or more of the FEC streams. Hence, each flow is typically assigned an individual recovery property, which defines aspects such as the delay and the required memory if one or the other is chosen. The receiver may select one or more repair flows, depending on its mode of operation. The receiver may decide whether or not to utilize repair flows based on the following considerations:

- The desired start-up and end-to-end latency. If a repair flow requires significant amount of buffering time to be effective, such repair flow might only be used in time-shift operations or in poor reception conditions, since use of such repair flow trades off end-to-end latency against DASH Media Presentation quality.
- FEC capabilities, i.e. the receiver may pick only the FEC algorithm that it supports.
- Which source flows are being protected; for example, if the repair flow protects source flows that are not selected by the receiver, then it may not select the repair flow.
- Other considerations such as available buffer size, reception conditions, etc.

If a receiver decides to acquire a certain repair flow then it must receive data on all source flows that are protected by that repair flow to collect the relevant packets.

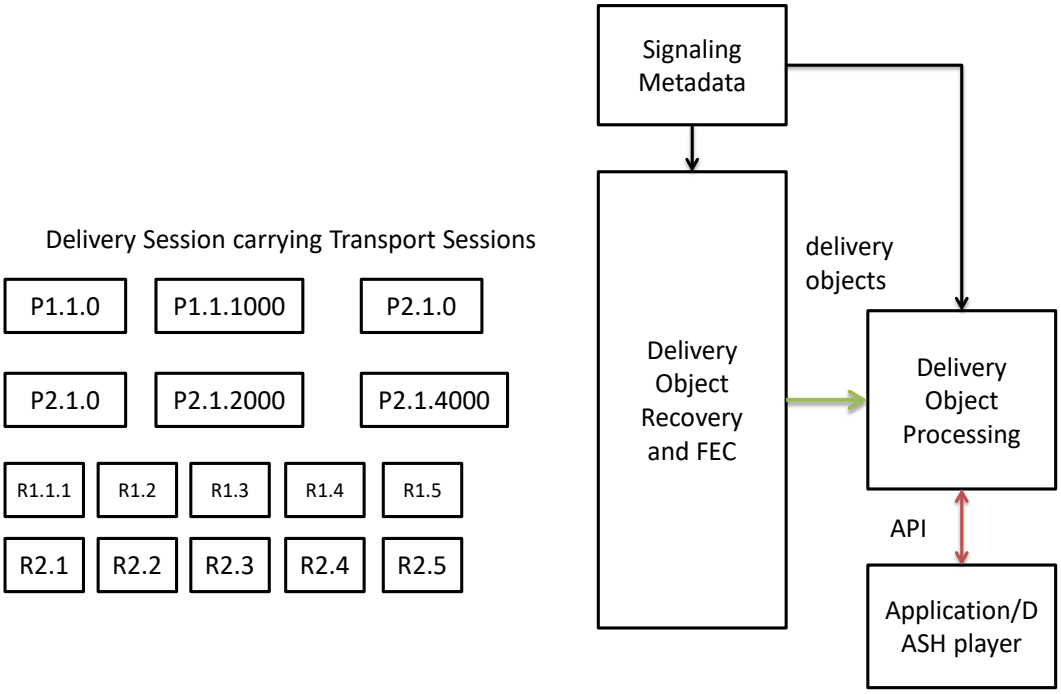


Figure A.4.9 ROUTE receiver with FEC.

A.4.5 Example Operation

To be able to recover the delivery objects that are protected by a repair flow, a receiver needs to obtain the necessary service signaling metadata fragments that describe the corresponding

collection of delivery objects that are covered by this repair flow. A repair flow is defined by the combination of an LCT channel, a unique TSI number, as well as the corresponding protected source flows.

If a receiver acquires data of a repair flow, it is expected to collect all packets of all protected Transport Sessions.

Upon receipt of each packet, whether it's a source or repair packet, the receiver proceeds with the following steps in the order listed.

1) The receiver is expected to parse the packet header and verify that it is a valid header. If it is not valid, then the packet shall be discarded without further processing.
2) The receiver is expected to parse the TSI field of the packet header and verify that a matching value exists in the service signaling for the repair flow or the associated Protected source flow. If no match is found, the packet shall be discarded without further processing.
3) The receiver processes the remainder of the packet, including interpretation of the other header fields, and using the source FEC Payload ID (to determine the start_offset byte position within the source object), the Repair FEC Payload ID, as well as the payload data, reconstructs the decoding blocks corresponding to a FEC super-object as follows:
   a) For a source packet, the receiver identifies the delivery object to which the received packet is associated, using the session information and the TOI carried in the payload header. Similarly, for a repair object the receiver identifies the FEC super-object to which the received packet is associated, using the session information and the TOI carried in the payload header.
   b) For source packets, the receiver collects the data for each FEC super-object and recovers FEC super-objects in same way as in Section A.3.9.2. The received FEC super-object is then mapped to a source block and the corresponding encoding symbols are generated.
   c) With the reception of the repair packets, the FEC super-object can be recovered.
   d) Once the FEC super-object is recovered, the individual delivery objects can be extracted.

A.4.6 Repair Protocol

If FEC is included and `@minBufferTime` of the repair flow declaration is present, then `@minBufferTime` shall convey the minimum buffer time for the repair flow.

A.5 SECURITY CONSIDERATIONS

Refer to ALC as defined in RFC 5775 [27].

Annex B: Signaling Instance Examples

B.1 HIERARCHY SIGNALING STRUCTURE

The diagram in Figure B.1.1 illustrates the delivery of two S-TSID instances over ROUTE. One S-TSID provides access information for the LCT channels, belonging to ROUTE session #1, which carries the content components of Service_X. The second S-TSID provides access information for the LCT channel(s) belonging to ROUTE session #N, which carries the content components of Service_Y. This example depicts the delivery of the SLT, as Low Level Signaling, via UDP/IP encapsulation and carried as a Link Layer packet over PLP#0 (pre-configured as the most robust PLP).

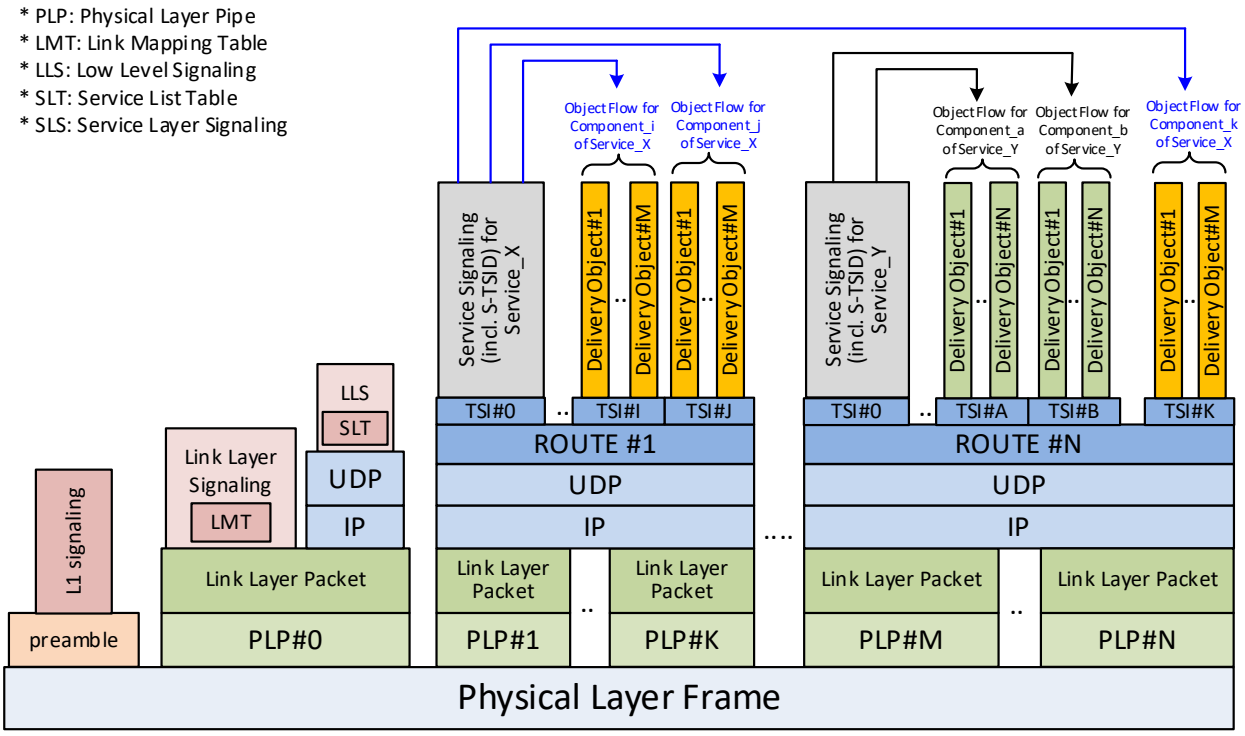


Figure B.1.1 ATSC 3.0 ROUTE hierarchical signaling architecture.

The diagram in Figure B.1.2 illustrates the delivery of two MP tables over MMTP. One MP table provides access information for the MMTP packet flows, belonging to MMTP session #1, which carries the content components of Service_X. The second MP table provides access information for the MMTP packet flows belonging to MMTP session #N, which carries the content components of Service_Y.

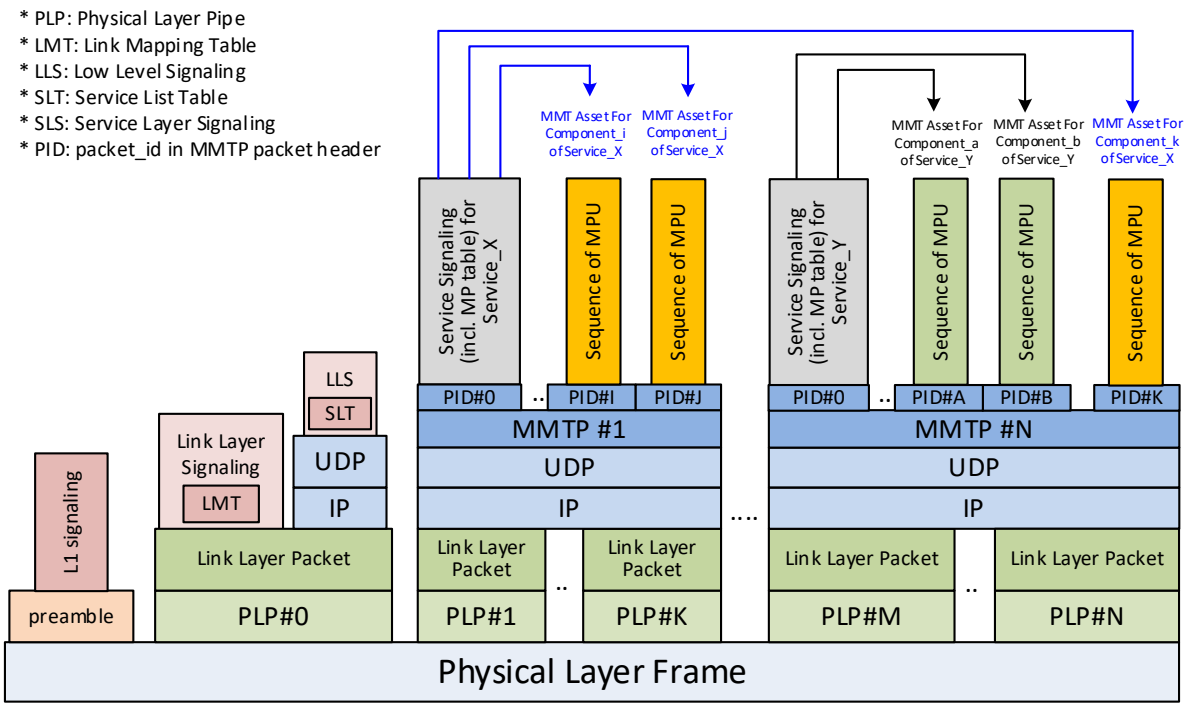


Figure B.1.2 ATSC 3.0 MMTP hierarchical signaling architecture.

B.2 FAST SERVICE SCAN

Fast Service Scan process is described below. ATSC 3.0 receivers can use the LMT of the Link Layer Signaling and the SLT of the LLS to perform the Fast Service Scan.

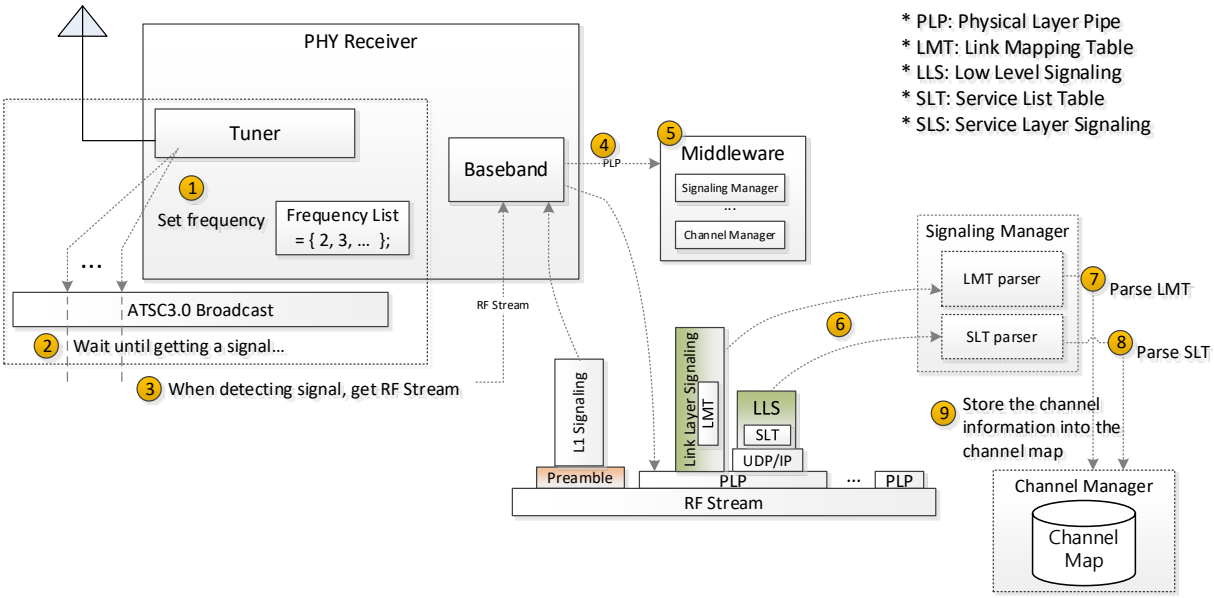


Figure B.2.1 Fast service scan signaling flow.

The Fast Service Scan process is described here.

1) Tuner in receiver will step through frequencies using a predefined frequency list.

2) For each frequency, the tuner determines if a signal is present.
3) When detecting a signal at a given frequency, the baseband processor will extract the L1 Signaling of the Preamble.
4) The baseband processor will send PLP that contains Link Layer Signaling and LLS to the middleware that will extract them from PLP data.
5) The middleware contains functional modules such as a signaling manager, a channel manager and other parts. After receiving PLP data that contains Link Layer Signaling and LLS from the baseband processor, the middleware will pass it to any functional module which manages any type of data (e.g. signaling, audio/video/cc or app-based enhancement) and controls the delivery of data to the appropriate parser and internal cache.
6) The middleware module will extract the LMT from the Link Layer Signaling and will pass the LMT to the LMT parser. The middleware module will extract the SLT from the LLS by `LLS_table_id` which has the value '0x01'. The middleware module will pass the SLT to the SLT parser.
7) The LMT parser parses the LMT and extracts the information which is essential for producing a channel map (e.g. PLPID, Session information [IP address and port number]).
8) The SLT parser parses the SLT and extracts the information which is essential for producing a channel map (e.g. service id, short service name, broadcast SLS signaling information and hidden in the map).
9) Information is stored into the channel map.

B.3 FULL SERVICE SCAN

If receivers perform a full scan with service signaling (USBD or USD) for each service, they can store more abundant information. For example, a longer service name can be acquired from the USD, and it can be stored in the channel map by matching the `@serviceId` values in the SLT and the USD.

The diagram in Figure B.3.1 illustrates a signaling flow for ROUTE full service scan.

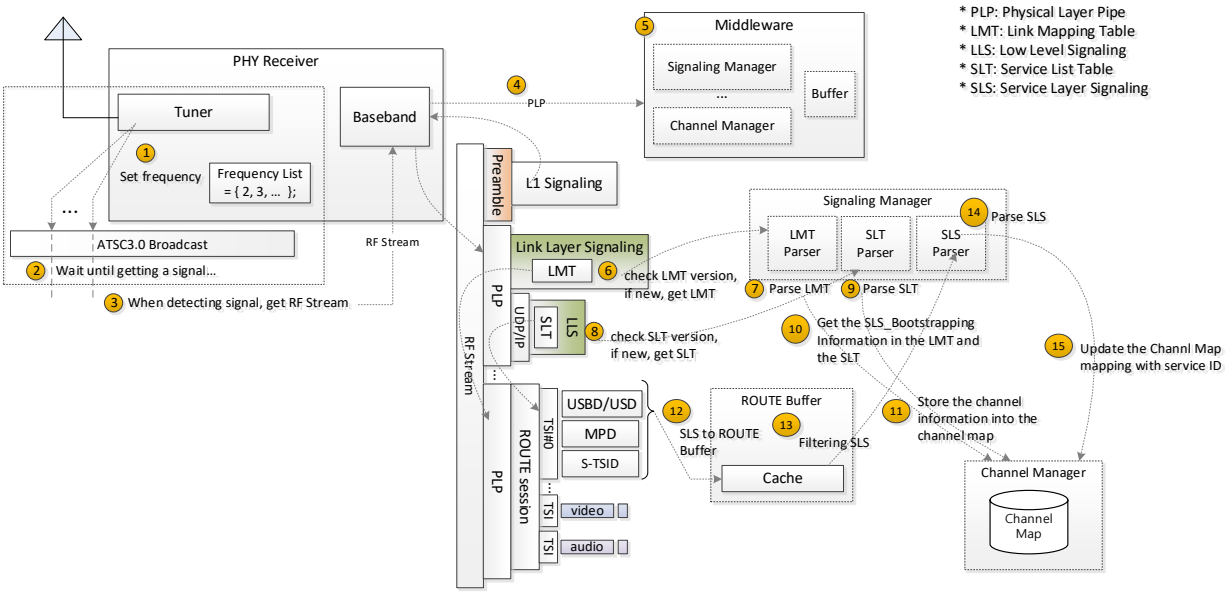


Figure B.3.1 ROUTE full-service scan signaling flow.

The steps for a ROUTE full service scan are described here:

1) Tuner in receiver will step through frequencies using the pre-defined frequency list.
2) For each frequency, tuner will wait until it gets a signal.
3) When detecting a signal from a frequency, baseband processor will extract the L1 Signaling of the Preamble.
4) The baseband process will send PLP that contains Link Layer Signaling and LLS to the middleware that will extract them from PLP data.
5) The middleware contains functional modules such as a signaling manager, a channel manager, buffer and other parts. After receiving PLP data that contains Link Layer Signaling and LLS from the baseband processor, the middleware will pass it to any functional module of it which manages any type of data (e.g. signaling, audio/video/cc or app-based enhancement) and controls data to the appropriate parser and internal cache.
6) Receivers should check the LMT version is new or not. If the version is new, the middleware module can gather LMT and pass the LMT to the LMT parser.
7) LMT parser will parse the data, and extract the information.
8) Receivers should check the `LLS_table_version` of the `LLS_table()` containing the SLT is new or not. It is best to parse it even if it has the same version number as on the last scan. The version number could have wrapped around and by chance be the same number as it was before. If the version is new, the middleware module can gather SLT and pass the SLT to the SLT parser.
9) SLT parser will parse the data, and extract the information.
10) Receiver gets the SLS Bootstrapping information from the LMT and the SLT.
11) Information will be stored into the channel map.
12) Receiver can acquire SLS data the SLS Bootstrapping information from the LMT and SLT and pass it to the internal buffer.
13) Receiver uses the signaling filtering scheme to extract the USD from the SLS and stores it.
14) USD is parsed by the signaling parser. (It is best to parse it even if it has the same version number as on the last scan. The version number could have wrapped around and by chance be the same number as it was before.)
15) Receivers can update the channel map by mapping with the `@serviceId`.

The diagram in Figure B.3.2 illustrates a signaling flow for MMTP full-service scan.

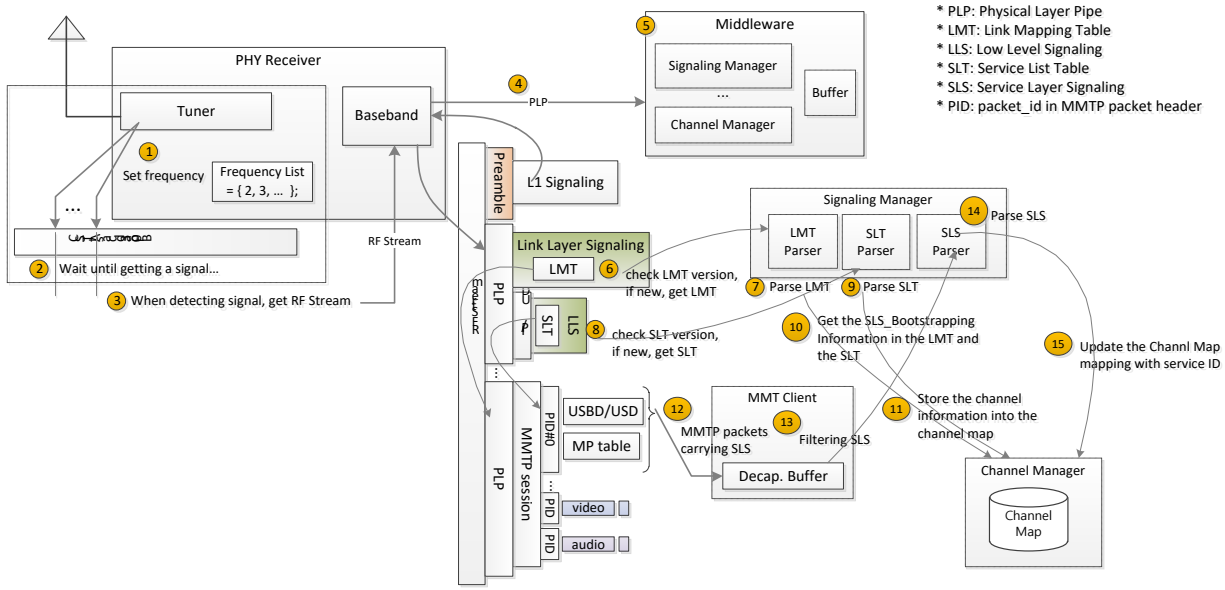


Figure B.3.2 MMTP full-service scan signaling flow.

The steps for a full service scan are described here:

1) Tuner in receiver will step through frequencies using the pre-defined frequency list.
2) For each frequency, tuner will wait until it gets a signal.
3) When detecting a signal from a frequency, baseband processor will extract the L1 Signaling of the Preamble.
4) The baseband process will send PLP that contains Link Layer Signaling and LLS to the middleware that will extract them from PLP data.
5) The middleware contains functional modules such as a signaling manager, a channel manager, buffer and other parts. After receiving PLP data that contains Link Layer Signaling and LLS from the baseband processor, the middleware will pass it to any functional module of it which manages any type of data (e.g. signaling, audio/video/cc or app-based enhancement) and controls data to the appropriate parser and internal cache.
6) Receivers should check the LMT version is new or not. If the version is new, the middleware module can gather LMT and pass the LMT to the LMT parser.
7) LMT parser will parse the data, and extract the information.
8) Receivers should check the LLS_table_version of the LLS_table() containing the SLT is new or not. It is best to parse it even if it has the same version number as on the last scan. The version number could have wrapped around and by chance be the same number as it was before. If the version is new, the middleware module can gather SLT and pass the SLT to the SLT parser.
9) SLT parser will parse the data, and extract the information.
10) Receiver gets the SLS Bootstrapping information from the LMT and the SLT.
11) Information will be stored into the channel map.
12) Receiver can locate MMTP packets carrying SLS data using the SLS Bootstrapping information from the LMT and SLT and pass it to the MMTP Client.
13) MMTP Client extract the SLS including the USD and stores it.

14) USD is parsed by the signaling parser. (It is best to parse it even if it has the same version number as on the last scan. The version number could have wrapped around and by chance be the same number as it was before.)
15) Receivers can update the channel map by mapping with the `@serviceId`.

B.4 SERVICE ACQUISITION IN THE PURE BROADCAST (ONE ROUTE/MMTP SESSION)

When video and audio segments are delivered via pure broadcast with one ROUTE session, the service signaling will be structured like below.

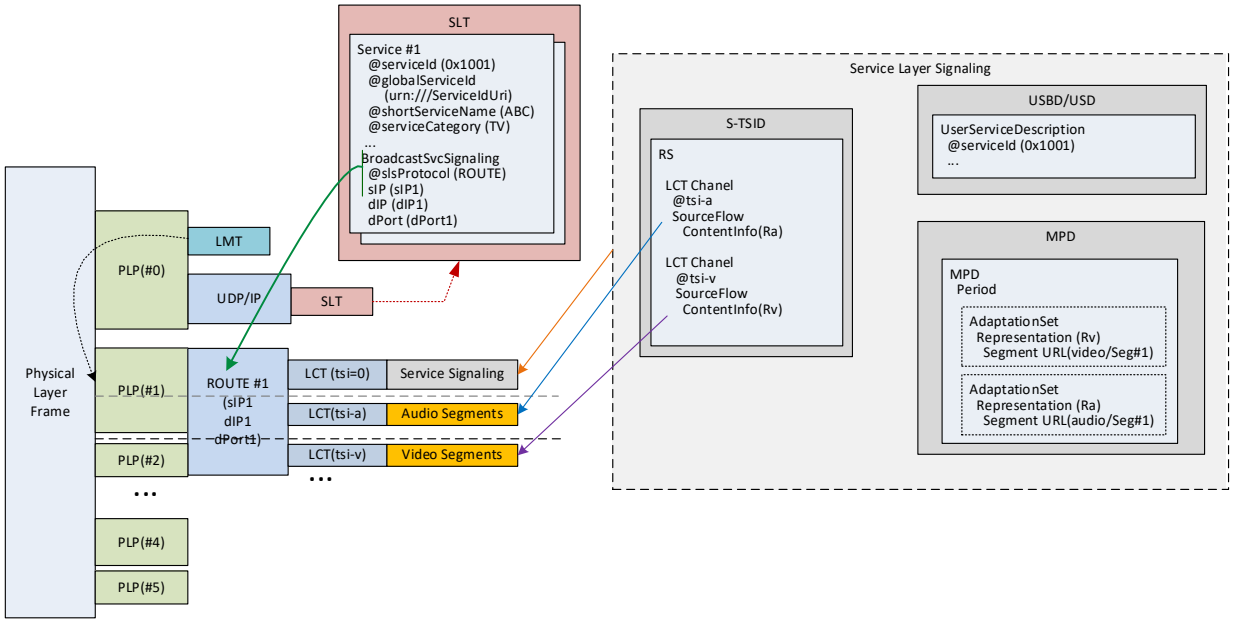


Figure B.4.1 Service acquisition in the pure broadcast (one ROUTE session).

1) USD, S-TSID and MPD will be acquired together and should be parsed. All tables will be needed for service acquisition.
2) Select which Representations to present. In this case, S-TSID should be checked to determine which Representations are delivered via broadcast
3) Receivers will send the information from the signaling (USD, S-TSID and MPD) to the segment acquisition module providing the user preference with them. For example, user prefers the Spanish audio language over the English audio language.
4) In pure broadcast case, receivers can know where to get components without any `DeliveryMethod` element in the USD.

When video and audio MPUs are delivered via pure broadcast with one MMTP session, the service signaling will be structured like below.

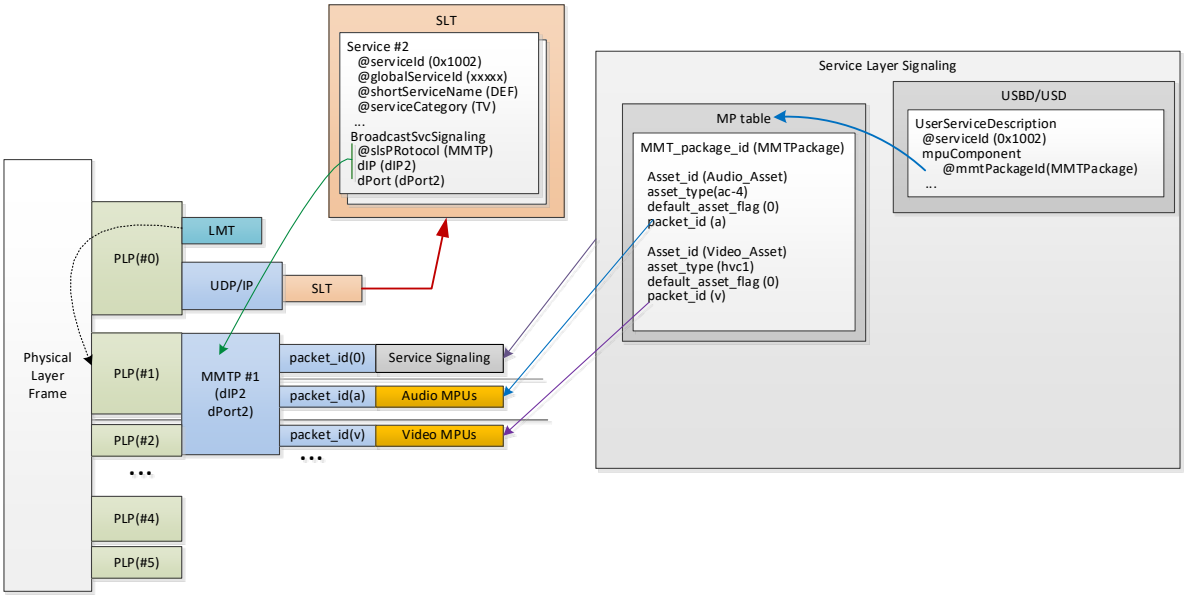


Figure B.4.2 Service acquisition in the pure broadcast (one MMTP session).

1) USD, and MP_table will be acquired together and should be parsed. All tables will be needed for service acquisition.
2) Select which Assets to present. In this case, MP table should be checked to determine which Assets are makred as default Assets.
3) Receivers will send the information from the signaling (USD, and MP table) to the MPU acquisition module providing the user preference with them. For example, user prefers the Spanish audio language over the English audio language.
4) In pure broadcast case with one MMTP session, a component can be acquired by getting MMTP packets having the same packet_id value as the one for the Asset corresponding to the component provided in MP table.

B.5 SERVICE ACQUISITION IN THE PURE BROADCAST (MULTIPLE ROUTE/MMTP SESSIONS)

One service can be composed of multiple ROUTE sessions. In this case, the S-TSID will contain the additional ROUTE session information to allow access to all the Representations. However, this additional information may be optional to render the service.

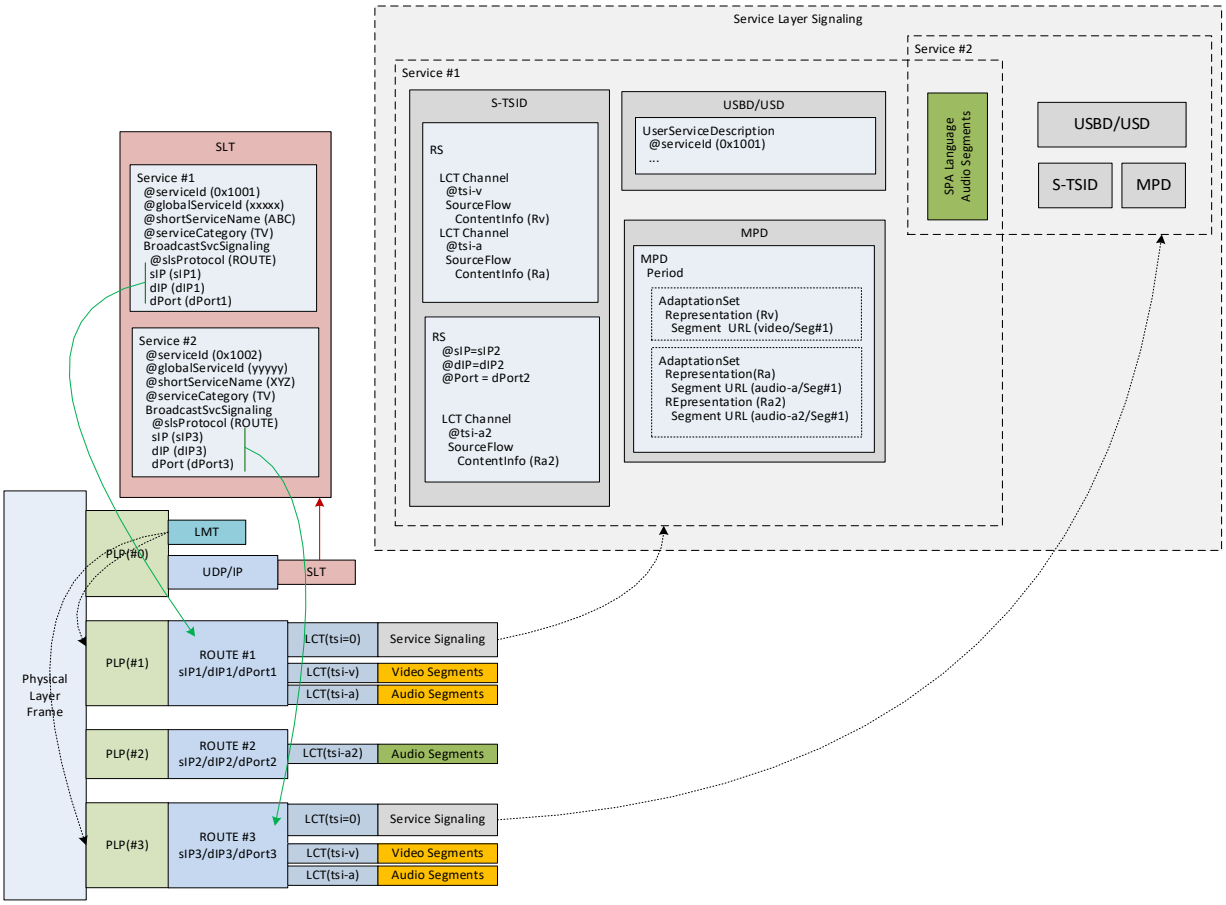


Figure B.5.1 Service acquisition in the pure broadcast (multiple ROUTE sessions).

One service can be delivered by more than one MMTP session as illustrated in Figure B.5.2. In this case, the location of the components can be provided by information for identifying an MMTP session (destination IP address and destination port number) and information for identifying MMTP subflow carrying the component in the MMTP session (packet_id) in the MP table.

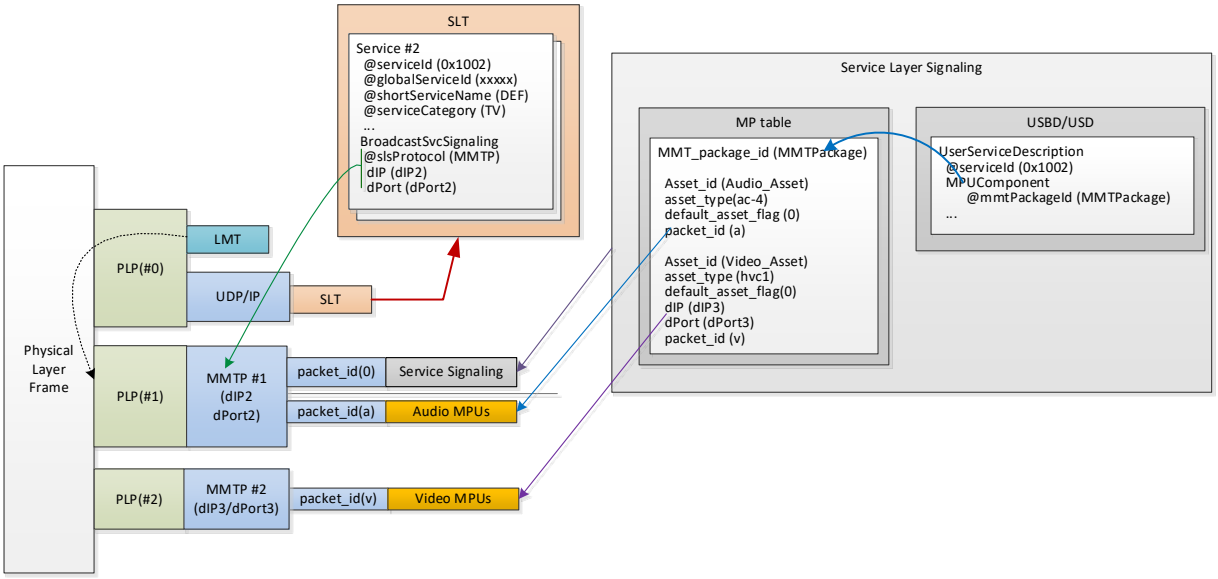


Figure B.5.2 Service acquisition in the pure broadcast (multiple MMTP sessions).

Furthermore, one service can be composed of one MMTP session carrying linear content components and one ROUTE session carrying application-based features as illustrated in Figure B.5.3. In this case, the USBD contains the access information for the S-TSID fragment for a ROUTE session carrying locally-cached service contents.

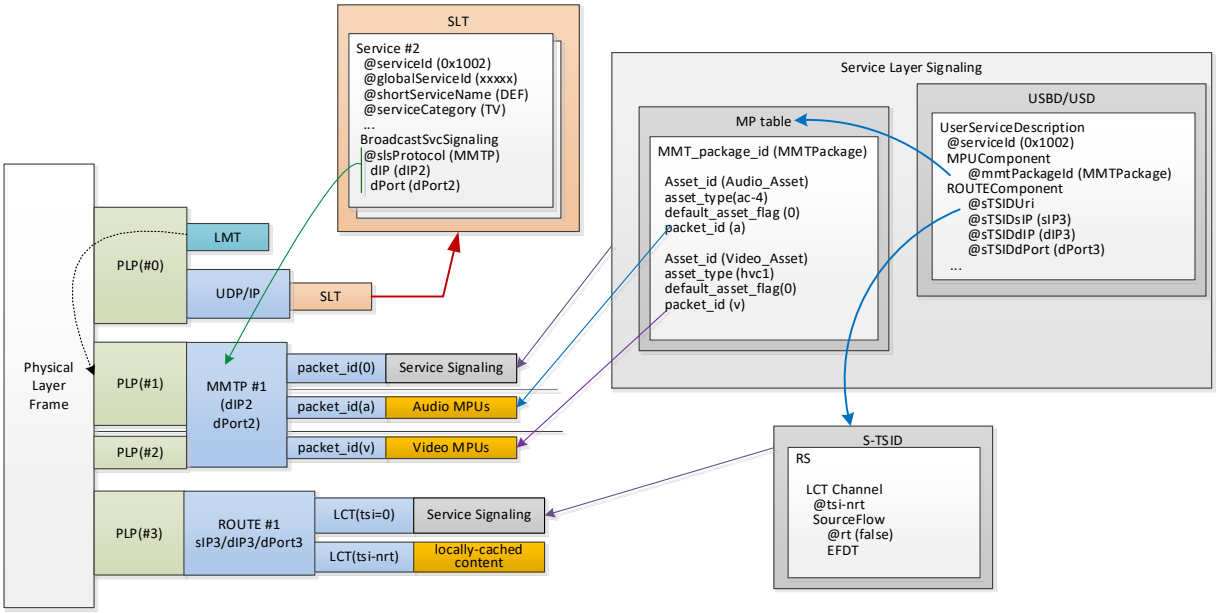


Figure B.5.3 Service acquisition in the pure broadcast (one MMTP session and one ROUTE session).

B.6 ESG BOOTSTRAPPING VIA BROADBAND

ESG bootstrapping via broadband will be signaled in SLT. In this example, all ESG data will be delivered via broadband. Therefore, the ESG broadcast bootstrapping information will be replaced with the ESG broadband bootstrapping information – the attribute `svcInetUrl@urlType` will

indicate if the type of URL is ESG or other. The details of the query mechanism to acquire the ESG data via broadband are outside the scope of this document.

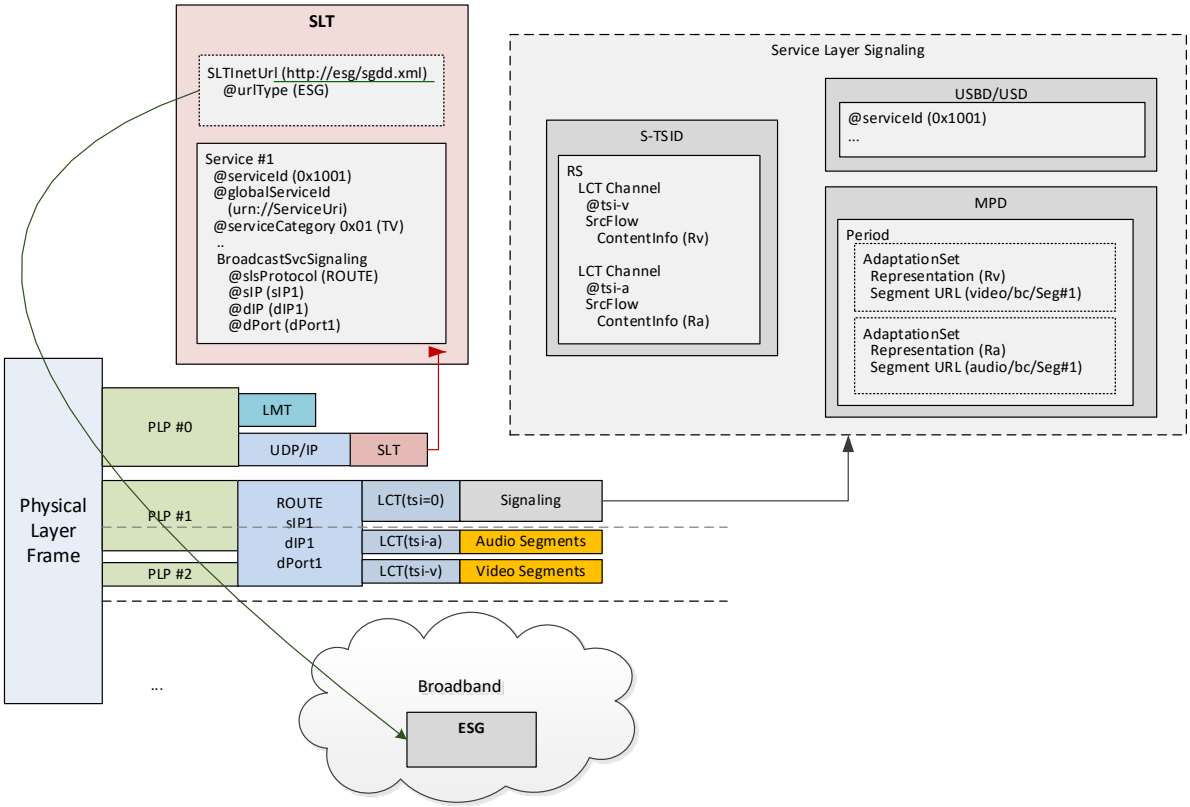


Figure B.6.1 ROUTE ESG bootstrapping via broadband.

B.7 HYBRID DELIVERY (MULTIPLE AUDIO LANGUAGE)

In the ROUTE case, when two or more audio components in different languages are delivered via different delivery paths, one via broadcast and another via broadband, the S-TSID describes all the broadcast components so that the ROUTE client can retrieve the desired components. Moreover, the USD contains URL patterns for broadband and URL patterns for broadcast, so that when a DASH Client issues a request for a Segment, the receiver middleware can describe which Segments will be delivered through which path. The middleware will then know which Segments to request from a remote broadband server, and which ones to look for in the broadcast.

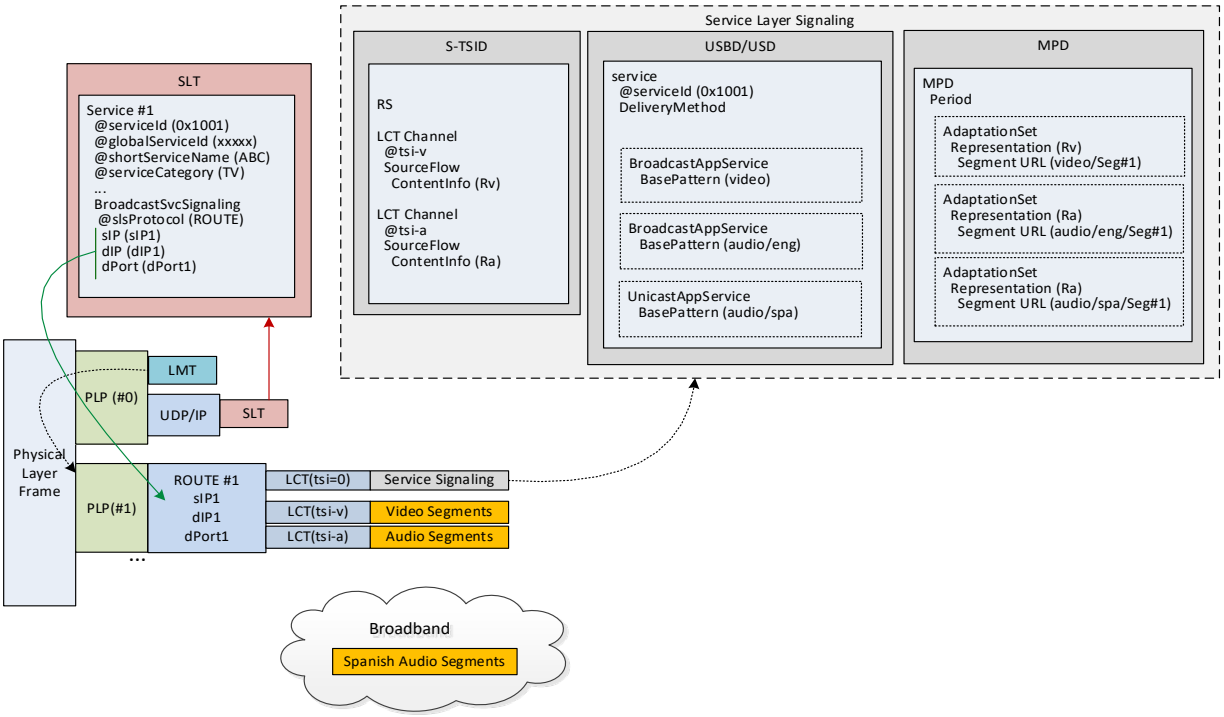


Figure B.7.1 ROUTE hybrid (multiple audio languages).

In the MMTP case, when two or more audio components in different languages are delivered via different delivery paths, one via broadcast and another via broadband, the MP table describes all the broadcast components so that the MMTP Client can retrieve the desired components. Moreover, the USBD contains a `broadbandComponent` element that provides a reference to an MPD fragment containing descriptions for additional audio components delivered over broadband.

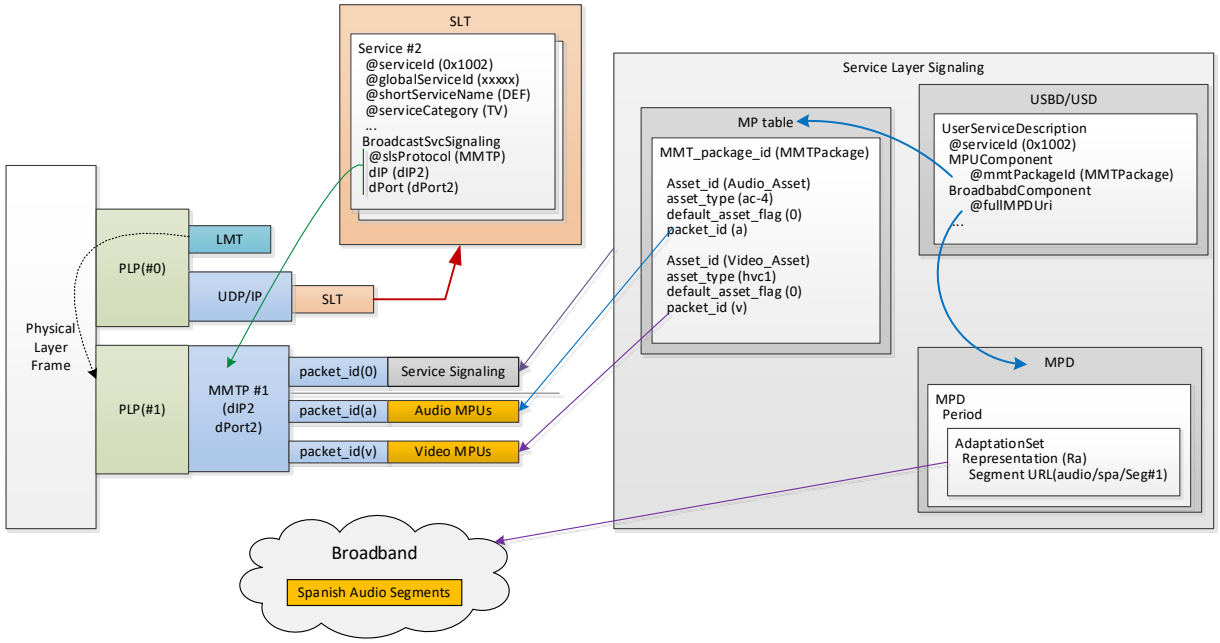


Figure B.7.2 MMTP hybrid (multiple audio languages).

B.8 HANDOFF (BROADCAST TO BROADBAND, AND BACK)

Receivers can hand off reception from broadcast to broadband and back using the signaling described in the USD. The USD describes which components will be delivered via broadcast or broadband. The receiver middleware can retrieve the components from broadcast when that is possible or from broadband if broadcast reception is lost.

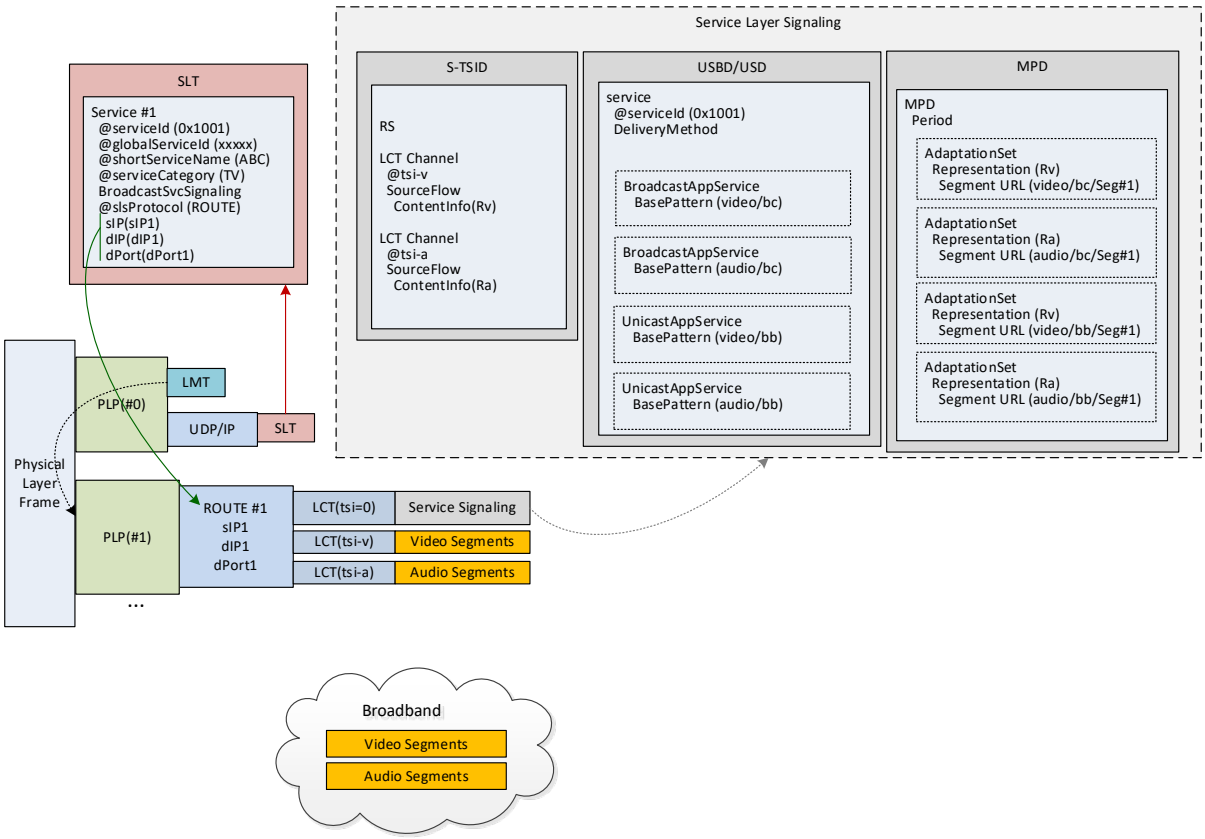


Figure B.8.1 Handoff (broadcast to broadband, and back).

B.9 SCALABLE CODING (CAPABILITY IN THE SLT)

Receivers can do scalable coding using one service delivery by ROUTE.

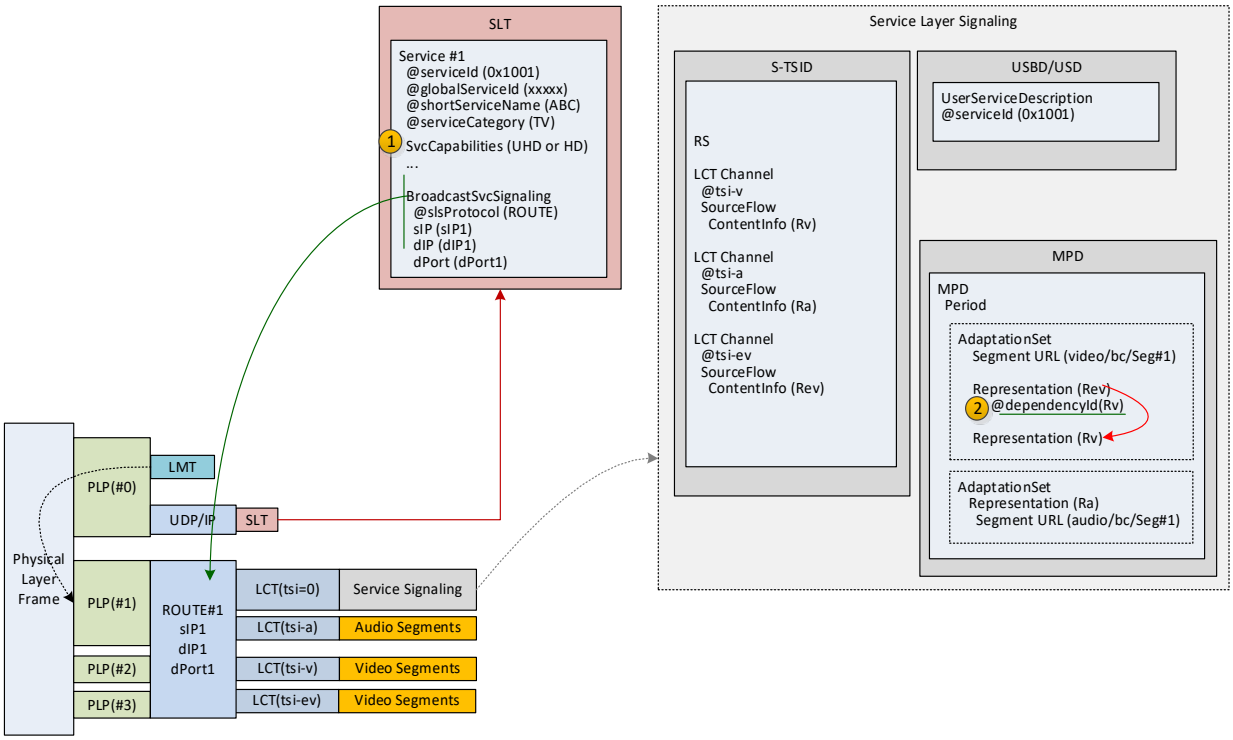


Figure B.9.1 Scalable coding for a linear TV Service delivered by ROUTE (capability in SLT).

1) The SLT may include all capabilities that are essential to render the service. In this example, the video resolution is the essential capability to decode the video, so the capability in the SLT will have the value as 'HD or UHD' (as well as capabilities for other components, such as audio or closed captions or perhaps applications). It means that this program will serve the HD or UHD service now.
2) Receivers can determine which component can be presented to render a UHD service or an HD service by using the MPD.

The scalable coding can also be supported by a service delivered by MMTP as illustrated in Figure B.9.2.

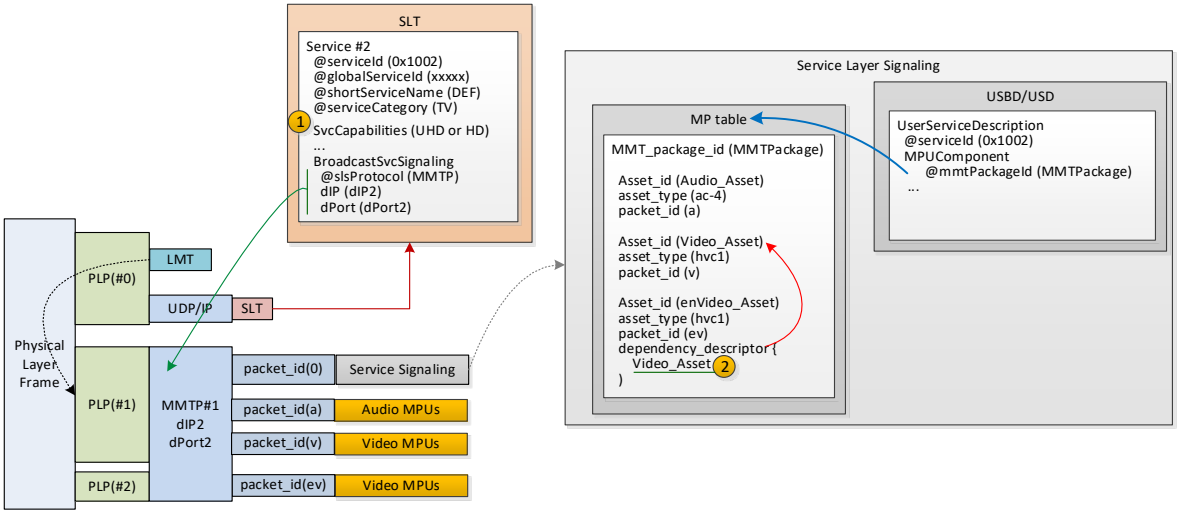


Figure B.9.2 Scalable coding for a linear TV Service delivered by MMTP (capability in SLT).

1) The SLT may include all capabilities that are essential to render the service. In this example, the video resolution is the essential capability to decode the video, so the capability in the SLT will have the value as 'HD or UHD' (as well as capabilities for other components, such as audio or closed captions or perhaps applications). It means that this program will serve the HD or UHD service now.
2) Receivers can know which component will be presented to render UHD service or HD service by using the MP table.

B.10 SLT DELIVERY PATH

The PLP carrying the SLT can be used to deliver service components as well.

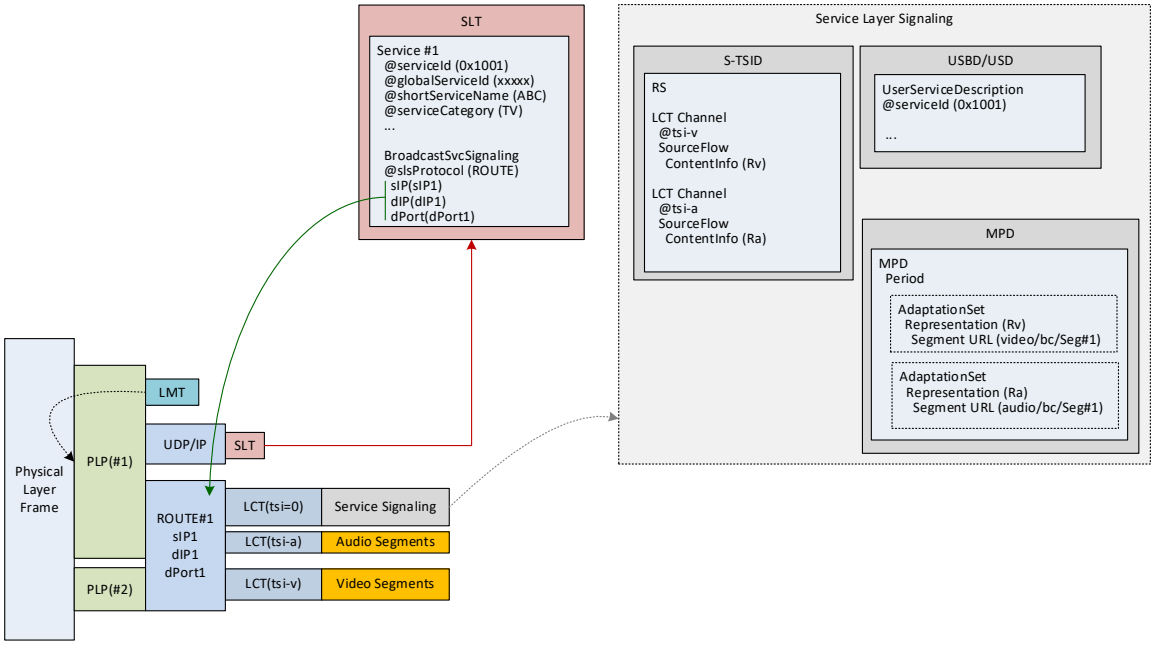


Figure B.10.1 ROUTE SLT delivery path.

B.11 MULTIPLE SLTs IN A BROADCAST STREAM

Multiple SLTs can be present in one broadcast stream.

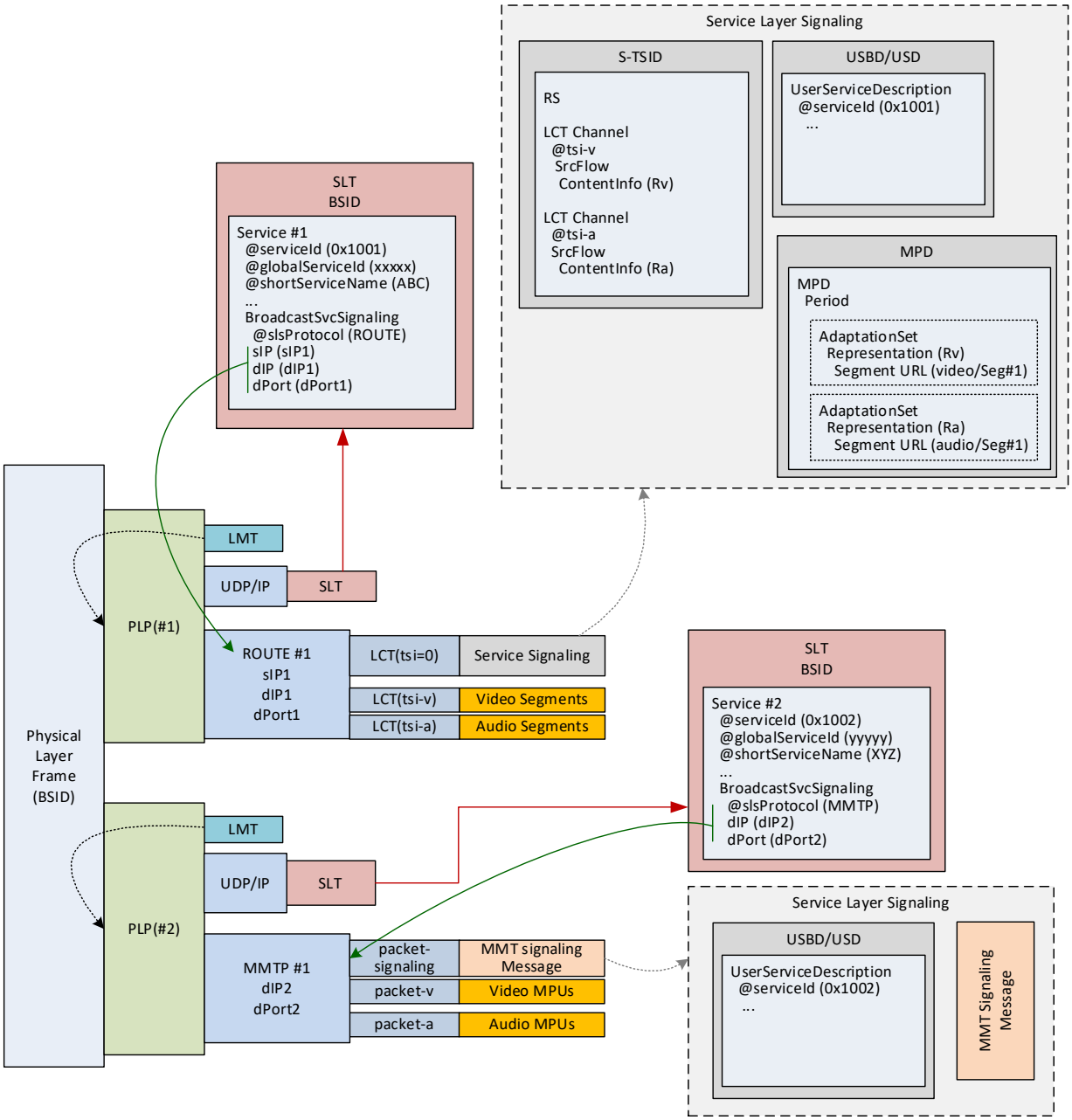


Figure B.11.1 Multiple SLTs.

B.12 PORTIONS OF A SERVICE DELIVERED IN MULTIPLE RF CHANNELS WITHOUT CHANNEL BONDING

The **`SLT.Service`** element associated with the essential portion of the Service has **`OtherBsid`** element which has the value of `@type` attribute equal to 2. The S-TSID describes LCT channels and ROUTE sessions for the components delivered in the Broadcast Stream which delivers S-TSID itself.

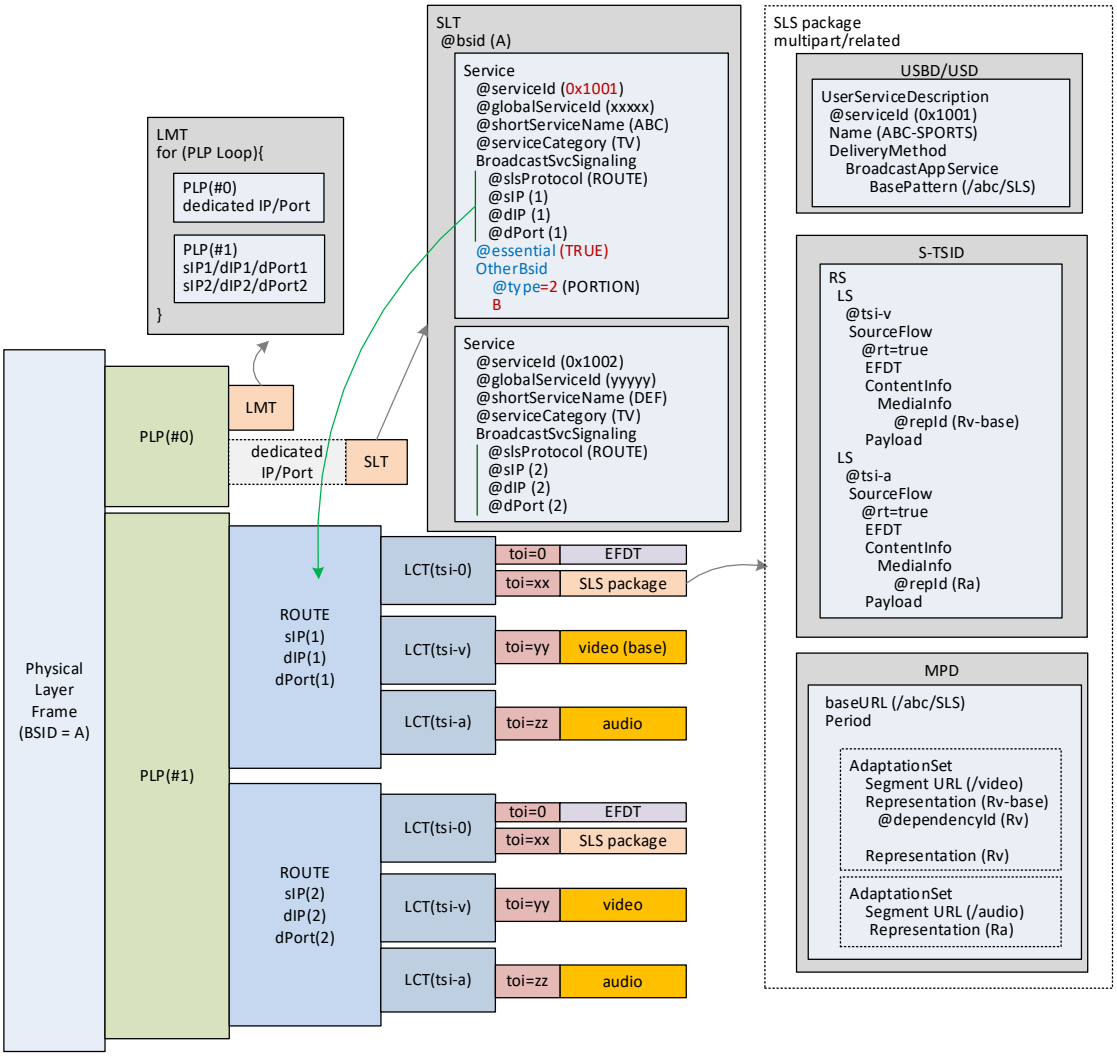


Figure B.12.1 Portions of a Service delivered in multiple RF channels without channel bonding (first RF Channel).

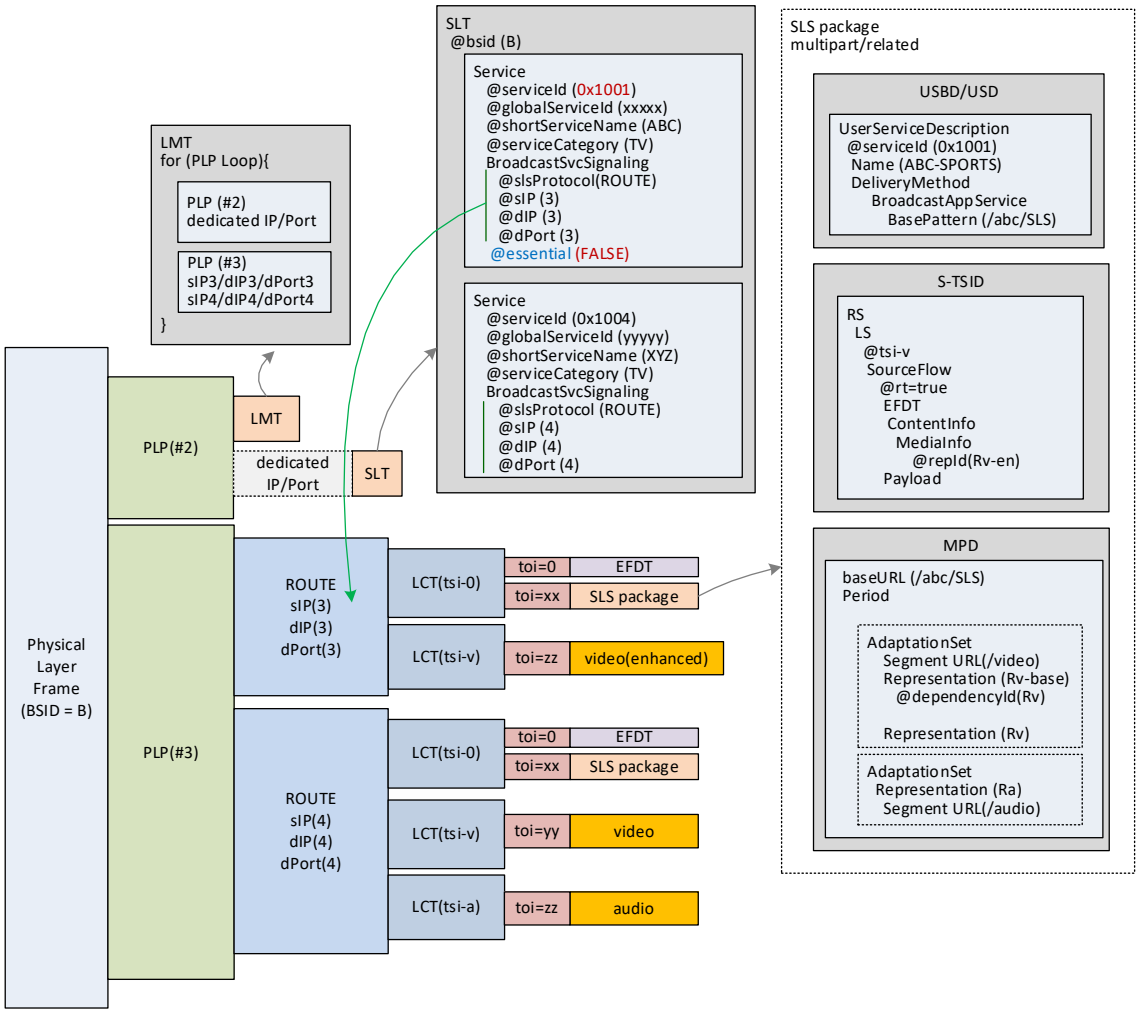


Figure B.12.2 Portions of a Service delivered in multiple RF channels without channel bonding (second RF Channel).

B.13 DUPLICATES OF A SERVICE DELIVERED IN MULTIPLE RF CHANNELS WITHOUT CHANNEL BONDING

The **`SLT.Service`** element has **`OtherBsid`** element which has the value of `@type` attribute equal to 1. There is no **`SLT.Service`**`@essential` attribute. The S-TSID describes LCT channels and ROUTE sessions for the components delivered in the Broadcast Stream which delivers S-TSID itself.

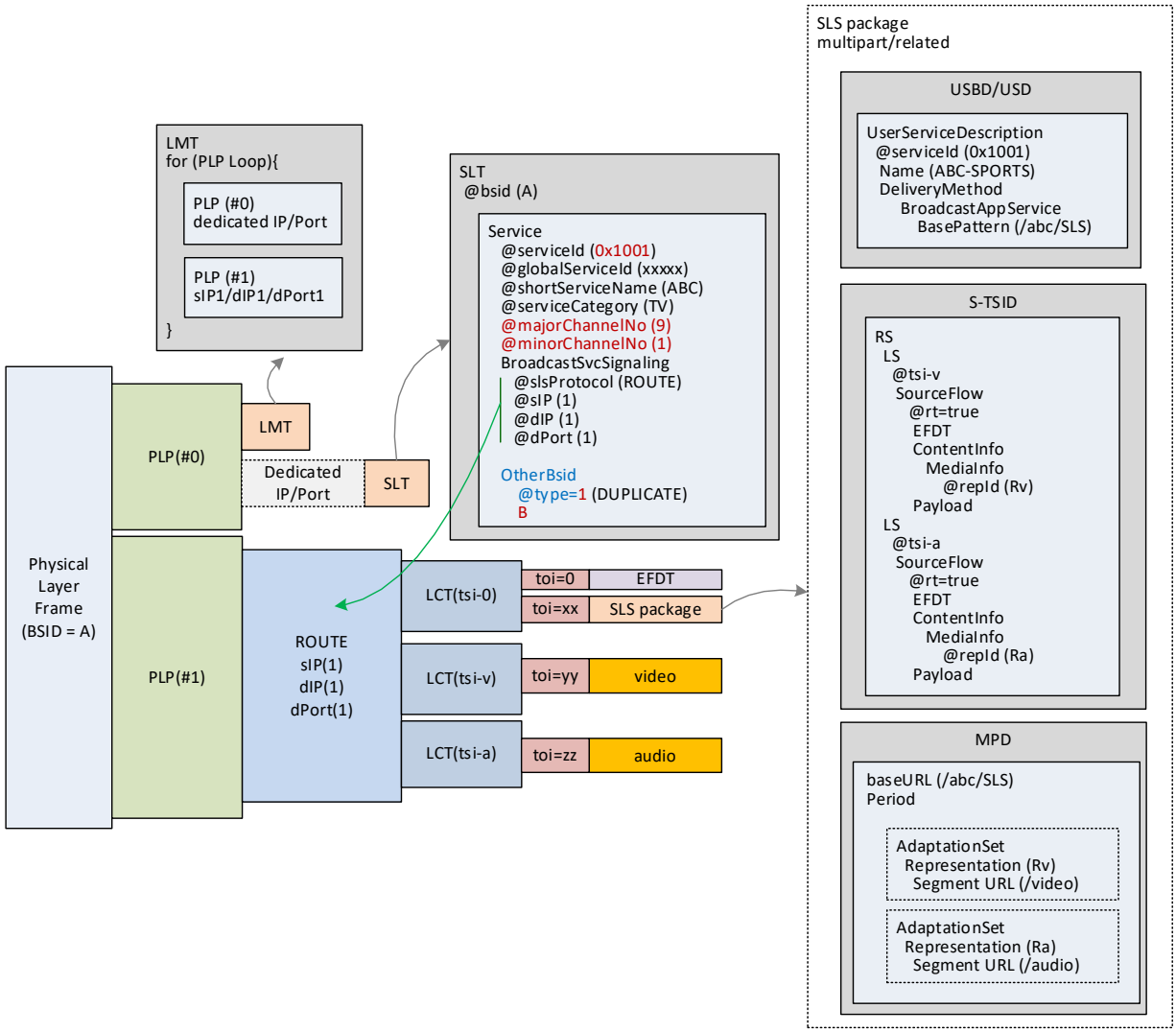


Figure B.13.1 Duplicates of a Service delivered in multiple RF channels without channel bonding (first channel).

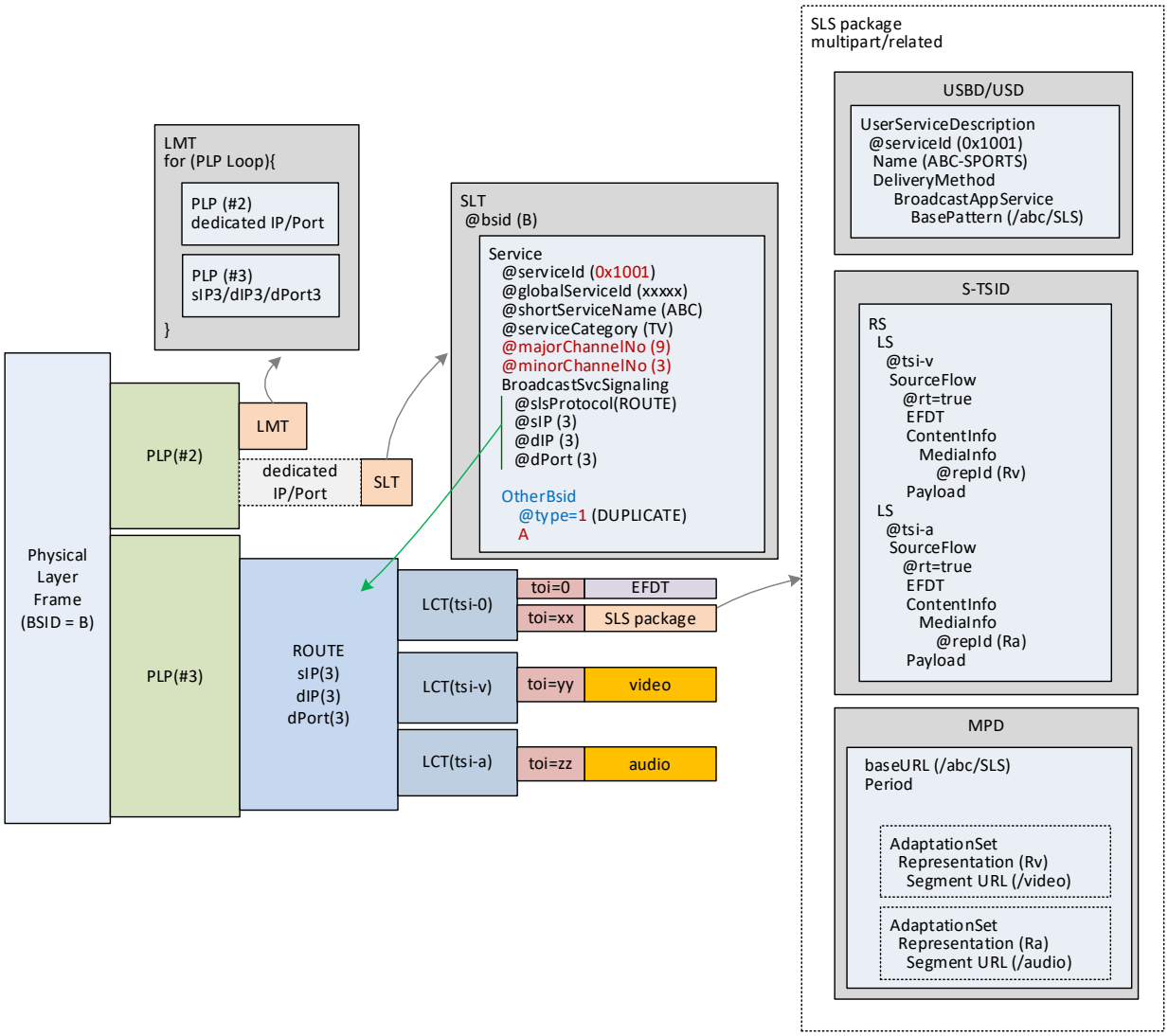


Figure B.13.2 Duplicates of a Service delivered in multiple RF channels without channel bonding (second channel).

B.14 PORTIONS OF A SERVICE DELIVERED IN MULTIPLE RF CHANNELS WITH CHANNEL BONDING – CASE#1: ESSENTIAL PORTION IS DELIVERED IN NON-BONDED PLP(S)

The **`SLT.Service`** element associated with the essential portion of the Service has **`OtherBsid`** element which has the value of `@type` attribute equal to 2. The S-TSID describes LCT channels and ROUTE sessions for the components delivered in the Broadcast Stream which delivers S-TSID itself.

The **`SLT`**`@bsid` attribute associated with the bonded PLP(s) has the value of "A B", and it is referenced by the **`SLT.OtherBsid`** element associated with essential portion of the Service.

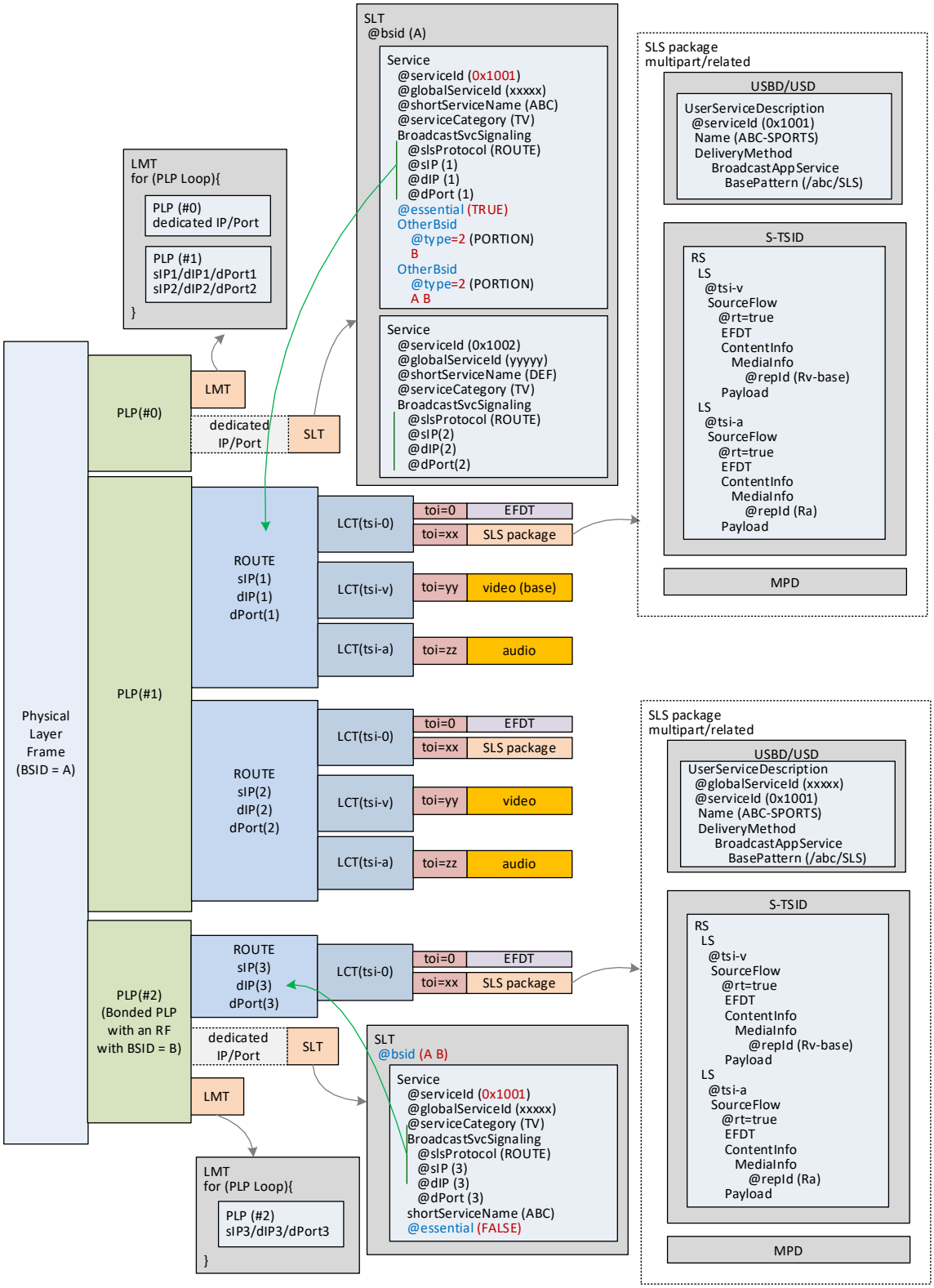


Figure B.14.1 Portions of a Service delivered in multiple RF channels with channel bonding – Case #1: essential portion is delivered in non-bonded PLP(s) (first channel).

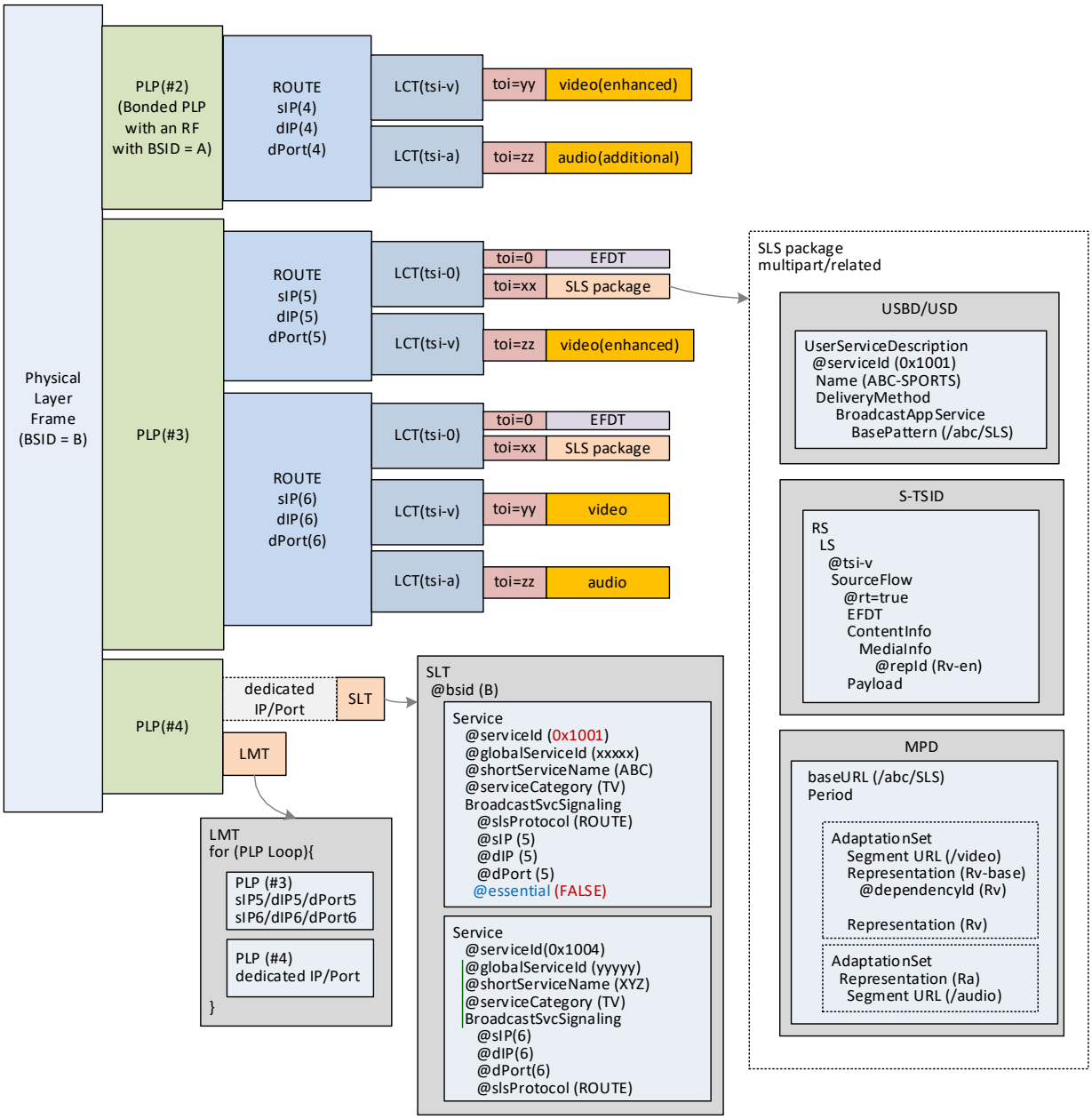


Figure B.14.2 Portions of a Service delivered in multiple RF channels with channel bonding – Case #1: essential portion is delivered in non-bonded PLP(s) (second channel).

B.15 PORTIONS OF A SERVICE DELIVERED IN MULTIPLE RF CHANNELS WITH CHANNEL BONDING – CASE #2: ESSENTIAL PORTION IS DELIVERED IN BONDED PLP(S)

Only the **`SLT`** associated with the essential portion of the Service lists the Service. The **`SLT.Service`** element associated with the essential portion of the Service has **`OtherBsid`** element which has the value of `@type` attribute equal to 2. The S-TSID is delivered in the bonded-PLP only and it describes LCT channels and ROUTE sessions for all the components of the Service.

The **`SLT`**`@bsid` attribute associated with the bonded PLP(s) has the value of "A B".

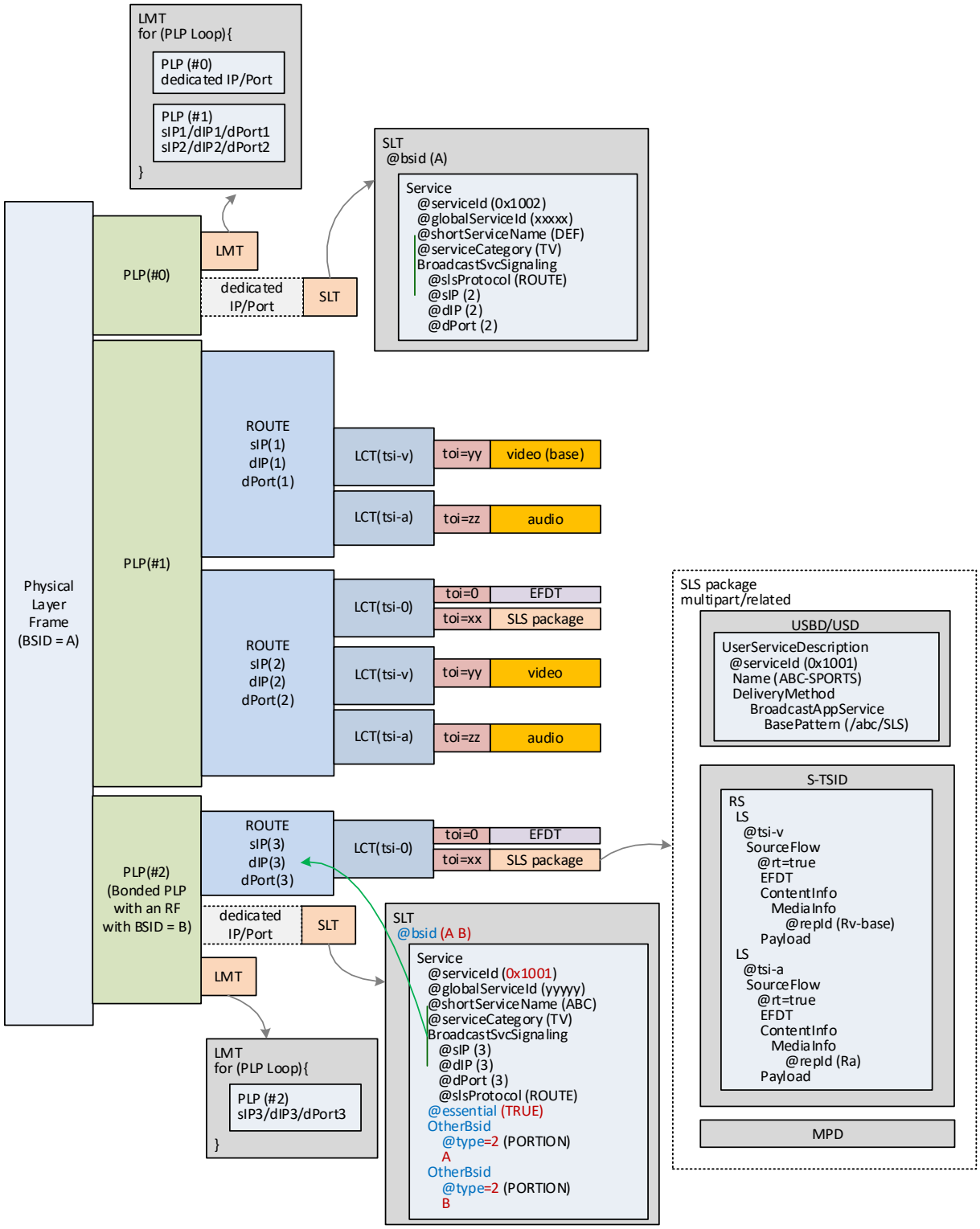


Figure B.15.1 Portions of a Service delivered in multiple RF channels with channel bonding – Case #2: essential portion is delivered in bonded PLP(s) (first channel).

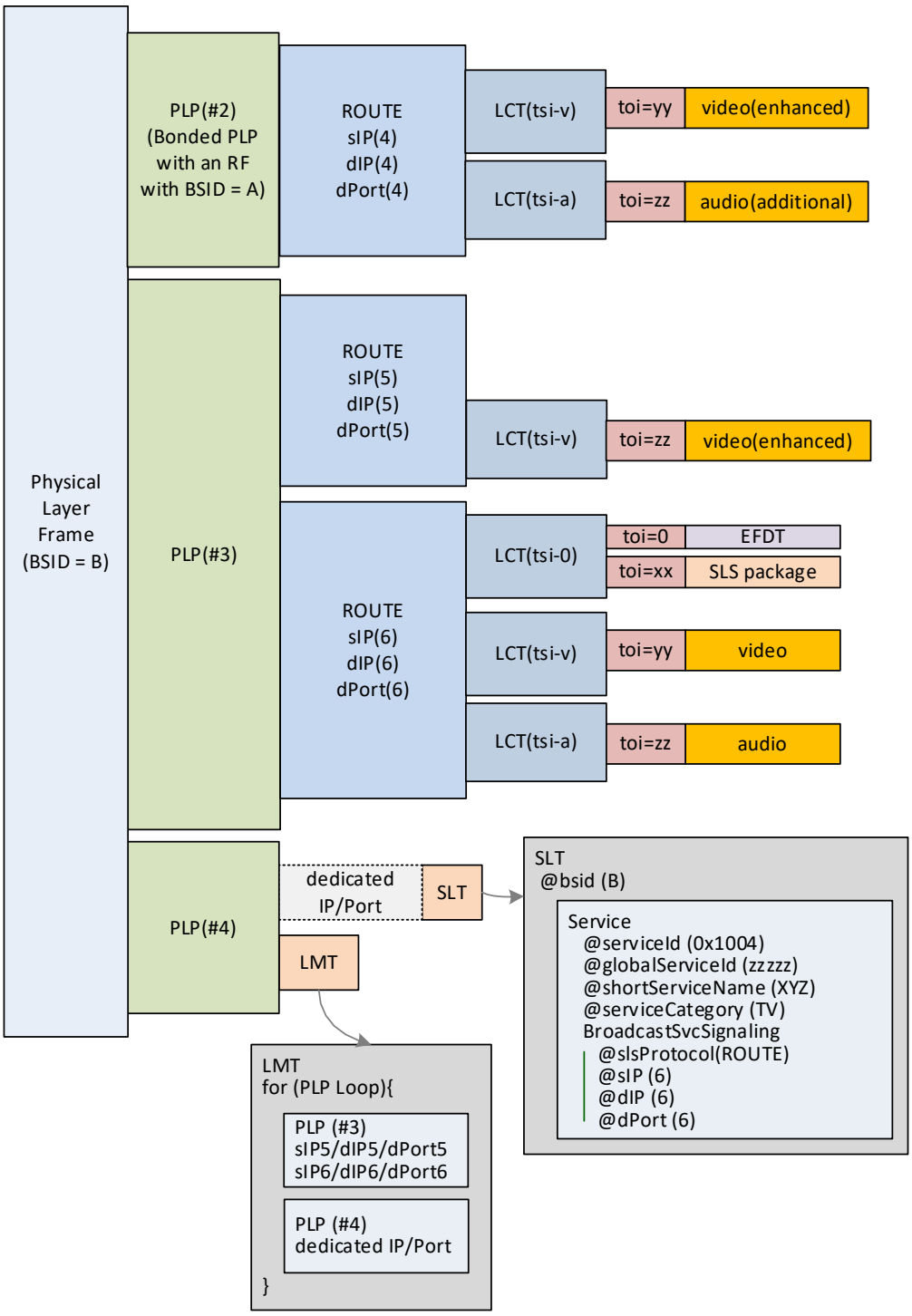


Figure B.15.2 Portions of a Service delivered in multiple RF channels with channel bonding – Case #2: essential portion is delivered in bonded PLP(s) (second channel).

B.16 DUPLICATES OF A SERVICE DELIVERED IN MULTIPLE RF CHANNELS WITH CHANNEL BONDING

The **`SLT.Service`** element has **`OtherBsid`** element which has the value of `@type` attribute equal to 1. There shall be no **`SLT.Service`**`@essential` attribute. The S-TSID describes LCT channels and ROUTE sessions for the components delivered in the Broadcast Stream which delivers S-TSID itself.

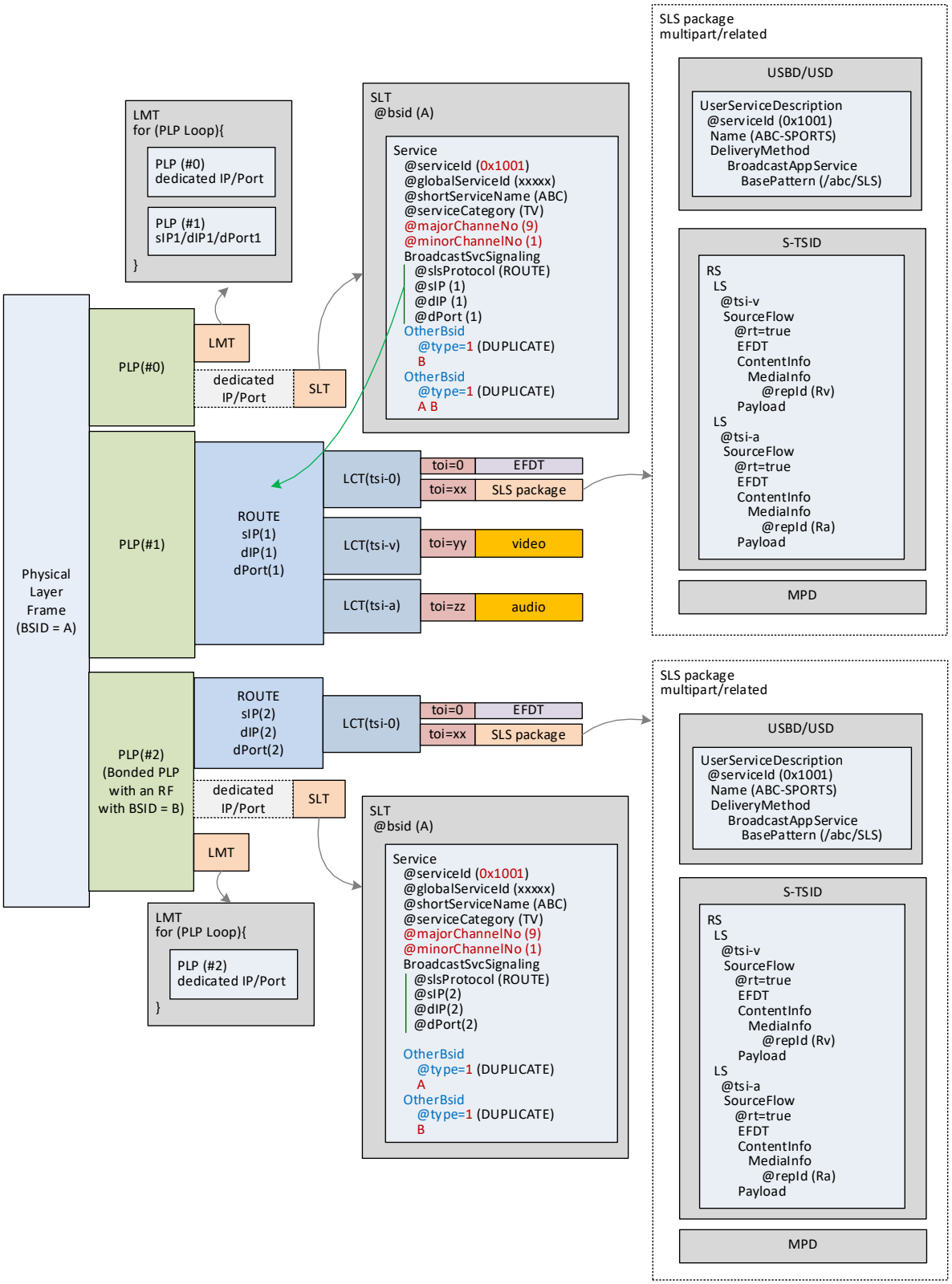


Figure B.16.1 Duplicates of a Service delivered in multiple RF channels with channel bonding (first channel).

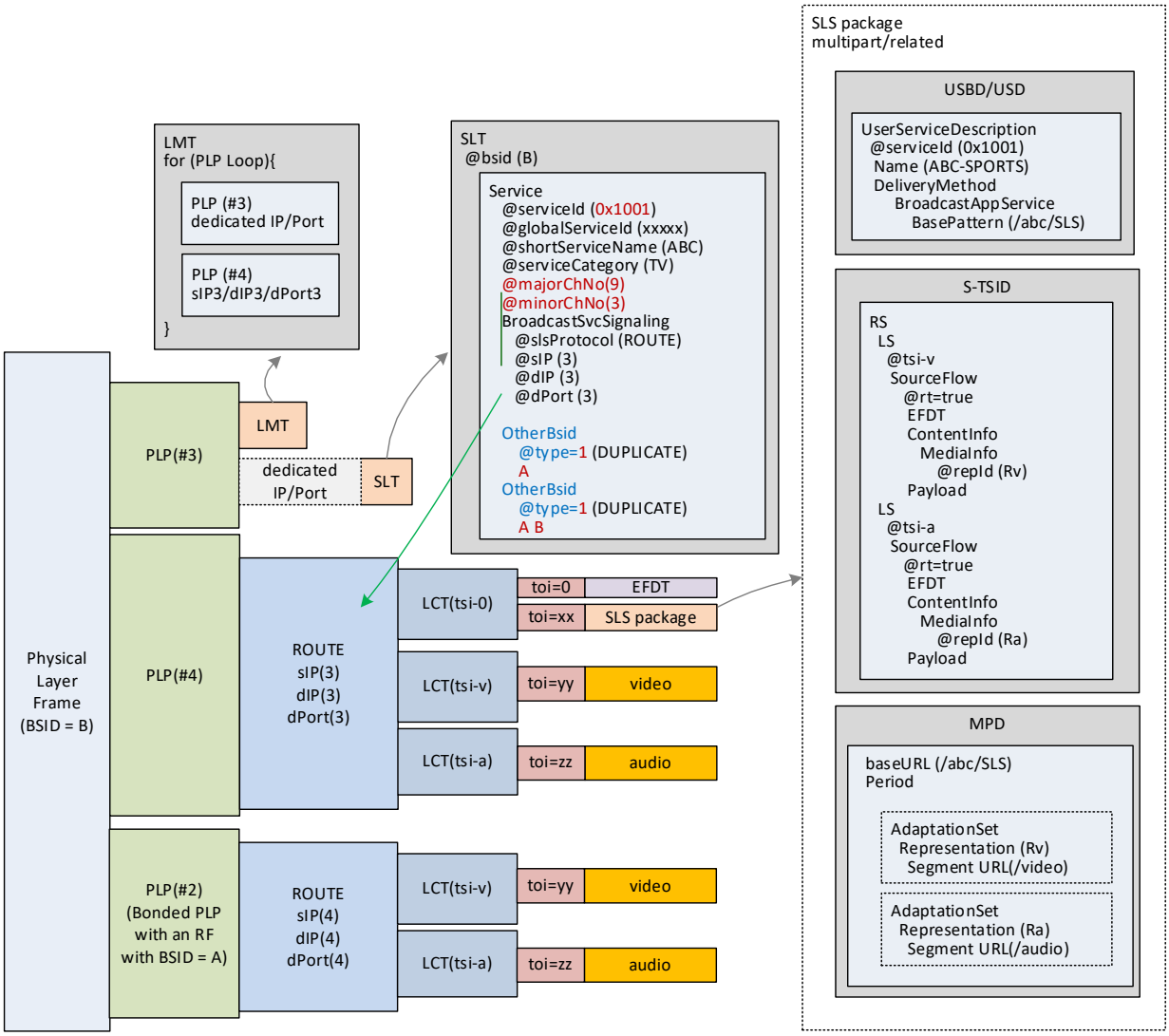


Figure B.16.2 Duplicates of a Service delivered in multiple RF channels with channel bonding (second channel).

Annex C: Filtering for Signaling Fragments

The TOI value for SLS packages shall be computed using the following formula:

$$\text{TOI} = \mathbf{G}*2^{31} + \mathbf{U}*2^{16} + \mathbf{S}*2^{17} + \mathbf{M}*2^{18} + \mathbf{A}*2^{19} + \mathbf{H}*2^{22} + \mathbf{D}*2^{23} + \mathbf{Version}$$

where:

G – is equal to 1 when the package containing the SLS fragment(s) is compressed with gzip [16] and 0 when the package is uncompressed;

U – is equal to 1 when the USBD/USD is contained in this package and 0 when USBD/USD is not contained in this package.

S – is equal to 1 when the S-TSID is contained in this package and 0 when S-TSID is not contained in this package.

M – is equal to 1 when the MPD is contained in this package and 0 when MPD is not contained in this package.

A – is equal to 1 when the APD is contained in this package and 0 when APD is not contained in this package.

H – is equal to 1 when the HELD is contained in this package and 0 when HELD is not contained in this package.

D – is equal to 1 when the DWD is contained in this package and 0 when DWD is not contained in this package.

Version – The version number of the package in the range 0-255. When the package contains a single fragment, this field shall contain the version number of that fragment. When the package contains multiple fragments, this field shall contain a version number for the package, so that it is possible to identify when some fragment contained in the object has changed. The version number of the package shall increment by 1 (modulo 2^8) each time any fragment in the object is changed (e.g. **Version** = (**Version** +1) mod 256).

Figure C.1 diagrams the bit positions of these fields within the 32-bit TOI field.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|G|   Reserved  |D|H| 0 |A|M|S|U|    Reserved   |    Version    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure C.1 TOI bit assignment for SLS packages.

Reserved – These bits shall be set to ‘0’ and are reserved for future ATSC use.

For example, for a gzipped package containing USBD/USD, S-TSID, MPD, and HELD, for version = 3, the TOI would be 2,152,136,707 = 0x80470003.

Annex D: Template-based Compression

D.1 DIFF AND PATCH OPERATION

An XML signaling fragment can be compressed not only using compression tools like gzip but also by alternative means such as the diff and patch tools. In the diff and patch process, an XML signaling template is pre-shared between sender and receivers. The process at the sender compares two XML files, the XML signaling template and the XML signaling instance, and produces an output of the difference between the two, referred to as Diff.

Diff is encapsulated in the element **`metadataEnvelope`** as an ordinary XML signaling instance. When Diff is generated by the sender, Diff is encapsulated in the element **`update`** then encapsulated in the **`metadataEnvelope`**. Diff is delivered to multiple receivers through the signaling channel. The receiver captures it and checks if content of element **`metadataFragment`** contains element **`diffUpdate`**, if it does, the receiver recognizes it has to be handled in this compression mode.

The receiver looks for signaling template of attribute `metadataURI` – `SignalingTemplateID` with optional attribute version – `SignalingTemplateVersion` in cache stores pre-shared signaling templates. If not found, the receiver will try to GET the signaling template with url of `SignalingTemplateID`.

The receiver recovers signaling instance by applying delivered Diff on retrieved signaling template. The instantiated signaling fragment will have a pair of attribute `metadataURI` – `SignalingInstanceID` and attribute version – `SignalingInstanceVersion`. Only difference between updated part against template (e.g. element or attribute value added, changed or removed, etc.) needs to be transported rather than the complete file. It is obvious that it could gain much more efficiency by diff and patch operation rather than ordinary compression schemes if the differences are very small compared to the original complete fragment.

The metadata envelope and metadata fragment may be compressed using gzip as described in MBMS [14].

The following XML instance notation shows how Diff is encapsulated in the element **`metadataEnvelope`**.

```
<metadataEnvelope xmlns="urn:3gpp:metadata:2005:MBMS:envelope">
<item metadataURI="SignalingInstanceID" version="SignalingInstanceVersion">
  <metadataFragment>
    <![CDATA[<diffUpdate>
        <templateID>SignalingTemplateID</templateID>
        <templateVersion>SignalingTemplateVersion</templateVersion> - optional
        <update>Diff</update>
        </diffUpdate>]]>
    </metadataFragment>
  </item>
</metadataEnvelope>
```

Figure D.1.1 illustrates the principles.

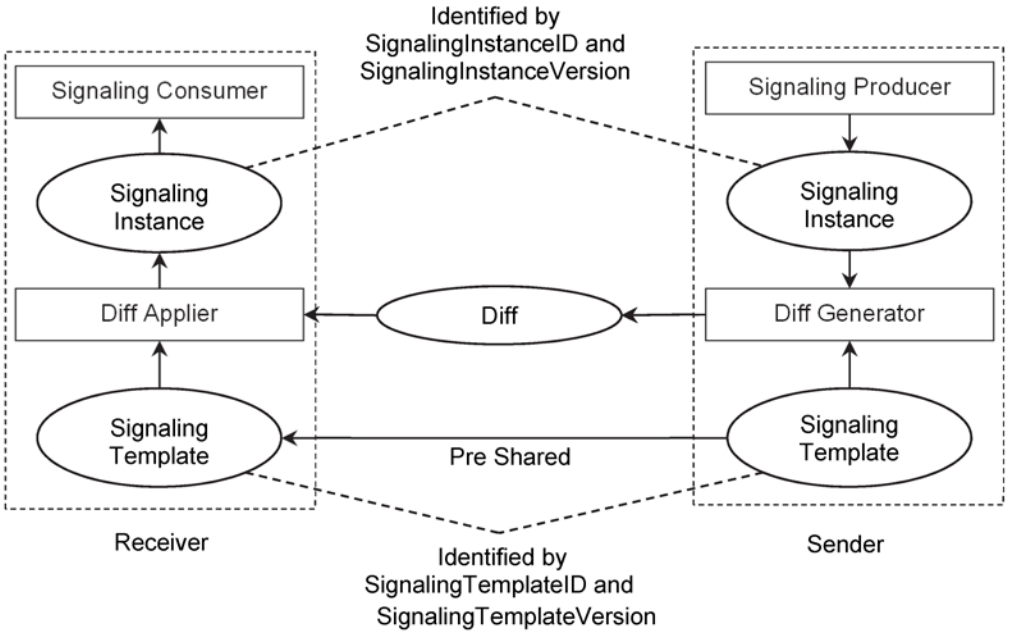


Figure D.1.1 Template based signaling fragment compression.

D.2 DIFF REPRESENTATION

The difference representation shall be an XML patch compliant to the XML Patch Operations framework as defined in RFC 5261 [24] and referenced in "MPD Patch" per the DASH-IF [12].

D.3 TEMPLATE PRE-SHARING

The signaling template fragment is identified by the url contained in the element **`templateID`**. The template itself is pre-shared by being fetched through HTTP(S) over broadband. When the receiver gets the diff message for the first time is cached for future use.

Annex E: Acquisition and Playback of Service Using MMTP

E.1 INTRODUCTION

A single MMT Package can be delivered over one or more MMTP sessions, each of which can be identified by a destination IP address and a destination UDP port number. In a broadcast system, a single RF channel can contain one or more logical channels, called PLPs (Physical Layer Pipe), and each PLP can carry one or more MMTP sessions. In addition, one MMTP session can be carried by more than one PLP.

In order to present a selected MMT Package, the receiver acquires MMTP packets carrying the selected MMT Package. MMT Signaling messages provide information on MMTP sessions carrying the MMTP packets, which can include a destination IP address, a destination UDP port number and a PLP id for each MMTP session. Note that multiple MMT Packages can be delivered by a single MMTP session and that multiple MMT Packages in the same service can be delivered simultaneously. Such MMT Packages do not overlap in their presentation times.

E.2 MAPPING BETWEEN THE PHYSICAL LAYER AND MMTP SESSIONS

A single RF channel may contain multiple PLPs and each PLP can carry one or more services. Furthermore, each service can be delivered over multiple MMTP sessions. Figure E.2.1 shows an example of an RF channel consisting of two PLPs carrying three MMTP sessions. Here a single RF channel (RF channel 1) carries two PLPs, namely PLP1 and PLP2. In PLP1, two MMTP sessions are multiplexed while there is only one MMTP session in PLP2.

In order to tune-in to a selected service, the receiver must know the location of MMTP sessions carrying the MMT Package associated with the selected service. The location of the MMTP sessions can be represented by a combination of an identifier of a broadcast stream, an identifier of a PLP, a destination IP address and a destination UDP port number all of which can be obtained from the SLT and MMT Signaling messages.

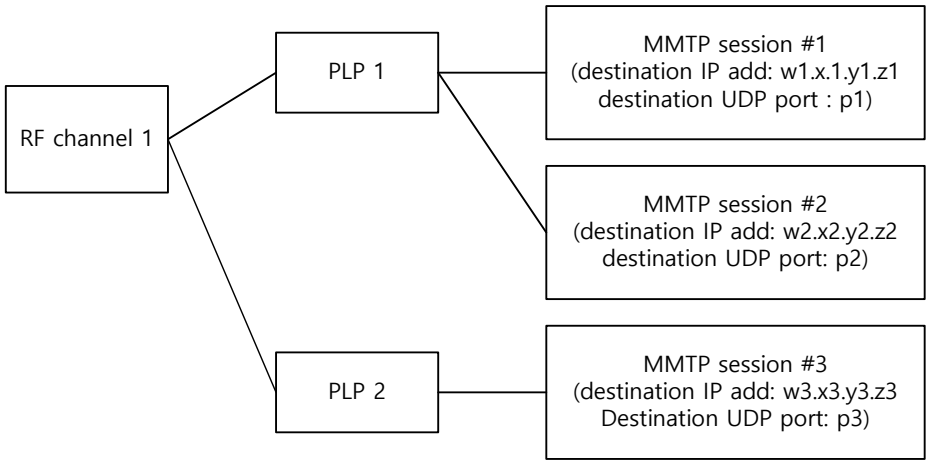


Figure E.2.1 A PHY channel consisting of two PLPs.

E.3 SERVICE ACQUISITION

This section describes a service acquisition process under the assumption that all of the service components are delivered in a single RF channel by a single MMTP session. During the channel scan, the SLS Bootstrapping information for the services carried by MMTP sessions can be acquired from the SLT. The USBD for the services can be obtained from the MMTP sessions signaled by the SLT.

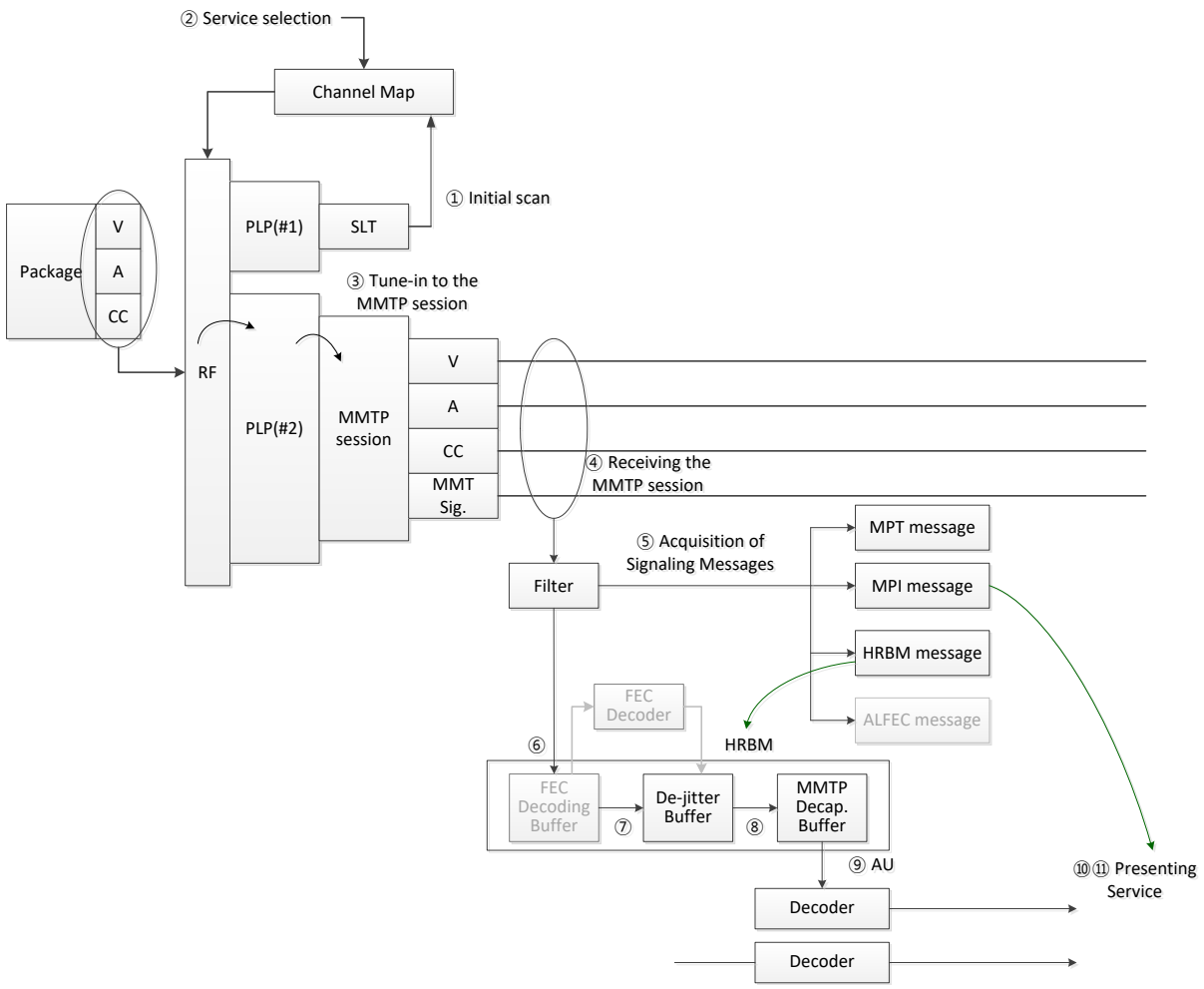


Figure E.3.1 MPU broadcast streaming channel change.

The steps 1-11 for the service acquisition shown in Figure E.3.1 above are:

1) The client acquires a Service List Table (SLT) containing information for rapid channel scan and Service Signaling messages containing service layer information including the MMT_package_id of a service and the location of the MPT message for the corresponding the MMT_package_id.
2) The User selects a Service and its MMT_package_id are identified.
3) Service selection is preceded by an RF channel and the PLP selection using information stored during the channel scan. The MMTP session carrying the MPT message associated for the selected service is acquired.
4) In the referenced MMTP session for the selected channel, MMTP packets containing various content components and signaling messages are obtained.
5) The receiver searches for MMTP packets carrying an MMT Signaling message by looking at the type field of MMTP packet headers to locate MMTP packets containing signaling messages. The MP table extracted from the MPT message is processed to get the list of MMT Assets (ATSC 3.0 content components) comprising the selected Service corresponding to the MMT_package_id. The MP table also provides other necessary information related to each asset such as packet_id, MMT_general_location_info,

MPU_timestamp_descriptor, etc. Other MMT Signaling messages, including MPI, HRBM and AL_FEC signaling messages are processed if necessary.

6) From the series of MMTP packets received, MMTP packets with packet_ids corresponding to each content component are selected (filtered) and stored in their corresponding FEC Decoding Buffers. MMTP packets with repair symbols corresponding to each content component are also received and stored separately.
7) MMTP packets received at the FEC Decoding Buffer are immediately copied into a corresponding De-jitter Buffer and the packet_sequence_number of each MMTP packet is checked to detect missing packets. If some packets are not received by the predefined time given by an AL_FEC message, then the AL-FEC code is applied to recover the missing packets, and the recovered packets are copied to the De-jitter Buffer immediately.
8) MMTP packets are buffered (delayed) at the De-jitter Buffer as instructed by HRBM parameters, which are obtained from HRBM signaling messages in step 5).
9) MMTP packets of MPUs are processed to allow Access Unites (AUs) of components to be decoded.
10) The first AU in an MPU is decoded by the appropriate decoder and presented at the time signaled in the MPU_timestamp_descriptor() (or MPI message if present). The timing information is provided by the server and embedded in the emission though an MMT Signaling message. The descriptor (or MPI message) instructs the associated decoder when the content being decoded is to be presented.
11) The next AU in the MPU is decoded and presented after the presentation time specified in the MPU's 'stts' box and 'ctts' box has passed after the presentation of the first AU. This step is repeated until the last AU of the MPU has been decoded and presented.

Annex F: Rating Region Table Requirements

F.1 INTRODUCTION

The Rating Region Table (RRT) concept was introduced in ATSC in A/65 [1]. In A/65, the RRT is specified as a binary table as part of the Program and System Information Protocol (PSIP). This annex normatively specifies the minimum requirements for a Rating Region Table (RRT) specified in XML format. Details are expected to be provided as needed by regulatory or standards-making bodies within each region.

In the present ATSC 3.0 standard, content advisories appearing within program metadata may refer to RRTs, as described in Section 7.2.3.4. Alternatively, content advisories may refer directly to other rating systems not defined by RRTs.

F.2 RRT REQUIREMENTS

The broadcast emission may include one or more RRTs, each corresponding to a particular identified value of Rating Region. RRTs shall be delivered encoded with XML instance documents. Each instance shall contain at least one, but not more than two, RRTs. The following specifications provide the general rules for construction of RRT instance documents.

RRTs shall be contained within the **`RatingRegionTables`** element, with the characteristics given in Table F.2.1.

The RRT shall be represented as an XML document containing a **`RatingRegionTables`** root element that conforms to the definitions in the XML schema that has namespace

`tag:atsc.org,2016:XMLSchemas/ATSC3/RRT/1.0/`

The definition of this schema is in an XML schema file, RRT-1.0-20170209.xsd, accompanying this standard, as described in Section 3.6 above. The XML schema xmlns short name should be `"rrt"`.

While the indicated XML schema specifies the normative syntax of the **`RatingRegionTables`** element, informative Table F.2.1 below describes the structure of the **`RatingRegionTables`** element in a more illustrative way. The specifications following the table give the normative semantics of the elements and attributes.

Table F.2.1 `RatingRegionTables` Element Structure

Element or Attribute	Use	Data Type	Description
`RatingRegionTables`			
`RatingRegionTable`	1..2		One or two Rating Region Tables
`@regionIdentifier`	1	unsignedByte	Identifies the rating region described.
`RegionIdText`	1..N	TextType	Human-readable string describing the rating region, e.g. “Canada.”
`Dimension`	1..N		One or more elements, each describing one rating dimension in the rating region
`@dimensionLevels`	1	unsignedByte	The number of levels for content advisory in this dimension.
`@dimensionGraduated`	0..1	boolean	Indicates whether the rating dimension is defined on a “graduated scale.”
`DimensionTitle`	0..N	TextType	Human-readable string describing the dimension.
`Rating`	1..N		Definition of each rating in the Dimension
`@ratingValue`	1	unsignedByte	The rating level value in integer form
`RatingValueAbbrev`	1..N	TextType	Abbreviated human-readable string describing the rating value.
`RatingValueString`	1..N	TextType	Human-readable string describing the rating value.

F.2.1 Rating Region Tables Semantics

The following text specifies the semantics of the elements and attributes in the Rating Region Tables.

`RatingRegionTables` – Root element, containing one or more instances of **`RatingRegionTable`**.

`RatingRegionTable` – A complex element describing the rating system for a given region.

`@regionIdentifier` – An unsigned 8-bit integer that shall indicate the rating region associated with this **`RatingRegionTable`** element. Values shall be as specified in CTA-766 [11]. Interpretation in a receiver of the `regionIdentifier` 0x01 RRT requires prior knowledge of CTA-766 [11], therefore transmission is unnecessary.

`RegionIdText` – One or more human-readable text strings, each formatted as an instance of **`TextType`** (see Table F.2.2.). Each shall indicate the region associated with this instance of **`RatingRegionTable`**, e.g. “Canada.” Strings may be provided in multiple different languages. No two strings shall be labeled with the same language.

`Dimension` – A complex type that shall provide information about one rated dimension for the given `regionIdentifier`. One or more **`Dimension`** elements shall be defined.

`@dimensionLevels` – A non-zero unsigned byte value that shall indicate the number of levels defined for the rating dimension.

`@dimensionGraduated` – A Boolean value that shall indicate whether or not the rating dimension describes ratings in a graduated scale. If the higher values of `Rating@ratingValue` indicate a greater amount of the rated content for each rating level, the value of `@dimensionGraduated` shall be `"true"`, otherwise the value of `@dimensionGraduated` shall be `"false"`. The default value (if not present) shall be `"false"`. When a rating is defined to be on a graduated scale, higher rating values represent increasing levels of rated content within the dimension.

`DimensionTitle` – One or more human-readable text strings, each formatted as an instance of **`TextType`** (see Table F.2.2). Each shall indicate the title of the dimension, e.g. “Violence.”

Strings may be provided in multiple different languages. No two strings shall be labeled with the same language.

`Rating` – A complex element providing information about each rating level. One or more instances of **`Rating`** may be present for a given rating dimension. In each `Rating` element the number of occurrences of the `RatingValueAbbrev` element shall be equal to the number of occurrences of `RatingValueString` element.

`@ratingValue` – A non-zero unsigned byte value that shall indicate the value of the rating level.

`RatingValueAbbrev` – One or more human-readable text strings, each formatted as an instance of **`TextType`** (see Table F.2.2). Each shall indicate an abbreviation of the rating level, e.g. “MV.” Strings may be provided in multiple different languages. No two strings shall be labeled with the same language.

`RatingValueString` – One or more human-readable text strings, each formatted as an instance of **`TextType`** (see Table F.2.2). Each shall indicate the rating level, e.g. “Mild Violence.” Strings may be provided in multiple different languages. No two strings shall be labeled with the same language.

The following informative table and the normative text that follow specifies the syntax and semantics of **`TextType`**.

Table F.2.2 `TextType` Element Structure

Element or Attribute	Use	Data Type	Description
`TextType`	1	string	
`@lang`	0..1	lang	The language of the string. Default is 'en' (English).

`TextType` – A text string, required.

`@lang` – The language of the string specified according to BCP 47 [32]. When the `@lang` attribute is not present, the value shall be assumed to be "`en`" (English).

Annex G: Emergency Message Signaling

G.1 EMERGENCY ALERT SYSTEM STRUCTURE

The AEA provides an emergency notification mechanism in ATSC 3.0, which is capable of forwarding a broad range of emergency data, which can include urgent bulletins, advisories, all-hazard warnings, emergency-related messaging, and other urgent information over an ATSC 3.0 system. This annex describes one manner in which the AEA may function in support of external public warning alerts sources, however the emergency information service provided by the AEA is not limited to this specific purpose, and may be utilized to convey additional forms of urgent messaging.

G.1.1 Overview of the System

With ATSC 3.0, the following types of emergency information can be delivered to TV receivers:

- Emergency Alert wake-up bits – allows a receiver that is not actively decoding and displaying pictures to become aware of an emergency. When a receiver that is capable of supporting the wake-up feature is not actively decoding and displaying pictures, but is monitoring the RF signal, it can check the PHY bootstrap signaling for the wake-up bits. If the wake-up bits in the PHY bootstrap indicate a new or updated emergency, a receiver can “wake up” and process emergency signaling information, including begin providing picture and/or sound to a viewer. Figure G.1.1 shows more specific receiver behaviors regarding how a receiver can be switched between the two different states.

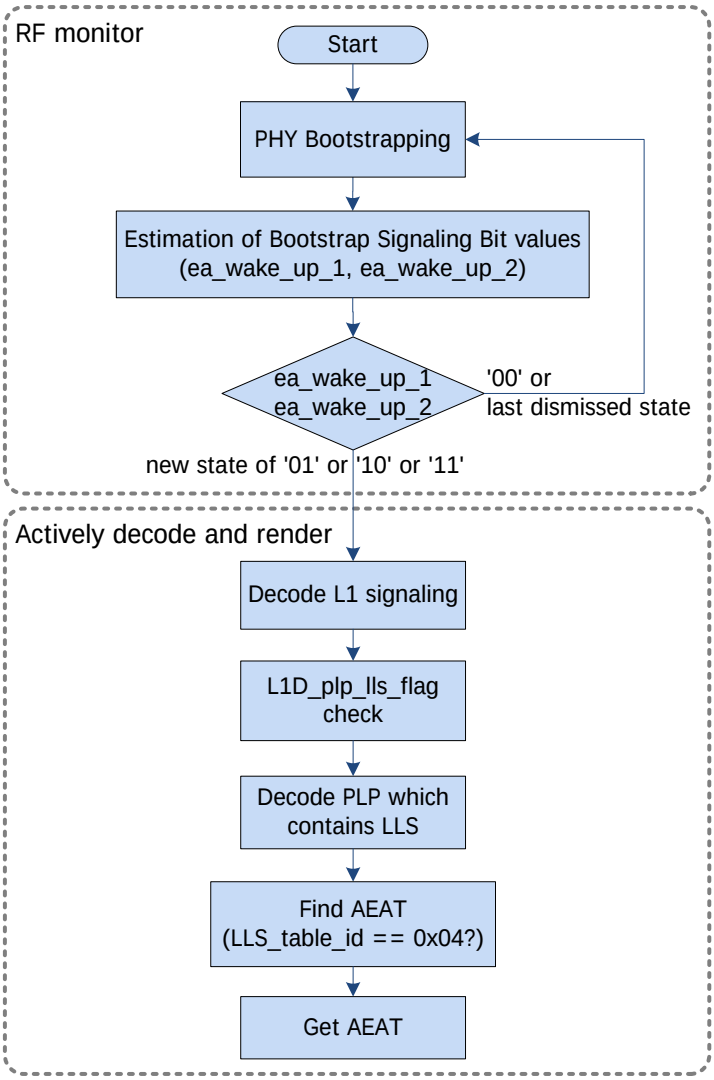


Figure G.1.1 A receiver behavior in two different states.

- Emergency Alert System (EAS) messages – normally broadcast today (2016) as "burned in" to the audio and video of a service. These messages may be received by TV stations as broadcast messages consisting of Frequency Shift Keyed (FSK) tones and audio, or as Common Alerting Protocol (CAP) messages – XML documents based on version 1.2 of the CAP specification [58].
- Broadcaster initiated emergency public information messages – urgent information the broadcaster desires to deliver to TV receivers that is separate or supplemental to an EAS message, or non-EAS emergency or urgent information.
- Emergency Alert Application (EAA) – A broadcaster application conforming to the ATSC 3.0 Interactive Content standard, A/344 [45], which can provide further information pertaining to the alert.
- Rich media files – content files that provide additional information about an emergency alert. These rich media files may be referenced from within the CAP message, or may be resources used by the EAA.

- Emergency-related programming – programming such as local news coverage that provides viewers with information about the ongoing emergency

More information about each of these is provided below.

Figure G.1.2 below illustrates the overall Emergency Alert System architecture in the United States.

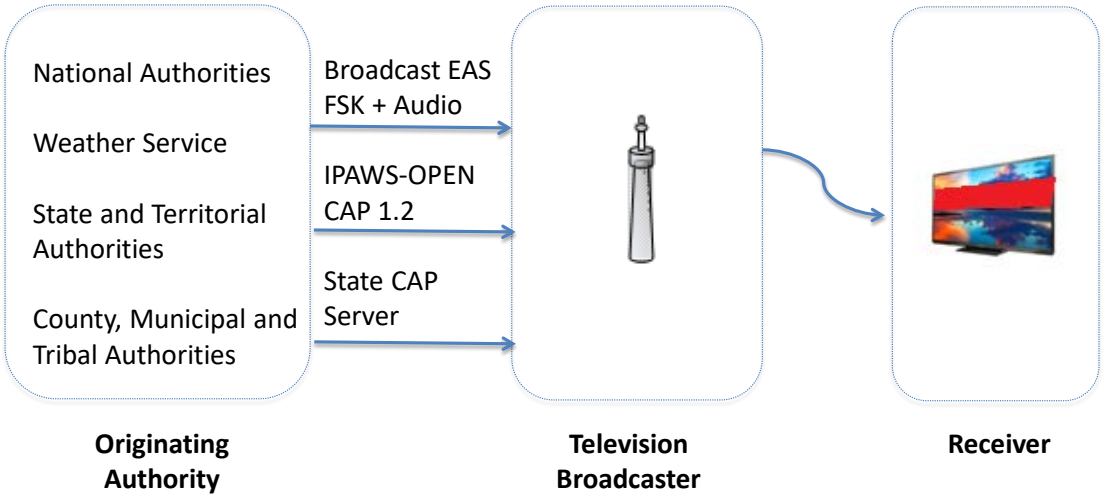


Figure G.1.2 High level architecture of the U.S. Emergency Alert system.

Emergency alert architectures can vary from country to country, and even region to region within a country. Figure G.1.3 and Figure G.1.4 below illustrate the differences in public alert and warning architectures in the United States and Canada, respectively.

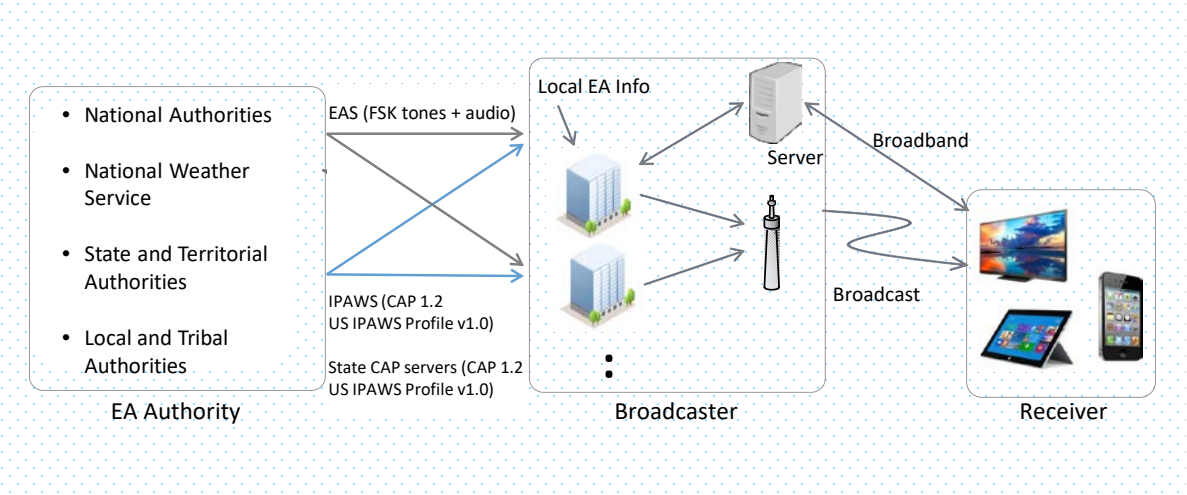


Figure G.1.3 Information Flows in the U.S. Emergency Alert System.

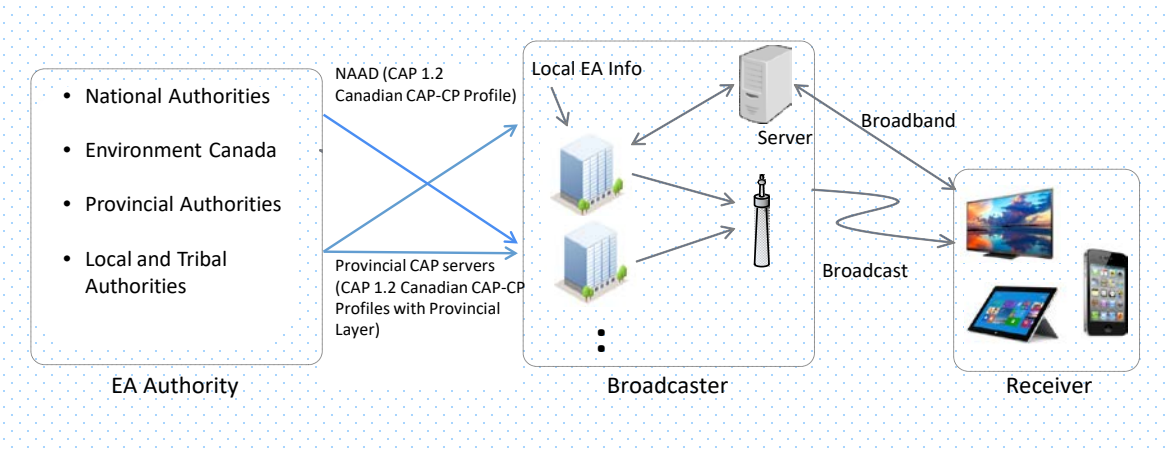


Figure G.1.4 Information Flows in the Canadian National Alert Aggregation and Distribution System.

In the U.S., as illustrated in Figure G.1.3, Alert Originating Authorities can provide Emergency Alert System (EAS) messages to broadcasters via the broadcast Emergency Alert System network, and/or they can provide CAP messages to broadcasters via IP-based CAP services, such as the FEMA IPAWS-OPEN system , and/or state CAP services. Broadcasters are required to convert CAP messages to EAS messages for presentation to viewers (voluntarily for most local event codes, mandatorily for national-level event codes).

In Canada, as illustrated in Figure G.1.4, Alert Originating Authorities can provide CAP alert messages to broadcasters via the National Alert Aggregation and Dissemination System (NAAD), or via provincial level CAP systems (mandatorily for alerts marked “Broadcast Immediate,” and voluntarily otherwise). Notably, the CAP XML format in Canada differs substantially from the CAP XML format used in the U.S.

Aspects common to any region include:

- EAS message presentation typically will consist of an “open” or “burned” in display generated at the station via character generator or media keyer.
- Broadcasters also can structure AEA messages for display as a “closed” message to receivers by extracting necessary information from CAP messages into an AEA-MF formatted message, or adding addition information and/or media resources into an AEA message. Multiple Alert elements from multiple sources can be merged into a single LLS table (AEAT) composed of multiple AEA messages.
- Broadcasters may add or modify information in AEA messages before delivering them.
- Broadcasters may also add rich media references to AEA messages and broadcast the rich media files as well as the AEA messages.
- Broadcasters may distribute and launch an Emergency Alert Application coincident with the alert in order to provide viewers with further information related to the event, optionally including interactive elements. Such further information could include school closing information, modifications to bus routes, locations of emergency shelters, or recommendations for dealing with the emergency.

G.1.2 Flow of Emergency Alert Signaling and Rich Media Contents

Figure G.1.5 below provides a more detailed view of example flows of emergency alert information, where the station has received an emergency alert message from an external (governmental) source.

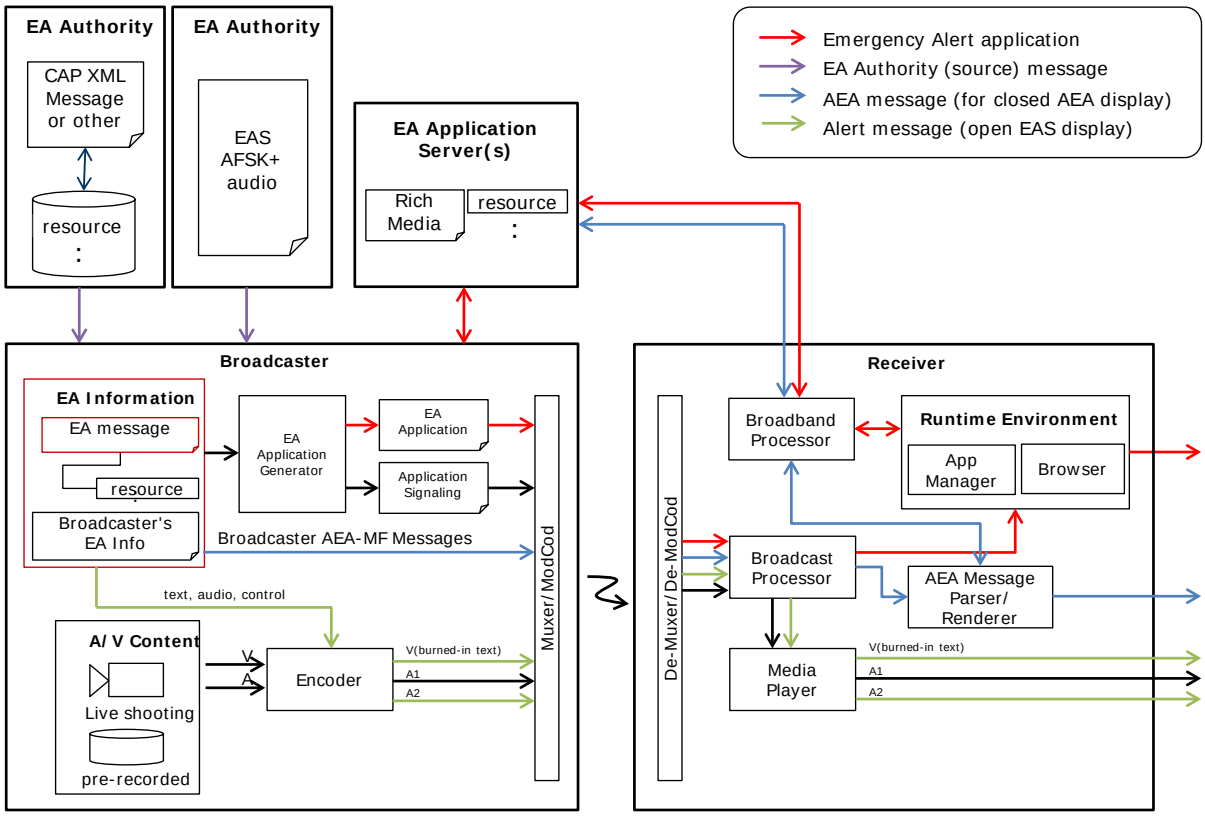


Figure G.1.5 Emergency Alert data flows.

In the example shown in the upper-left of Figure G.1.5, the Broadcaster has received an emergency alert event by means of a CAP message[10]. The broadcaster receives the emergency alert message via CAP and/or EAS FSK. With the exception of mandatory event types, the broadcaster has the option of not broadcasting a received alert message, or it may be passed on (after processing) within the broadcaster's program content to the public.

If the broadcaster decides to disseminate the contents of a particular incoming CAP message, after editing, it can be included in the broadcast multiplex (as "Broadcaster's AEA Message" in the figure. The AEA message may represent:

- the same textual content as in the conventional on-screen EAS display, as a companion to that alert;
- additional multimedia that may have been sent by the Originating Authority;
- additional textual content or multimedia inserted by the broadcaster, to supplement the conventional EAS display;

[10] Notification methods other than use of the CAP message are possible.

- transmission of the alert as an AEA-only, without the conventional on-screen EAS display.

The text of the AEA message may be edited within the broadcast plant before being passed to viewers, for example to add additional information. The broadcaster also has the option to add rich media elements such as graphics or even multimedia (e.g. video or audio clips).

The incoming AEA message may also be passed to an "EA Application Generator," which could generate a broadcaster application (or content for an existing application) to be distributed and used by the application runtime environment in the ATSC 3.0 receiver to provide an interactive experience related to the emergency.

The block labeled "Receiver" on the right side of Figure G.1.5 includes audio/video rendering and supports the ATSC 3.0 runtime application environment. As shown, depending on the format of the URLs referencing EA content, the rich media files can be provided either via broadband access to the EA App Server, or via broadcast delivery.

The block labeled "Application Signaling" can cause application resources such as media files usable by the application to be distributed. The URLs in the AEA message referencing rich media may be resolved by either the broadcast or the broadband path, depending on the broadcaster's choice for file distribution.

Green arrows in Figure G.1.5 represent emergency information included in video ("burned-in") or included in the audio program content.

G.1.2.1 Emergency Alert Message from External Sources

Figure G.1.6 below provides a more detailed view of example flows of emergency alert information, where the station is originating emergency content, from its own sources. This content could include emergency messaging, emergency news, multimedia, or other urgent content/information.

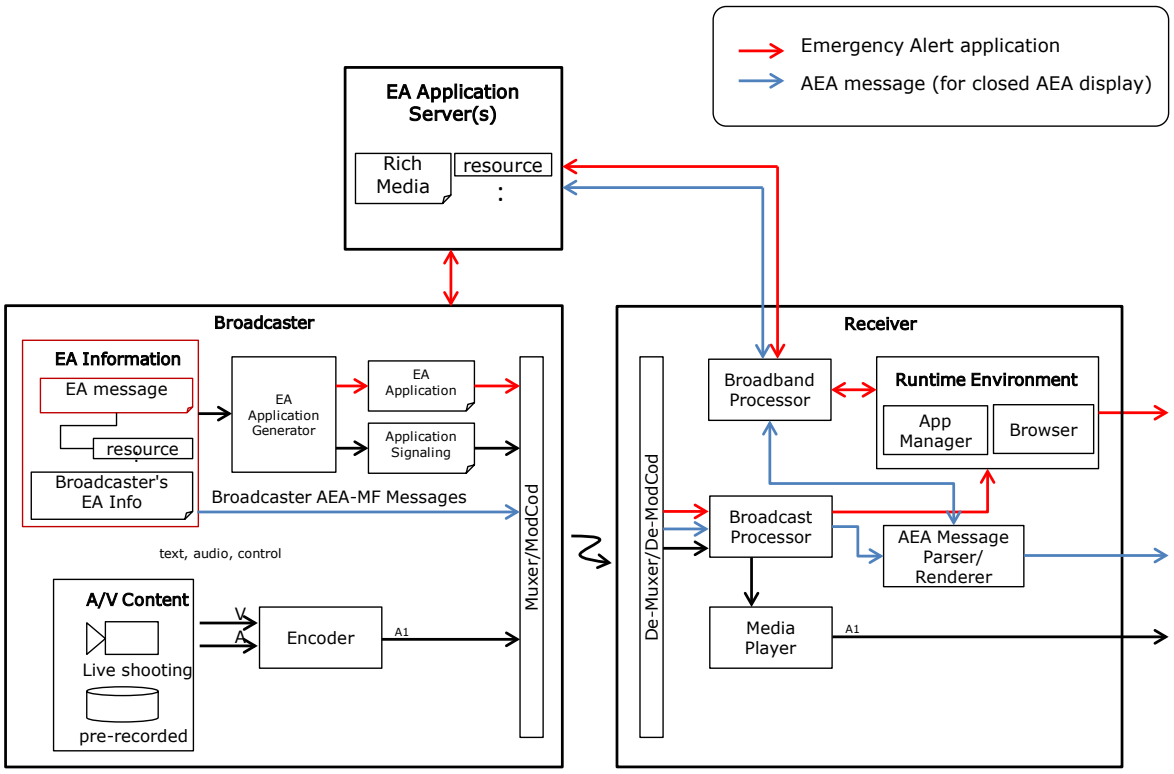


Figure G.1.6 Emergency Alert data flows.

In the example shown in Figure G.1.6, the Broadcaster is initiating an AEA to deliver non-EAS messaging to receivers. In this example, the Broadcaster creates an AEA, which may also include multimedia resources such as video or graphics. The emergency information is delivered to the receiver as an AEA message only ("closed" display) by the receiver, and is not delivered from the station as an "open" or "burned in" display.

G.2 MEANING OF WAKE-UP BITS

Two bits carried in fields ea_wake_up_1 and ea_wake_up_2 are delivered in Bootstrap Symbols 1 and 2, respectively, as described in A/321 [3], Section 6.1.1.1. Taken together, the two bits comprise a value labeled **EA Wakeup** herein. The value of **EA Wakeup** indicates one of four possible states, including three "positive" (i.e., non-zero) states, each of which indicates that at least one AEA message is active in the physical transmission channel in which the Bootstrap is carried. By examining the value of **EA Wakeup** and any transitions between values, it is possible to differentiate a new wake-up notification from a previous wake-up notification. The **EA Wakeup** values are delivered in such a way that a receiver can detect changes in them even when the receiver is not actively receiving content. These values allow receivers to detect that emergency information is available for presentation to users and also to detect a change of **EA Wakeup** value so that a wake-up event that was dismissed by a user can be distinguished from a new wake-up event (i.e., a new, but concurrent, emergency situation).

The two wake-up bits specified in A/321 [3], Section 6.1.1.1, are:

- ea_wake_up_1 within bootstrap_symbol_1() as specified in Table 6.2 of A/321
- ea_wake_up_2 within bootstrap_symbol_2() as specified in Table 6.4 of A/321

These two bits shall be concatenated to form a 2-bit value of **EA Wakeup**, with ea_wake_up_1 carrying the least-significant bit and ea_wake_up_2 carrying the most significant bit.

The meanings of the values of **EA Wakeup** shall be as specified in Table G.2.1.

Table G.2.1 Meaning of **EA Wakeup**

Value	Meaning
'00'	No emergency to wake up devices is currently signaled
'01'	Emergency to wake up devices - setting 1
'10'	Emergency to wake up devices - setting 2
'11'	Emergency to wake up devices - setting 3

When an AEA message with `AEA@wakeup="true"` is present, the value of **EA Wakeup** shall be non-zero. The value of **EA Wakeup** shall change when an AEA message with `AEA@wakeup="true"` is added. The **EA Wakeup** value may change when an AEA message with `AEA@wakeup="true"` is changed. When there are no AEA messages with `AEA@wakeup="true"`, the value of **EA Wakeup** shall be '00'.

When the value of **EA Wakeup** is non-zero, at least one AEA message with `AEA@wakeup="true"` shall be transmitted at least once per second or once per physical layer frame, whichever is less frequent, for the two minutes following an **EA Wakeup** value change. Thereafter, at least one AEA message with `AEA@wakeup="true"` shall be transmitted at least once every 10 seconds for the remainder of the period during which the AEA message continues to signal `AEA@wakeup="true"`.

Each Service is signaled in an LLS, and an LLS signals one or more Services as described in Section 6. There may be more than one LLS broadcast per RF channel, as described in Section 6.2. The ALP that carries the LLS describing a particular Service is required to be included in the set of ALPs that carry the components of that Service, as described in Section 5. When present, an AEA message is carried in an AEAT, which, in turn, is carried in LLS, as described in Section 6.5. Additionally, if an AEA message with `AEA@wakeup="true"` is carried in any AEAT (in any LLS), the **EA Wakeup** value is required to be non-zero and is required to be zero otherwise (see above).

The value of **EA Wakeup** does not represent a one-to-one correspondence to a particular emergency alert. Any new AEA message having `AEA@wakeup="true"` requires a change in the **EA Wakeup** value. Cancellation of the latest alert will not affect the **EA Wakeup** value so long as another alert that triggered a change in the **EA Wakeup** value remains active. Only when all events that triggered changes in the value of **EA Wakeup** have expired, or the broadcaster chooses to cancel the wake-up signaling, will the **EA Wakeup** value revert to zero.

If the user turns off a receiver while the **EA Wakeup** value is non-zero, it is expected that the receiver will remain off unless a new, non-zero **EA Wakeup** value is transmitted.

G.3 EMERGENCY ALERT SYSTEM OPERATION

In the alerting system currently deployed in the U.S., each EAS message that is burned into the audio/video programming starts with an audio tone to announce its presence. This is followed by text overlaid on the video, as a static or scrolled banner, and by audio either in a secondary audio channel or replacing the program audio in the audio track. The total duration of an EAS message is typically less than two minutes.

G.4 ADVANCED EMERGENCY ALERTING (AEA)

G.4.1 AEA Delivery

When an AEA-MF message is delivered in an AEAT, it shall be delivered in one or more Low Level Signaling (LLS) channels in the broadcast stream, with the delivery mechanism specified in Section 6.5 of the present document. If a broadcast stream is shared by multiple broadcasters and contains multiple LLS channels, each LLS channel may contain a AEAT with one or more AEA-MF, and AEA-MF messages in different channels may be the same or may be different. Each AEA-MF message as specified in Section 6.5 of the present document, contains the pertinent descriptive information about the emergency event, optional multimedia resources (described in G.4.2 below), and presentation information (incuding priority, intended audience, and if the alert should be delayed).

G.4.2 Rich Media Content

Rich Media content may be signaled via **`Media`** elements of a AEA-MF message, and when present they may be signaled in the SLT, and/or they may be delivered via broadband.

When a rich media resource is delivered via broadband, the **`Media`**`@url` attribute is formed as an absolute URL and it references a file on a remote server. When a rich media resource is delivered via broadcast ROUTE, the attribute is formed as a relative URL. The relative URL matches the `Content-Location` attribute of the corresponding **`File`** element in the Extended FDT Instance for the LCT [26] channel delivering the file, or the Entity header of the file.

As with other services, the SLT points to SLS signaling for the service where the content can be found.

When a broadcaster transmits an app to handle presentation of AEA-MF message information and the rich media resources signaled in an AEA-MF message, it can be done via an app-based enhancement that is signaled to be a part of every service in the broadcast stream. For details, see A/337, Application Signaling and Triggers [7] and A/344, Interactive Content [45]. The ROUTE session(s) that carry the rich media shall be included in the S-TSID for each service in the broadcast stream.

Annex H: Media Type Registrations

This Annex documents new media types registered by IANA at https://www.iana.org/assignments/media-types/media-types.xhtml#application.

> *Notice to editors*: any changes to this Annex are subject to review by IETF and IANA as described in IETF BCP 13 [51].

H.1 S-TSID

Type name:
application
Subtype name:
route-s-tsid+xml
Required parameters:
None.
Optional parameters:
charset
If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also "Encoding considerations" below.
Encoding considerations:
Same as for application/xml, except constrained to UTF-8. See IETF RFC 7303 [53], Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, "binary", applies.
Security considerations:
This media type inherits the issues common to all XML media types – see RFC 7303 [53] Section 10. This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors. This type does not employ executable content, but since it is explicitly extensible then executable content could appear in an extension. This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security, if needed.
Interoperability considerations:
The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document's namespace and in other namespaces.
Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.
Published specification:

This media type registration is an integral part of ATSC A/331, "Signaling, Delivery, Synchronization, and Error Protection", Annex H. The payload is defined in Section 7.1.4. This specification and XML schema for the content are available at www.atsc.org/standards (the schema(s) are provided in a zip file).

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.sls

Macintosh file type code(s):

"SLS1"

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

None

Author:

ATSC.

Change controller:

ATSC.

H.2 USD FOR ROUTE

Type name:

application

Subtype name:

route-usd+xml

Required parameters:

None.

Optional parameters:

charset

If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also "Encoding considerations" below.

Encoding considerations:

Same as for application/xml, except constrained to UTF-8. See IETF 7303, Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, "binary", applies.

Security considerations:

This media type inherits the issues common to all XML media types – see RFC 7303 [53] Section 10. This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors. This type does not employ executable content, but since it is explicitly extensible then executable content could appear in an

extension. This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security, if needed.

Interoperability considerations:

The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document's namespace and in other namespaces.

Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

Published specification:

This media type registration is an integral part of ATSC A/331, "Signaling, Delivery, Synchronization, and Error Protection", Annex H. The payload is defined in Section 7.1.3. This specification and XML schema for the content are available at www.atsc.org/standards (the schema(s) are provided in a zip file).

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.rusd

Macintosh file type code(s):

"RUSD"

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

None

Author:

ATSC.

Change controller:

ATSC.

H.3 USD FOR MMTP

Type name:

application

Subtype name:

mmt-usd+xml

Required parameters:

None.

Optional parameters:

charset

If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also "Encoding considerations" below.

Encoding considerations:

Same as for application/xml, except constrained to UTF-8. See IETF 7303, Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, “binary”, applies.

Security considerations:

This media type inherits the issues common to all XML media types – see RFC 7303 [53] Section 10. This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors. This type does not employ executable content, but since it is explicitly extensible then executable content could appear in an extension. This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security, if needed.

Interoperability considerations:

The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document’s namespace and in other namespaces.

Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

Published specification:

This media type registration is an integral part of ATSC A/331, “Signaling, Delivery, Synchronization, and Error Protection”, Annex H. The payload is defined in Section 7.2.1. This specification and XML schema for the content are available at www.atsc.org/standards (the schema(s) are provided in a zip file).

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.musd

Macintosh file type code(s):

"MUSD"

Person & email address to contact for further information:

Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)

Intended usage:

COMMON

Restrictions on usage:

None

Author:

ATSC.

Change controller:

ATSC.

H.4 APD

Type name:

application

Subtype name:

route-apd+xml

Required parameters:

None.

Optional parameters:

charset

If specified, the charset parameter must match the XML encoding declaration, or if absent, the encoding is determined from the XML document itself. See also "Encoding considerations" below.

Encoding considerations:

Same as for application/xml, except constrained to UTF-8. See IETF 7303, Section 9.1. For the purpose of filling out the IANA Application for Media Type, the value, "binary", applies.

Security considerations:

This media type inherits the issues common to all XML media types – see RFC 7303 [53] Section 10. This media format is used to describe broadcast and broadband services. This format is highly susceptible to manipulation or spoofing for attacks desiring to mislead a receiver about a session. Both integrity protection and source authentication is recommended to prevent misleading of processors. This type does not employ executable content, but since it is explicitly extensible then executable content could appear in an extension. This media type does not provide any confidentiality protection and instead relies on the transport protocol that carries it to provide such security, if needed.

Interoperability considerations:

The published specification describes processing semantics that dictate behavior that must be followed when dealing with, among other things, unrecognized elements and attributes, both in the document's namespace and in other namespaces.

Because this is extensible, conformant processors may expect (and enforce) that content received is well-formed XML, but it cannot be guaranteed that the content is valid to a particular DTD or Schema or that the processor will recognize all of the elements and attributes in the document.

Published specification:

This media type registration is an integral part of ATSC A/331, "Signaling, Delivery, Synchronization, and Error Protection", Annex H. The payload is defined in Section 7.1.7. This specification and XML schema for the content are available at www.atsc.org/standards (the schema(s) are provided in a zip file).

Applications that use this media type:

ATSC 3.0 television and Internet encoders, decoders and other facility and consumer equipment.

Additional information:

File extension(s):

.rapd

Macintosh file type code(s):

"RAPD"

Person & email address to contact for further information:
Editor, Advanced Television Systems Committee (jwhitaker@atsc.org)
Intended usage:
COMMON
Restrictions on usage:
None
Author:
ATSC.
Change controller:
ATSC.

— End of Document —