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

A/331:2025-10
16 October 2025

ATSC / 1300 I Street, N.W., Suite 400E / Washington, D.C. 20005 USA / +1-202-872-9160

ATSC, the Broadcast Standards Association, is an international, non-profit organization developing voluntary standards and recommended practices for broadcast television and multimedia data distribution. ATSC member organizations represent the broadcast, professional equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries. ATSC also develops implementation strategies and supports educational activities on ATSC standards. ATSC was formed in 1983 by the member organizations of the Joint Committee on Inter-society Coordination (JCIC): the Consumer Technology Association (CTA), the Institute of Electrical and Electronics Engineers (IEEE), the National Association of Broadcasters (NAB), the Internet & Television 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
A/331:2017 Standard approved	6 December 2017
A/331 Candidate Standard Revision approved	30 April 2018
Updated CS approved	23 February 2019
A/331:2019 approved	20 June 2019
Typo corrected on page 31: "…per Section 7.6 of DASH-IF [12]" changed to Section 7.5.	6 August 2019
Typo corrected on page 10: carriage return added before "UML"	29 August 2019
A/331:2020 (rollup of Amendments No. 1 through 9 to A/331:2019)	16 January 2020
Table 7.6 errata addressed	8 September 2020
Amendment No. 1 approved	27 October 2020
Amendment No. 2 approved	19 January 2021
A/331:2021 (rollup of Amendments No. 1 and No. 2 to A/331:2020)	19 January 2021
Amendment No. 1 approved	22 April 2021
Amendment No. 2 approved	27 May 2021
Amendment No. 3 approved	8 June 2021
Amendment No. 4 approved	1 October 2021
Amendment No. 5 approved	9 November 2021
A/331:2022-02 approved (Revision of A/331:2021 with Amendments No. 1, 2, 3, 4, and 5)	15 February 2022

A/331:2022-03 (references to ATSC documents updated)	31 March 2022
Typo and formatting error in Section 8.2.1.2.2 corrected	9 May 2022
Corrigendum No. 1 approved	22 September 2022
A/331:2022-09 (rollup of Corrigendum No. 1 to A/331:2022-03)	22 September 2022
A/331:0222-09 Amendment No. 1 approved	30 November 2022
A/331:2022-11 (rollup of Amendment No. 1 to A/331:2022-09)	30 November 2022
Corrected a typo in Table 6.11: @refAEAid should be @refAEAId “UTC” as defined in Section 3.3 adjusted to agree with term as defined by ITU-R	14 December 2022
A/331:2022-11 Amendment No. 1 approved	8 February 2023
A/331:2022-11 Amendment No. 2 approved	16 February 2023
A/331:2023-02 (rollup of Amendments No. 1 and 2 to A/331:2022-11)	16 February 2023
A/331:2023-03 (references to ATSC documents updated)	28 March 2023
A/331:2023-03 Amendment No. 1 approved	16 August 2023
A/331:2023-08 (rollup of Amendment No. 1 to A/331:2023-03)	16 August 2023
A/331:2023-08 Amendment No. 1 approved	4 October 2023
A/331:2023-10 (rollup of Amendment No. 1 to A/331:2023-08)	4 October 2023
A/331:2023-10 Amendment No. 1 approved	13 December 2023
A/331:2023-12 (rollup of Amendment No. 1 to A/331:2023-10)	13 December 2023
A/331:2024-04 (references to ATSC documents updated)	3 April 2024
Error corrected in Table 7.2, changing the footnote numbers from ‘1’ to ‘7’	13 January 2025
A/331:2024-04 Amendment No. 1 approved	13 February 2025
A/331:2025-02 (rollup of Amendment No. 1 to A/331:2024-04)	13 February 2025
Callout in Figure 8.12 corrected, formatting correction in Table 7.35 corrected, TOC updated	15 March 2025
A/331:2025-02 Amendment No. 1 approved	18 June 2025
A/331:2025-06 (a rollup of A/331:2025-02 Amendment No. 1)	18 June 2025
A/331:2025-06 Amendment No. 1 approved	10 September 2025
A/331:2025-06 Amendment No. 2 approved	10 October 2025
A/331:2025-06 Amendment no. 3 approved	16 October 2025
A/331:2025-07 (references to ATSC documents updated)	17 July 2025
A/331:2025-10 (a rollup of A/331:2025-06 Amendments No. 1, 2, and 3)	16 October 2025

Table of Contents

Index of Figures

Index of Tables

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 elements 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 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 – (deprecated)
- 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, DC, 7 August 2013.

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

[3] ATSC: "ATSC Standard: System Discovery and Signaling," Doc. A/321:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[4] ATSC: "ATSC Standard: Physical Layer Protocol," Doc. A/322:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[5] ATSC: "ATSC Standard: Service Announcement," Doc. A/332:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[6] ATSC: "ATSC Standard: Service Usage Reporting," Doc. A/333:2025-07, Advanced Television Systems Committee, Washington, DC, 17 April 2025.

[7] ATSC: "ATSC Standard: Application Event Delivery," Doc. A/337:2025-10, Advanced Television Systems Committee, Washington, DC, 14 October 2025.

[8] ATSC: "ATSC Standard: AC-4 System," Doc. A/342-2:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[9] ATSC: "ATSC Standard: MPEG-H System," Doc. A/342-3:2025-07 Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[10] ATSC: "ATSC Standard: ATSC 3.0 Security and Service Protection," Doc. A/360:2024-04, Advanced Television Systems Committee, Washington, DC, 3 April 2024.

[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 1.1," DASH Interoperability Forum, June 12, 2018.
https://dashif.org/guidelines/

[13] ETSI: "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] ETSI: "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, January 2016.
http://www.etsi.org/deliver/etsi_ts/126300_126399/126346/13.03.00_60/ts_126346v130300p.pdf

[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 4122 "A Universally Unique IDentifier (UUID) URN Namespace," Internet Engineering Task Force, Reston, VA, July 2005.
http://tools.ietf.org/html/rfc4122

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

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

[23] IETF: RFC 5234, "Augmented BNF for Syntax Specifications: ABNF," Internet Engineering Task Force, Reston, VA, July 2006.
http://tools.ietf.org/html/rfc5234.

[24] 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

[25] 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

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

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

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

[29] 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

[30] 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

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

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

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

[34] 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.

[35] 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.

[36] 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 14496-15:2022, International Organization for Standardization/ International Electrotechnical Commission, Geneva, Switzerland.

[37] 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.

[38] 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.

[39] 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.

[40] ISO/IEC: "Information technology – High efficiency coding and media delivery in heterogeneous environments – Part 3: 3D audio," Doc. 23008-3:2022. International Organization for Standardization/International Electrotechnical Commission, Geneva, Switzerland.

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

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

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

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

[45] ISO/IEC 23001-7:2016, "Information technology – MPEG systems technologies – Part 7: Common encryption in ISO base media file format files."

[46] DASH-IF: "Identifiers / Content Protection", DASH Industry Forum. https://dashif.org/identifiers/content_protection/
[47] DASH-IF: IOP V5, Part 6 "Content Protection" DASH Industry Forum. https://dashif.org/guidelines/iop-v5#part-6-content-protection-and-security
[48] IETF: RFC 4648, "The Base16, Base32, and Base64 Data Encodings," Internet Engineering Task Force, Reston, VA, October 2006. https://www.rfc-editor.org/info/rfc4648
[49] EIDR: "Introduction to EIDR Video Services Registry," The Entertainment ID Registry Association, v0.3, 2016/11/18, http://eidr.org/documents/Introduction_to_the_EIDR_Video_Services_Registry.pdf.
[50] IETF: RFC 4151, "The "tag:" URI Scheme" Internet Engineering Task Force, Reston, VA, October 2005, http://tools.ietf.org/html/rfc4151.
[51] ISO/IEC: 23090-3:2023 | Rec. ITU-T H.266 (9/2023), "Information technology — Coded representation of immersive media — Part 3: Versatile Video Coding," Geneva, Switzerland.
[52] ISO/IEC: 23094-2:2021, "Information technology — General video coding — Part 2: Low complexity enhancement video coding" Geneva, Switzerland.

2.2 Informative References

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

[53] ATSC: "ATSC Mobile DTV Standard, Part 3 – Service Multiplex and Transport Subsystem," Doc. A/153:2013, Advanced Television Systems Committee, Washington, DC, 29 October 2013.
[54] ATSC: "ATSC Standard: ATSC 3.0 Interactive Content," Doc. A/344:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.
[55] ATSC: "ATSC Standard: Regional Service Availability," Doc. A/200:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.
[56] ETSI: "Universal Mobile Telecommunications System (UMTS); LTE; Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH) (3GPP TS 26.247 version 13.3.0 Release 13)," Doc. ETSI TS 126 247 v13.3.0 (2016-08), European Telecommunications Standards Institute, August 2016. http://www.etsi.org/deliver/etsi_ts/126200_126299/126247/13.03.00_60/ts_126247v130300p.pdf
[57] IEEE: "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," Doc. IEEE 1588-2008, Institute of Electrical and Electronics Engineers, New York, NY.
[58] 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
[59] IETF: RFC 2365, "Administratively Scoped IP Multicast," Internet Engineering Task Force, Reston, VA, July 1998. http://tools.ietf.org/html/rfc2365

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

[61] 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

[62] 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

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

[64] 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.

[65] 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.

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

[67] 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

[68] 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

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

[70] ATSC: "ATSC Standard: Captions and Subtitles," Doc. A/343:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[71] ATSC: "Code Point Registry", Advanced Television Systems Committee, Washington, D.C.,
https://www.atsc.org/documents/code-point-registry/

[72] ATSC: "ATSC Standard: Dedicated Return Channel for ATSC 3.0," Doc. A/323:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

[73] ATSC: ATSC Recommended Practice: Techniques for Establishing and Maintaining Audio Loudness for Digital Television, with Corrigendum No. 1", Doc. A/85:2013, Advanced Television Systems Committee, Washington, DC, 12 March 2013, corrigendum No. 1 approved 11 February 2021.

[74] U.S. Army Corps of Engineers, Hydrologic Engineering Center, "CWMS Schema."
https://www.hec.usace.army.mil/xmlSchema/cwms/hec-datatypes.htm

[75] International Civil Aviation Organization, "World Geodetic System – 1984 (WGS-84) Manual," 2nd ed, 2002.
https://gis.icao.int/eganp/webpdf/REF08-Doc9674.pdf

[76] ATSC: "ATSC Recommended Practice: Digital Rights Management (DRM)," Doc. A/362:2025-07, Advanced Television Systems Committee, Washington, DC, 17 July 2025.

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.

ATSC Reserved indicates that only ATSC has currently defined values. Industry Reserved indicates that other SDO's have defined values and users are advised to consult the ATSC Code Point Registry [71].

3.3 Acronyms and Abbreviations

The following acronyms and abbreviations are used within this document.

2D – Two dimensional
3D – Three dimensional
3GPP – 3rd Generation Partnership Program
AEA – Advanced Emergency Alert
AEA-MF – AEA Message Format
AEAT – AEA Table
ALC – Asynchronous Layered Coding
AL-FEC – Application Layer Forward Error Correction
ALP – ATSC 3.0 Link-layer Protocol
APD – Associated Procedure Description
API – Application Programming Interface
ATSC – Advanced Television Systems Committee
AU – Access Unit
A/V – Audio/Visual
BA – Broadcaster Application
BCP – Best Current Practice
BEST – Broadcast-Enabled Streaming Television
BIP – Broadcast IP
BMFF – Base Media File Format
BSID – Broadcast Stream ID
bslbf – bit string, left bit first
CAP – Common Alerting Protocol
CCI – Congestion Control Information
CENC – Common ENCryption
CMS – Cryptographic Message Syntax
CP – Codepoint
CRI – Clock Relation Information
CSS – Cascading Style Sheet
CTA – Consumer Technology Association
CVS – Coded Video Sequence
DASH – Dynamic Adaptive Streaming over HTTP
DASH-IF – DASH Industry Forum
DDE – Data Delivery Event
DF – Don’t Fragment
DOI – Digital Object Identifier
DRM – Digital Rights Management
DTV – Digital TeleVision
DVB-T – Digital Video Broadcasting – Terrestrial
DVR – Digital Video Recorder
DWD – Distribution Window Description
EA – Emergency Alert
EAN – Emergency Action Notification

EAS – Emergency Alert System
EB_n – Elementary Stream Buffer (n^{th} instance)
EDT – Eastern Daylight Time
EFDT – Extended File Delivery Table
EIDR – Entertainment IDentifier Registry
EME – Encrypted Media Extensions
ERT – Expected Residual Time
ESG – Electronic Service Guide
ESR5 – Erroneous-Second Ratio with one (1) out of 20 seconds (5%) with errors
ETSI – European Telecommunications Standards Institute
FCC – Federal Communications Commission
FDD – File Delivery Description
FDT – File Delivery Table
FEC – Forward Error Correction
FIC – Fast Information Channel
FLUTE – File Delivery over Unidirectional Transport
FSK – Frequency Shift Keyed/Keying
GB – GigaByte(s)
GFD – Generic File Delivery
GoP – Group of Pictures
GPS – Global Positioning System
HD – High Definition
HDTV – HD TeleVision
HELD – HTML Entry pages Location Description
HEVC – High Efficiency Video Coding
HRBM – Hypothetical Receiver Buffer Model
HTML – Hypertext Markup Language
HTML5 – Hypertext Markup Language, 5^{th} major version
HTTP – Hypertext Transfer Protocol
HTTPS – Secure Hypertext Transfer Protocol
Hz – Hertz
IANA – Internet Assigned Numbers Authority
ID – Identifier or Identity
IDR – Instantaneous Decode Refresh
IEC – International Electrotechnical Commission
IEEE – Institute of Electrical and Electronic Engineers
IERS – International Earth Rotation and Reference Systems Service
IETF – Internet Engineering Task Force
IP – Internet Protocol
IPv4 – Internet Protocol version 4
IS – Initialization Segment
ISO – International Standards Organization

ITU-R – International Telecommunication Union Radiocommunication Sector
JS – JavaScript
LAN – Local Area Network
LCT – Layered Coding Transport
LLS – Low Level Signaling
LMT – Link Mapping Table
LTE – Long-Term Evolution
MA3 – MMT ATSC 3.0 Signaling Message
MAC – Media Access Control
MBMS – Multimedia Broadcast/Multicast Service
MDE – Media Delivery Event
MHTML – MIME Encapsulation of Aggregate Documents, such as HTML
MIME – Multipurpose Internet Mail Extensions
MMT – MPEG Media Transport
MMTP – MPEG Media Transport Protocol
MP – MMT Package
MPD – Media Presentation Description
MPEG – Moving Picture Experts Group
MPI – Media Presentation Information
MPT – MMT Package Table
MPU – Media Processing Unit
MSB – Most Significant Bit
NACK – Negative ACKnowledgment
NAL – Network Abstraction Layer
NORM – NACK-Oriented Reliable Multicast
NRT – Non-Real Time
NTP – Network Time Protocol
OASIS – Organization for the Advancement of Structured Information Standards
OTI – Object Transmission Information
PAT – MPEG-2 Program Association Table
PHY – PHYsical layer
PLP – Physical Layer Pipe
PSIP – Program and System Information Protocol
PTP – Precision Time Protocol
QoS – Quality of Service
RAP – Random Access Point
RF – Radio Frequency
RFC – Request for Comments
RMT – Reliable Multicast Transport
ROUTE – Real-Time Object Delivery over Unidirectional Transport
RRT – Rating Region Table
RSAT – Regional Service Availability Table

SBN – Source Block Number
SCT – Sender Current Time
SDO – Standards Developing Organization
SDP – Session Description Protocol
SGC – Standard Geographical Classification
SLS – Service Layer Signaling
SLT – Service List Table
S/MIME – Secure/Multipurpose Internet Mail Extensions
SMPTE – Society of Motion Picture & Television Engineers
S-TSID – Service-based Transport Session Instance Description
TAI – International Atomic Time
TB_n – Transport Buffer (n^{th} instance)
TCP – Transmission Control Protocol
T-MDE – Transport Media Delivery Event
TOI – Transport Object Identifier
TOL – Transport Object Length
TR – Technical Report
T-RAP – Transport Random Access Point
TS – 'Technical Specification' or 'Transport Stream'
TSI – Transport Session Identifier
TSID – Transport Stream Identifier
TV – TeleVision
UDP – User Datagram Protocol
UHD – Ultra High Definition
UML – Unified Modeling Language
UMTS – Universal Mobile Telecommunications System
URI – Uniform Resource Identifier
URL – Uniform Resource Locator
URN – Uniform Resource Name
USBD/USD – User Service Bundle Description / User Service Description
UTC –Coordinated Universal Time[1]
UTF-8 – Unicode Transformation Format – 8-bit
UUID – Universally Unique Identifier
VCT – Virtual Channel Table
W3C – Worldwide Web Consortium
WAN – Wide Area Network
XML – eXtensible Markup Language
XSD – eXtensible Markup Language Schema Definition
XSL – eXtensible Stylesheet Language

[1] This aligns with the term and acronym defined by ITU-R.

3.4 Terms

The following terms are used within this document.

App-Based Feature – Service 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.

BEST Service, **BIP Service** – A Broadband Service Configuration as defined in Section 8.2.1.2.2, where low-level signaling is delivered via broadcast and all Components are delivered over broadband.

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.

Broadband Configuration – A Service where all Components are delivered over broadband only (HTTPS).

Broadcast Configuration – A Service where all Components are delivered over ROUTE or MMT only.

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

Broadcaster Application – See A/344 [54].

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.

Component – Single media coding sequence of Segments of, e.g., video, audio, or captions.

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.

DRM Data Service – Service that delivers arbitrary data to specified DRM system.

ESG Server – As described in Section 6.9.

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

Essential Component – A Component required for a meaningful presentation.

Extended FDT Instance – An instance of the FLUTE [31] FDT that includes extensions. An Extended FDT Instance 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 [31].

(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 [54]. Filter Codes may be associated with files (see `File@filterCodes` in Section A.3.3.2.3.1).

GNSS – Any satellite constellation that provides positioning, navigation, and timing services, such as Galileo (EU), GPS (USA), GLONASS (Russia), BeiDou (China), and NavIC (India).

IP-ID – Identification field of IPv4 packet.

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. 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. 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.

long form content – As defined in ATSC A/85 [73], "Show or program related material or essence. The typical duration is greater than approximately two to three minutes."

Media Delivery Event (MDE) – 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.

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.

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

MPI message – MMT signaling message containing an MPI Table.

MPI Table – MMT table containing presentation information.

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.

Receiver – See A/344 [54].

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.

SATNAV – A global or regional infrastructure comprising a constellation of satellites, ground control stations, and user receivers, designed to provide real-time positioning, navigation, and timing (PNT) data anywhere within the system's coverage area.

Service – A collection of Components presented to the user in aggregate; Components can be of multiple media types; a Service can be either continuous or intermittent.

Service Configuration – One of the configurations listed in Section 5.3.

Service RAP –A point in the delivery of the data required to start a Service, at which point a receiver can acquire that Service with minimal delay.

Service Type – One of the types listed in Section 5.3.

short form content – As defined in ATSC A/85 [73], "Advertising, commercial, promotional or public service related material or essence. Also termed 'interstitial' content. The typical duration is less than approximately two to three minutes."

Signaling Server – As described in Section 6.9.

SLS (Service Layer Signaling) – Signaling which provides information for discovery and acquisition of ATSC 3.0 Services and their Components.

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

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.

S-TSID (Service-based Transport Session Instance Description) – An SLS XML fragment which provides the overall session description information for the transport session(s), i.e. LCT channel(s) of ROUTE session(s), which carry the 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 [44] 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 [31] 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 schemas defined in this document can be found at https://atsc-schemas.org/atsc3.0/a331/20240403/.

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 is shown 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 [38] 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 [65]. 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 feature, c) a file containing ESG or EA information, or d) a file containing (opaque) data to be consumed by a DRM system client 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).

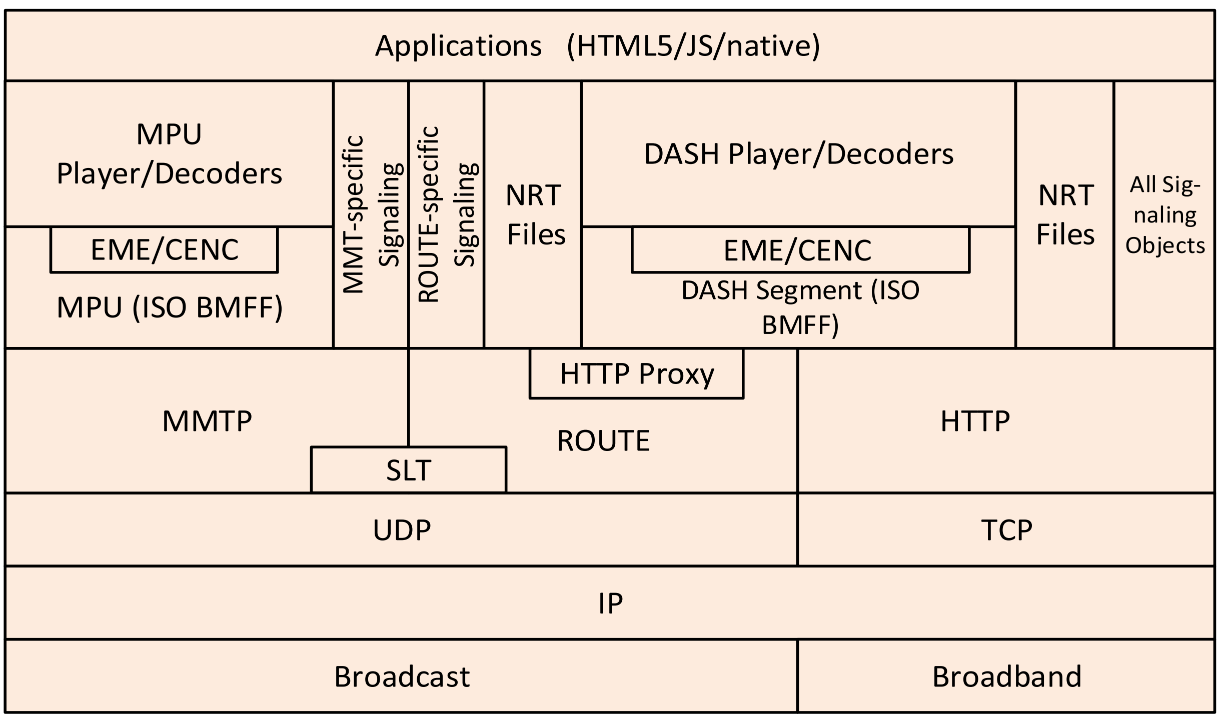

Figure 4.1 Conceptual protocol stack.

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) [35] 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 (Broadcast Configuration).
- 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 Broadband Configuration.
- 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 [38] 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 [32] 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.

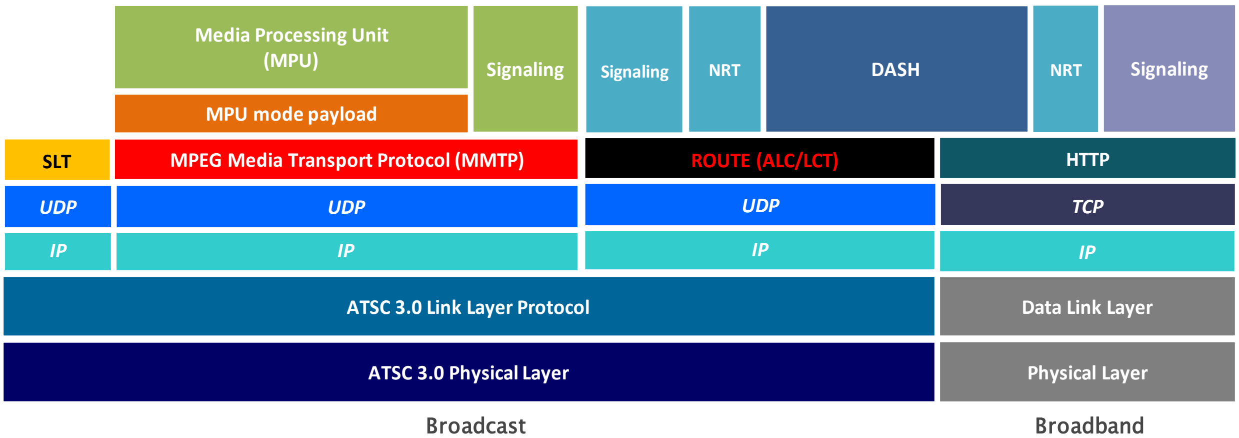

Figure 5.1 ATSC 3.0 receiver protocol stack.

Service Signaling provides Service discovery and description information, and comprises two functional elements: 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 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) are shown in Figure 5.2.

For ROUTE/DASH Services delivered over broadcast, the SLS is carried either on a Signaling Server or 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.

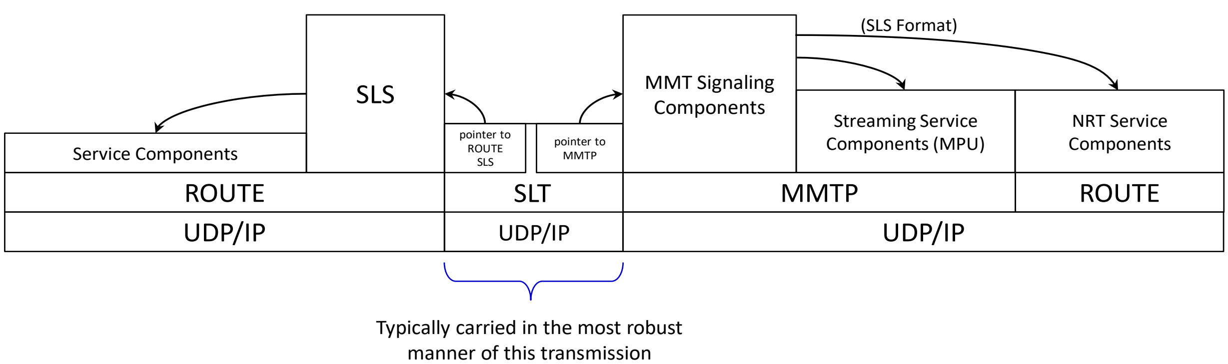

Figure 5.2 Service List Table references to Services.

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.

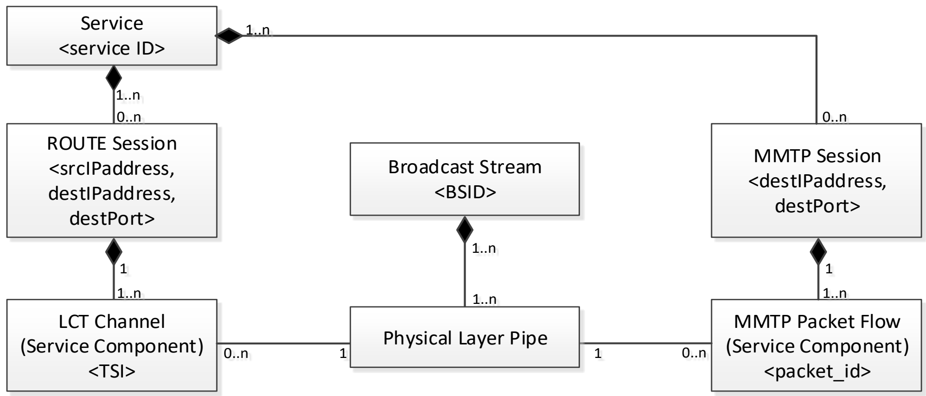

Figure 5.3 UML Diagram showing relationships among Service Management, Delivery, and Physical Layer entities.

5.3 Service Types and Service Configurations

The types of ATSC 3.0 Services that are currently defined are:

1) Linear Audio/Video Service
2) Linear Audio-Only Service
3) App-Based Service
4) ESG Service
5) (Deprecated)
6) DRM Data Service

These Service Types correspond to the values of `SLT.Service@serviceCategory`. New Service Types may be defined in future versions of this Standard. Properties of the above Service Types are shown in Table 5.1 to provide guidance in selecting the proper Service Type.

Table 5.1 Service Type Properties

Service Type	Video (HEVC)	Audio (AC-4)	Captions (IMSC1)	Held/Apps (A/344)	Undefined Data
Reserved (0)	X	X	X	X	X
Linear A/V (1)	M	M	O	O	X
Linear audio only (2)	X	M	O	O	X
App-Based (3)	O	O	O	M	X
ESG (4)	X	X	X	X	X
Deprecated (5)	X	X	X	X	X
DRM (6)	X	X	X	X	O
Data (7)	O	O	O	O	O

M – mandatory, O – optional, X – forbidden

The Service Configurations of ATSC 3.0 Services that are currently defined are:

1) Broadcast
2) Broadband

Note: Other configurations are under development; e.g., “hybrid.”

These Service Configurations correspond to the values of `SLT.Service@configuration`. New Service Configurations may be defined in future versions of this Standard. See Section 8.2.1.2 for more information. Service Configuration applies to the delivery of Components only, and does not apply to signaling or other data delivery (e.g., data from a Signaling Server or an ESG Server.

5.4 Rules Regarding ROUTE or MMTP Usage

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

a) For a broadcast delivery of a Linear Service without app-based feature, the Service's 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 feature, the Service's Components are carried by:
   - One or more ROUTE sessions, and
   - Zero or more MMTP sessions.

   Use of both MMTP and ROUTE for streaming Components in the same Service shall be disallowed.

c) For broadcast delivery of an app-based Service, an ESG Service, or a DRM Data Service, the Service's Components are carried by:
   - One or more ROUTE sessions.

d) For broadcast delivery of a Data Service the Service's Components are carried by:
   - One or more ROUTE sessions, or
   - One or more MMTP sessions.

Each ROUTE session comprises one or more LCT channels which carry as a whole, or in part, the 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 Component. An MMTP packet flow may carry MMT signaling messages or Components formatted per MMT [38] 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) 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.

5.5 Broadcast Streams

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.

A Service that consumes all available time interleaving memory as described in A/322 Section 7.1.2 [4] shall include LLS and SLS tables/fragments in at least one of the PLPs describing and carrying that Service. For example, a PLP with convolutional time-interleaving using the maximum allowed time-interleaving memory allocation additionally is required to carry one set of LLS and SLS tables/fragments for the Service(s) it is carrying. Similarly, for Service delivery via a group of hybrid time-interleaved PLPs using the maximum allowed time-interleaving memory allocation, the set of LLS and SLS tables/fragments for the Service(s) carried is required to be carried in one of the PLPs of that group.

There are a certain number of "groups" present in a given Broadcast Stream. A group is an abstraction of "broadcaster" or station. The identity of a group is expressed in signaling by a **LLS_group_id** defined in the LLS Table Format as shown in Table 6.1. Each Service in a Broadcast Stream is associated with a single **LLS_group_id** value. This group feature allows multiple stations to operate on a single RF channel with a substantial degree of independence, for example by each Broadcaster sending their Services under a unique group signaled in **LLS_group_id**.

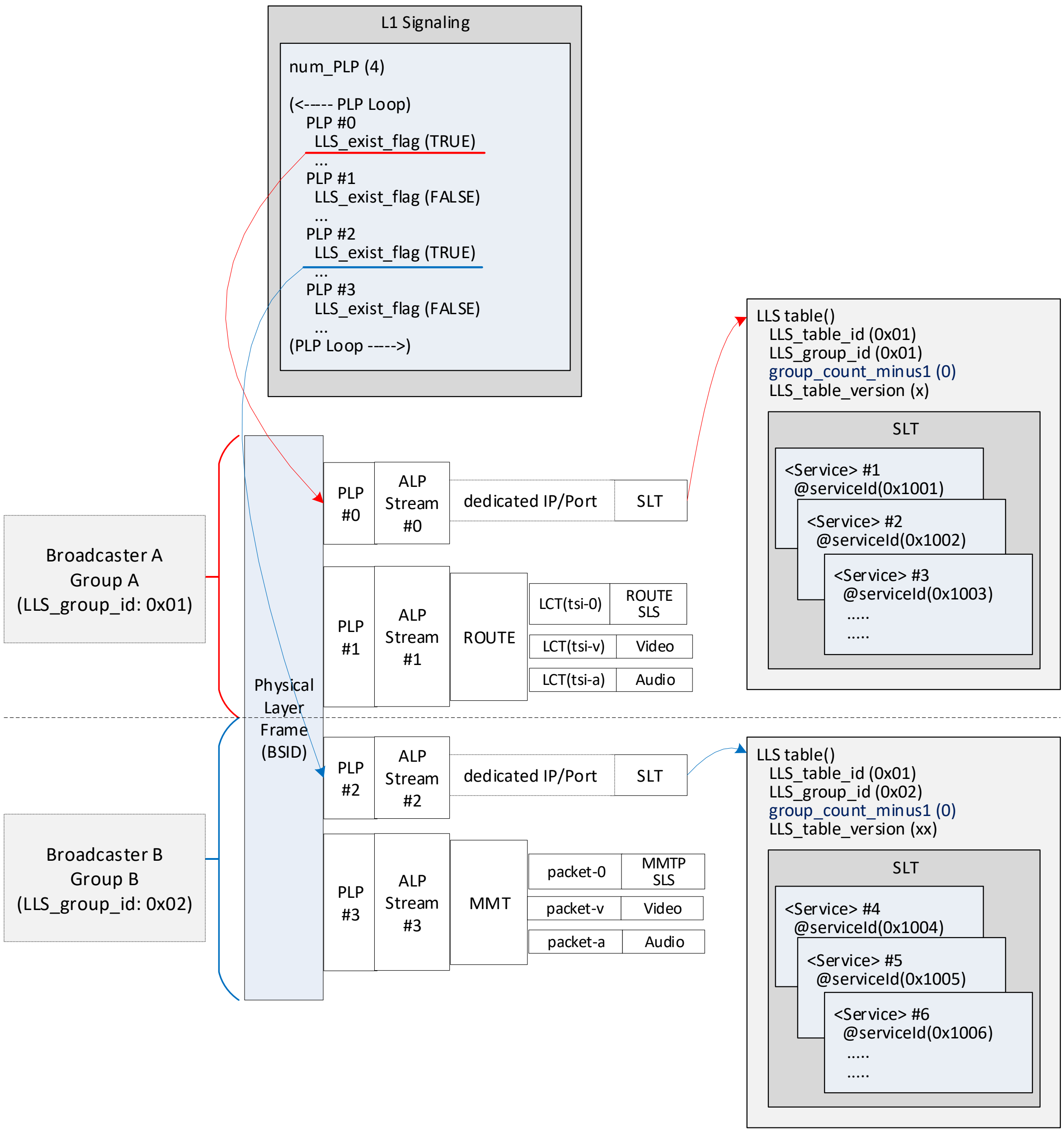

Figure 5.4 Example of **LLS_group_id** for group signaling.

5.6 Service Identifiers

Each Service is identified by two forms of Service identifier: a compact form that is used in the SLT that is unique within a single set of bonded PLPs, and a globally unique form that is used in 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 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.

An 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 that is unique within the scope of the parent MMTP session. Properties common to all MMTP packet flows, and certain properties of individual 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 related MMTP session.

5.7 Service Language Signaling

An important aspect of Service Signaling is the ability (and duty) to actively and appropriately signal the language(s) of each provided Service, including audio, captions, subtitles (if present), and any emergency Service.

Service language shall be declared for long form content only, not for short form content (the Service language signaled for short form content should be the Service language of the long form content). Short form content may be of a different spoken language than the long form content in some cases. (Examples might be an English spoken-language national advertisement inserted into a local Spanish spoken-language newscast or a Cantonese spoken-language advertisement inserted into a Mandarin spoken-language program.)

In modern audio systems, it is common to transport and make available Components affiliated with more than one language for a given long-form program. When this is the case, the language of each such Component shall be declared as that Component's primary, i.e., "native" language. Note that both of the transport mechanisms for ATSC 3.0 (MMTP/MPU and ROUTE/DASH) provide the ability for broadcasters to declare all of the available audio languages for the Components comprising a given streaming Service. This ability permits viewers who prefer, say, Mandarin to Cantonese to configure their receiving devices to automatically make that selection for them. It also permits broadcasters to always provide to their listeners specific spoken-language Services, even if "dubbed".

5.8 Service Delivery via Multiple RF Channels

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 in other RF channel(s). The **SLT.Service.OtherBsid** element is used to signal whether a Service is delivered in more than one RF channel. And the **OtherBsid**@type attribute is used to indicate the type of a set of Components of a Service, either "portion" or "duplicate".

Each Service represented by portions shall have one "essential" portion that is sufficient for a meaningful presentation of the Service without the use of the other portion(s), i.e., "non-essential" portion(s) (although also using the other portion(s) may provide a more appealing presentation). The **SLT.Service**@essential attribute is used to indicate whether a portion of a Service is either essential or non-essential.

5.8.1 Multiple RF Channel Service Delivery without Channel Bonding

A portion or a duplicate of a Service may be delivered in a single RF channel without channel bonding.

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 messages, defined in subclause 10.3.4 of ISO/IEC 23008-1 [38]) shall follow the signaling rules 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 appear. Each of these multiple listings of a Service, referencing its portions or duplicates, shall have the same value of Service ID, and except for OtherBsid@type=3, the same value of major/minor channel number. This consistency of values 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 also shall have one or more **OtherBsid** element(s) indicating the BSID(s) of the Broadcast Stream(s) in which the other portion(s) or duplicate(s) can be found.
- For ROUTE/DASH, an S-TSID shall be delivered in a PLP of 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.

5.8.2 Multiple RF Channel Service Delivery with Channel Bonding

A portion or a duplicate of a Service may be delivered in a pair of RF channels using 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 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 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 PLPs that deliver any portions of the Service shall list the Service. Each of these multiple listings of the Service, each referencing a particular 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 BSID(s) of the Broadcast Stream(s) where the other portion(s) can be found.
   - For ROUTE/DASH, S-TSIDs shall be delivered (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 session(s) and LCT channel(s) 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 session(s) and LCT channel(s) 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 BSID(s) of the Broadcast Stream(s) in which the other Service portion(s) 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, SLTs, S-TSIDs and MPT messages shall follow the signaling rules described below:

- SLTs associated both with non-bonded and with bonded PLP(s) that deliver duplicates of a Service shall list the Service. Each of these multiple listings of the Service referencing 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 BSID(s) (Broadcast Stream Identifier(s)) of the Broadcast Stream(s) in which the other duplicate(s) can be found.
- For ROUTE/DASH, S-TSIDs shall be delivered (a) in a single, non-bonded PLP of each RF channel that delivers a Service duplicate and (b) in a single, bonded PLP of each of the bonded RF channels that deliver a Service duplicate. In (a), each S-TSID instance shall describe the ROUTE session(s) and LCT channel(s) 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 session(s) and LCT channel(s) for each Component of the Service duplicate delivered by the corresponding bonded PLP(s) in each of the bonded RF channels to which the PLP carrying that S-TSID belongs.

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

Note 1: Multiple tuners are required to recover an ALP packet stream from bonded PLPs, i.e., the same number of tuners is required as the number of RF channels carrying the bonded PLPs.

Note 2: For a single-tuner receiver tuned to a channel with a bonded PLP, when the receiver is acquiring a Service carried by non-bonded PLPs, the receiver has to be able to acquire the SLT describing that Service. If the S-TSID of the 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.

5.9 Security Policies

ATSC 3.0 has a set of security tools defined in A/360 [10]. This section defines the usage policies for these tools in the ATSC 3.0 signal.

All HTTP connections defined in this document shall be secure bidirectional broadband IP connections as defined in A/360 Section 5.1.1 [10]. All uses of “HTTP” (for the URI scheme) in this document can be assumed to be “HTTPS” unless stated to the contrary.

Low Level Signaling (LLS) tables defined in this Standard (see Table 6.1) shall be transmitted with signatures (i.e., signed), using the method specified in A/360, Section 5.2.2.3 [10] using the LLS SignedMultiTable as defined in Section 6.7. Note that LLS tables can be defined in other specifications, e.g. A/360 CertificationData, TTA, and UserDefined. Their respective specifications are responsible for determining signing policy. In addition to the SignedMultiTable requirements above, LLS tables defined in this Standard may also be transmitted (as unsigned) outside the SignedMultiTable using their defined LLS_table_id. In all cases the SignedMultiTable is required. Some LLS tables have repetition-rate requirements (e.g., an SLT is required to be repeated at least once every 5 seconds). These repetition-rate requirements shall be met separately by each form (signed or unsigned) present in the stream (i.e., if both signed and unsigned forms of SLT are transmitted, a signed form is required at least once every 5 seconds, and an unsigned form is required at least every 5 seconds).

Except when delivered over secure bidirectional broadband IP conections using the mechanisms defined in A/360, Section 5.1.1 (HTTPS) [10], all ROUTE Service Layer Signaling (SLS) fragments defined in this Standard shall be signed using the method specified in A/360, Section 5.2.2.4 [10], and all MMT Service Layer Signaling (SLS) defined in this Standard shall be signed according to A/360, Section 5.2.2.5 [10]. Note that SLS tables can be defined in other specifications which are responsible for determining signing policy of the tables defined. Note also that SLS fragments delivered over secure bidirectional broadband IP connections could be signed, but are already required to be protected by the mechanisms of A/360, Section 5.1.1 [10].

Given that data in the ATSC 3.0 signal is signed, the CertificationData LLS table as defined in A/360, Section 5.2.2.2 [10] shall be present.

All executable code shall be signed using the method specified in A/360, Section 5.2.1 [10]. It shall be signed by both the author and the broadcaster.

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). The types of LLS information, each in the form of an LLS Table and defined in this Standard are: Service List Table (SLT), Rating Region Table (RRT), System Time fragment, Advanced Emergency Alert Table (AEAT), and Onscreen Message Notification fragment. Note that additional LLS tables exist as indicated in the ATSC Code Point Registry [71], including the Certification Data Table as defined in A/360 [10] and the Dedicated Return Channel Table as defined in A/323 [72].

6.1 IP Address Assignment

LLS shall be transported in IP packets with address `224.0.23.60` and destination port `4937/udp`.[2] 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[3],or (b) in the range of `239.255.0.0` to `239.255.255.255`[4], 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[5] 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;
- 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.

---

[2] IANA has assigned this multicast address to AtscSvcSig and this port address as atsc-mh-ssc. For details, see https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xml and https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[3] 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.

[4] This IP address range is the "IPv4 Local Scope," per RFC 2365 [59], Section 6.1.

[5] 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.

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 28 carrying Services with Virtual Channel Numbers 28.1 and 28.2, and also an ATSC 3.0 emission on RF channel 30 carrying Services with Virtual Channel Numbers 28.3 and 28.4, may use 28 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.

The address and port numbers of an IP/UDP stream carrying any Component of a Service shall change only upon a Service RAP of that Service.

6.1.1 Restriction on IP Fragmentation

The first IP packet sent after a Service RAP shall be a new packet, i.e., not fragmented across the Service RAP.

When IP header compression is used and IP fragmentation is not present, IP-ID shall be set to 0x00, and the DF field shall be set to 1. When IP header compression is used and IP fragmentation is either present or its presence is unknown, sequential IP-ID shall be used. When sequential IP-ID is in use, its value shall increase monotonically and shall wrap to 0x00 (16-bit) upon its reaching maximum. Random IP-ID shall not be used.

6.2 LLS Table Format

UDP/IP packets delivering LLS data shall be formatted per the bit stream syntax given in Table 6.1. As mentioned in Section 6, the list of LLS tables as shown in Table 6.1 is incomplete – see the ATSC Code Point Registry [71] for other valid LLS table types (and their associated LLS_table_id values). 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[6].

[6] 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	Section 6.3
break;		
case 0x02:		
RRT	var	See Annex F
break;		
case 0x03:		
SystemTime	var	Section 6.4
break;		
case 0x04:		
AEAT	var	Section 6.5
break;		
case 0x05:		
OnscreenMessageNotification	var	Section 6.6
break;		
case 0xFE:		
SignedMultiTable	var	Section 6.7
break;		
case 0xFF:		
UserDefined	var	Section 6.8
break;		
default:		
reserved	var	
}		
}		

LLS_table_id – An 8-bit unsigned integer that shall identify the type of table delivered in the body. LLS_table_id value 0x00 shall be ATSC Reserved, and undefined values shall be Industry Reserved. See the ATSC Code Point Registry [71]. Note: All ATSC-defined LLS table fragments can be extended with user-private information using standard XML techniques. See Section 3.6. Note: There can be multiple independent sets of LLS tables using groups of tables. See LLS_group_id.

LLS_group_id – An 8-bit unsigned integer that shall associate this instance of LLS_table() with a group of tables with the same LLS_group_id. The scope of this field is the broadcast stream. Note: Sources of groups of tables need to coordinate so as not to collide on this value. 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 groups of tables in the ALP packet stream carried in this PLP. A value of 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 0), 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].

SignedMultiTable – A set of one or more LLS table instances as defined in Section 6.7. **SignedMultiTable** itself is not compressed (e.g., with gzip). However, the contents of each table it contains is individually compressed, or not, as defined by each table.

UserDefined – A set of user-defined XML data as defined in Section 6.8 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 [34], and the Fast Information Channel (FIC) found in ATSC A/153, Part 3 [53]. For a receiver first encountering the broadcast emission, this is the place to start. The SLT 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 the SLT 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-1.0-20250604.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 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 SLS files available via broadband for all 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 DRM Data 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 Type, 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.
@hideInGuide	0,,1	boolean	Indicates whether the Service is shown in an ESG display.
@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.

@drmSystemID	0..1	listOfanyURI	Specifies the DRM System ID(s) related to this service.
@configuration	0..1	token	Declares the Service Configuration.
CodecStrings	0..1		Container for MPD Codecs strings
codecs	1 N	string	Delimited lists of MPD codecs strings per RFC6381 [30]
@kind	1	token	Type of codecs described
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 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 acquire ESG or SLS files available via broadband 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.
OtherRf	0..N		Supplies details about the transmission location and strength of other broadcasts which are indicated in SLT.Service.OtherBsid.
@OtherBsidRf	1	unsignedShort	Indicates the center channel RF frequency (in MHz) of the broadcast with the broadcast identifier of SLT.Service.OtherRf@OtherBsid.

@otherBsid	1	unsignedShort	Identifier of the Broadcast Stream broadcast on **OtherRf**. Linkage between SLT.Service.OtherBsid and SLT.Service.OtherRf is via the value of otherBsid.
@lat	1	float	Latitude of transmitter location (-90.0 ≤ lat ≤ 90.0).
@long	1	float	Longitude of transmitter location (-180.0 ≤ long < 180.0).
@elev	1	integer	Antenna radiation center Height Above Mean Sea Level (nearest meter).
@erp	1	integer	Antenna Effective Radiated Power (ERP), in kW.
Directional	0..N		Describes the directionality of the transmitter/antenna.
@heading	1	nonNegativeInteger	Heading in degrees, 0 ≤ @heading < 360.
@strength	1	float	The relative field value of the emission in the direction of @heading.
@haat	0..1	integer	The Height Above Average Terrain in the @heading direction.

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 white-space delineated 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. Note that for urlType=1 (SLS Signaling Server) this does not provide a mechanism to define how the Components referenced in the MPD are delivered. See Section 8.2.1.2 for more information.

Table 6.3 Code Values for `urlType`

`urlType`	**Meaning**
0	ATSC Reserved
1	URL of Signaling Server (providing access to the Service Layer Signaling, as specified in Section 6.9).
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 a single set of bonded PLPs. Note that there can be more than one set of tables in an RF channel.

`@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. When `@serviceCategory` is set to "Data", `@globalServiceId` shall be present and shall conform to a URL based on either EIDR Video Service IDs [49] or "tag:" as per RFC 4151 [50]. The form of EIDR is: https://doi.org/10.5239/[eidrVideoServiceID] where eidrVideoServiceID is the EIDR Video Service ID that corresponds to a Service registered with EIDR. The EIDR Video Service ID is an identifier in a database curated by EIDR. Video Service IDs begin with "10.5239".

The form of tag: shall be as tag:[DNSname],[year]:[uniqueString] where: DNSname is a registered domain name controlled by the broadcaster. This DNS should remain in the control of the broadcaster even if there is no site content. year is any year in which the broadcaster controlled the DNSname. uniqueString is a string of symbols that is unique to the broadcaster and used 1:1 to identify a Service. In order to use this core "tag:" syntax for other purposes by the broadcaster, it is recommended that the string begin with "`globalServiceID`", but any discriminator including none will satisfy the requirements in this section.

`@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.

`@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.

`@serviceCategory` – 8-bit integer that indicates the Service Type. The value shall be coded according to Table 6.4.

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

serviceCategory	Service Type
0	ATSC Reserved
1	Linear A/V Service
2	Linear audio only Service
3	App-Based Service
4	ESG Service (program guide)
5	(Deprecated
6	DRM Data Service (DRM Data)
7	Data Service
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.

`@hidden` – Boolean attribute that when present and set to `"true"` shall indicate that the Service is not accessed by the user by direct entry of Service or channel number. When `@hidden` is "`true`", Services are skipped when the user is "channel surfing", and appear as if undefined if accessed by direct Service virtual channel entry. Typical applications for hidden Services are test signals and NVOD services. Whether a hidden Service and its ESG data may appear in ESG displays depends on the state of the `@hideInGuide` attribute.The default value shall be `"false"` when not present.

`@hideInGuide` - A Boolean attribute that indicates, when set to "`false`", that the service and its ESG data may appear in ESG displays. This attribute shall be ignored for Services which do not have `@hidden` set to "`true`", so that non-hidden Services and their ESG data may always be included in ESG displays regardless of the state of this `@hideInGuide` attribute. Typical applications for hidden Services with the `@hideInGuide` set to "`true`" are test signals and Services accessible through application-level pointers, e.g., private broadcast services. 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.

@drmSystemID – This attribute identifies one or more DRM systems related to this service as a space-separated list of xs:anyURI. When @serviceCategory is "6" (DRM Data Service), this attribute is required and shall contain a single URI. This attribute shall be set to a URI identifying the DRM system ID associated with this DRM Data Service. The URI shall contain a UUID formatted as a URN, the same as the value of the @schemeIdUri used for the DASH MPD **ContentProtection** Descriptor, per Section 7.5 of DASH-IF [12]. The "urn:uuid:" prefix shall be included. More than one DRM Data Service may appear in a given SLT. The same value in the @drmSystemID attribute may be included in multiple instances of a Service identified as an DRM Data Service. When @serviceCategory is other than "6", this attribute is optional. When present, it shall contain one or more URIs. These URIs have the same syntax and semantics as for @serviceCategory="6" except that they identify one or more DRM systems that can decode the service. Receivers can, in advance of an MPD, initialize a supported DRM system. Receivers not supporting any of the DRM systems listed can choose not to offer the service to the viewer. Contrary to DASH-IF ATSC 1.1 [12], initialization segments should not contain any pssh boxes as per DASH-IF IOP V5, Part 6, 6.3 [47]) and Receivers may ignore such boxes when encountered.

@configuration – This attribute identifies the Service Configuration. It should be present always, but when not present the default Service Configuration can be determined as described in Section 8.2.1.2. When present it shall be one of the following string tokens:

- Broadband
- Broadcast

Note: Other configurations are under development; e.g. "hybrid."

CodecStrings – This element shall contain one or more Codecs elements indicating what codecs and versions are in use in the Service. The list indicates all the codecs that are used and if a Receiver can decode one codec in each value of kind, that is sufficient to decode the Service. A receiver might use this field to determine if it is capable of correctly decoding and presenting the content that is part of this Service with the available codecs; and perhaps whether the Service should be listed in a guide.

Codecs – This element describes the content kind and codec(s) and shall be a value from Table 6.5.

Table 6.5 Values for kind

kind	Meaning
video	The media signaled with the Codecs string are video media
audio	The media signaled with the Codecs string are audio media
captions	The media signaled with the Codecs string are captions media
other values	ATSC Reserved

SimulcastTSID – The content of this optional element is a 16-bit number that shall reference the TSID value of an ATSC 1.0 broadcast emission channel carrying the same programming content. The TSID value shall be set to the ATSC 1.0 VCT channel_TSID field as defined in

ATSC A/65 [1] Section 6.3. 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, when present. When 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, when present. When 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 6.6.

Table 6.6 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. Note that for urlType=1 (Signaling Server) this does not provide a mechanism to define how the Components referenced in the MPD are delivered. See Section 8.2.1.2 for more information.

`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 6.7. 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 **`Service`** element, the **`OtherBsid`**`@type` attribute values of all these elements shall be equal. When the value of `@type` is set to "3", at least one **`OtherRf`** element corresponding to **`OtherBsid`** shall be present.

Table 6.7 Code Values for **`SLT.Service.OtherBsid`**`@type`

type	**Meaning**
0	ATSC Reserved
1	Duplicate
2	Portion
3	MFN Cooperating/Appropriate Alternative service[1]
4-255	ATSC Reserved
Notes: [1] This standard makes no requirements on what constitutes an alternative Service to this Service.	

`OtherRf` – This optional element supplies details about the transmission location and strength of other broadcasts which are indicated in `SLT.Service.OtherBsid`. In the case where there are multiple BSID values (e.g., a bonded pair), there shall be one `OtherRf` field per BSID. In the case where there are multiple transmitters per BSID (e.g., a Single Frequency Network), there shall be one `OtherRf` field per transmitter. Note that the `OtherRf` element describes a single transmitter, and thus there can be one or more `OtherRf` elements per `OtherBsid`, and some services have multiple values in `SLT.Service.OtherBsid` (e.g., bonded channels) which yield one `OtherRf` data structure (e.g., one per BSID in the bonded set).

@OtherBsidRF – This unsigned short integer attribute shall indicate the center frequency in MHz of an RF emission which carries the Broadcast Stream identified by the OtherBsid value.

@OtherBsid – This unsigned short integer attribute shall indicate identifier of the broadcast stream that is described by this attribute. There is no default value when OtherBsid element is not present.

@lat – This attribute shall indicate the latitude of the location of the transmitter, on RF frequency @otherBsidRf. @lat shall be in the range of -90.0 to 90.0 indicating degrees and fraction of a degree (not degrees, minutes, seconds) of the latitude of a transmitter location. See "latitude-decimal-type" in [74] for context. Receivers might not utilize all precision supplied but are expected to utilize at least the first three decimal places of the value. Note that some regulatory agencies provide this data to the nearest tenth of a second.

@long – This attribute shall indicate the longitude of the location of the transmitter, on RF frequency @otherBsidRf. @long shall be in the range of -180.0 to 180.0 indicating degrees and fraction of a degree (not degrees, minutes, seconds) of the longitude of a transmitter location. Receivers might not utilize all precision supplied but are expected to utilize at least the first three decimal places of the value. Note that some regulatory agencies provide this data to the nearest tenth of a second.

@elev – This attribute shall indicate the antenna radiation center in Height Above Mean Sea Level to the nearest meter of the transmitter emitting on RF frequency @otherBsidRf.

@erp – This attribute shall indicate the Effective Radiated Power of the transmitter in kW on RF frequency @otherBsidRf.

Directional – This optional element describes the relative field strength of the broadcast emission at one or more compass points. Typically, this is due to a directional broadcast antenna. Note that if this field is provided, it should include at least 8 entries (e.g., N, NE, E, SE, S, SW, W, NW), and note also that regional regulatory authorities may provide this information (e.g., the in the U.S., the FCC regulatory database contains directional data at every 10°).

@heading – This attribute shall indicate the heading, in degrees, described by @strength. This attribute shall be in the range $0 \leq$ heading < 360.

@strength – This attribute shall indicate the relative strength of the signal in direction @heading.

@haat – This attribute, if present, shall indicate the Height Above Average Terrain, over the range from 3.2 km to 16 km from the transmitter, of the antenna radiation center to the nearest meter (this field may be a negative number), in the @heading direction. Note that these values (worldwide) can be obtained using the online calculator found at https://www.fcc.gov/media/radio/haat-calculator; for locations outside the U.S. the "GLOBE" database should be selected.

6.4 System Time Fragment

Partial samples of Physical Layer Time are delivered in the ATSC PHY layer as the least significant 32 bits of the number of seconds, a 10-bit fraction of a second (in units of milliseconds), and optionally 10-bit microsecond and nanosecond elements, since January 1, 1970 00:00:00, International Atomic Time (TAI), which is the Precision Time Protocol (PTP) epoch as defined in IEEE 1588 [57]. The most significant 16 bits of the number of seconds are carried as described below (@ptpPrepend in Table 6.8). 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-1.0-20211208.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 at least once every five seconds.

While the indicated XML schema specifies the normative syntax of the **SystemTime** element, informative Table 6.8 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 6.8 SystemTime Element Structure

Element or Attribute Name	Use	Data Type	Description
SystemTime			
@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 (which can be in the current calendar month or the following calendar month) on which the transition into or out of daylight saving time is to occur.
@dsHour	0..1	unsignedByte (range 0..23)	Indicates the local hour at which the transition into or out of daylight saving time is to occur (0–23).

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 [58] 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. PTP Seconds refers to the seconds portion of the time distributed by the PTP timescale and as noted is referenced to 1 January 1970 TAI. 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 time 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. In nearly all places, when daylight saving time is in effect, the local time is one hour later than when daylight saving time is not in effect. This field shall be included whenever daylight saving time is in effect at the transmitter location. The 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 at some time during the next month's time (defined to be the number of days in the current calendar month.) `@dsDayOfMonth` shall be included whenever there will be a daylight saving time transition during the next month's time. `@dsDayOfMonth` shall be set to the local day of the month on which such transition is to occur. `@dsDayOfMonth` shall be omitted whenever there will not be a daylight saving time transition during the next month's time. 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 (0–23), 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 6.9 illustrates the progression of states of signaling daylight saving time throughout the year.

Table 6.9 Daylight Saving Time Signaling Throughout the Year

Conditions	dsStatus	dsDayOfMonth	dsHour
At the beginning of the year (January) daylight saving time is off.	not present ("false")	not present	not present
This is the status of the fields until:			
The transition into daylight saving time is less than one month's time away, at which time dsDayOfMonth takes the value day_in, dsHour takes the value hour_in, and dsStatus is not present, indicating it is not yet daylight saving time. (The transition is to occur on the day_in day of this month or the following 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, so long as the current date is March 16 or later.)	not present ("false")	day_in	hour_in
This is the status of the fields until:			
The daylight saving time transition has occurred, at which time dsStatus is present and set to "true", indicating that daylight saving time is on, and attributes dsDayOfMonth and dsHour are not present.	"true"	not present	not present
This is the status of the fields until:			
The transition out of daylight saving time is less than one month's time away, at which time dsDayOfMonth takes the value day_out, dsHour takes the value hour_out, and 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 this month or the following 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, so long as the current date is September 28 or later.)	"true"	day_out	hour_out
This is the status of the fields until:			
The daylight saving time transition, at which time dsStatus takes the value "false" (or is not present), indicating that daylight saving time is off. Attributes dsDayOfMonth and dsHour are not present.	not present ("false")	not present	not present
This finishes the cycle.			

6.5 Advanced Emergency Information Table

The AEAT (Advanced Emergency Information Table) is one of the instances of LLS information. The AEAT is composed of one or more AEA (Advanced Emergency Information) messages. An AEA message is formatted in the Advanced Emergency Information-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 use 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 broadcaster 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 definition of this schema is in an XML schema file, AEAT-1.0-20210322.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 6.10 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 6.10 AEAT Element Structure

Element or Attribute Name	Use	Data Type	Short Description
AEAT			Root element of the **AEAT**
@aeaTableId	0..1	string	The identifier of the AEAT table.
AEA	1..N		The identifier of the start of a specific Advanced Emergency Information message within AEAT 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.
@subAudience	0..1	aeat:listOfStrings	Subgroup for non-public messages.
@aeaType	1	string	The type 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 shall not appear when the @aeaType is “cancel”.
@category	0..1	string	A code identifying the event category of the AEA message.
@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	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".
@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". It is optional 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 that 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 **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.

AEAT@aeaTableId – This attribute, if present, shall be a string value uniquely identifying an AEAT (table) within a given RF channel (or within the data delivered by bonded RF channels). The @aeaTableId shall be restricted to the 62 alphanumeric characters using the ASCII character set of 0x30 through 0x39, 0x41 through 0x5A, 0x61 though 0x7A, the dash (0x2D), dot (0x2E) and underscore (0x5F) characters. This element is used to identify a given version of an incremented AEA Table. Any change in the presence or value of an **AEA@aeaId** will cause a change in the value of the **AEAT@aeaTableId**.

AEA – Advanced Emergency Information 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**. Note that in the case of @aeaType of "cancel", the **Header**, **AEAText**, **Media**, **LiveMedia** and **Media** elements shall not be present.

AEA@aeaId – This attribute shall be a string value uniquely identifying the AEA message within a given RF channel (or within the data delivered by bonded RF channels). This attribute shall be restricted to the 62 alphanumeric characters using the ASCII character set of 0x30 through 0x39, 0x41 through 0x5A, 0x61 though 0x7A, the dash (0x2D), dot (0x2E) and underscore (0x5F) characters. This element is used to associate updates or cancels 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 6.11.

Table 6.11 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@subAudience – The string(s) shall identify the intended subgroup(s) for restricted or private audiences. The union of the @subAudience strings shall describe the audience for the message. The value(s) shall be defined as agreed by the broadcaster and the restricted entity (subAudience) that uses it. As such, the general public will not be aware of this information. This field shall appear when @audience is "restricted" or "private". For example, this field might contain a string which describes all freshmen ("grade9"), and a string which describes all sophomores ("grade10"). Together those string values would include the audience consisting of all students within both 9th and 10th grades (the union of 9th and 10th graders).

AEA@aeaType – This string shall identify the category of the AEA message. The value shall be coded according to Table 6.12.

Table 6.12 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 a previous emergency alert message. In this case, @refAEAId shall appear.
"cancel"	Indicates that AEA message is cancelling a previous emergency alert message. In this case, @refAEAId shall appear, and all attributes of the **AEAT.AEA.Header**, **AEAText**, **LiveMedia**, and **Media** elements shall not be present.
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 6.13. When it is not present, there is no default value.

Table 6.13 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 for 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@category – This attribute shall identify the event category of the AEA message formatted as a string denoting the value itself. When category is not present, there is no default value. When the value of **AEA**@audience is "public", then the value for **AEA**@category shall be from the values in Table 6.14.

Table 6.14 Code Values for `AEAT.AEA@category`

Category	Meaning
"ADVISORY"	Urgent Advisory (e.g., from a newsroom) that does not fit into the categories below.
"HEALTH"	Health Related (including environmental), including infectious disease information (e.g. COVID instructions), pollen and other environmental health hazards.
"WEATHER"	Urgent meteorological information and updates.
"EMERGENCY"	General emergency and public safety information, including fire, geological and other situations.
"SCHOOL"	Urgent information related to schools/educational institutions, including school closures, reopenings, in-school emergencies, and other important information.
"COMMUNITY"	Urgent information of local relevance, including utility/infrastructure issues, sanitation, government closures, etc.
"TRANSIT"	Urgent Transit/Transport/Traffic information, such as road closures, major traffic incidents, public transit delays, etc.
"OTHER"	Messages not categorizable above, such as test or "heartbeat" messages.
Other	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`). Messages of type "`cancel`" shall not contain the `Header` element.

`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"` (because when `@aeaType` is `"cancel"`, the message "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)).

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 [33]. 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 [67] 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 [69]. 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 area described by three or more vertices, expressed in latitude/longitude pairs that form a closed, non-self-intersecting loop. Each coordinate pair shall be expressed in decimal degrees. Latitude shall be in the range of -90.0 to 90.0 indicating degrees and decimal fractions of a degree (not degrees, minutes, seconds) of the latitude of each vertex. Longitude shall be in the range of -180.0 to 180.0 indicating degrees and decimal fractions of a degree (not degrees, minutes, seconds) of the longitude of each vertex. Receivers might not utilize all precision supplied but are expected to utilize at least the first three decimal places of latitude and longitude. The value of Location shall conform to the syntax described in the following ABNF for "polygon-coordinates":

```
; !syntax("abnf")
; Polygon coordinates: requires 3 or more coordinate pairs, each separated by a space.
; Example: "45.7606,4.8351 40.7128,-74.0060 -34.9293,-58.3754"
polygon-coordinates = coordinate-pair 2*(" " coordinate-pair)

; Latitude and Longitude in decimal degrees format, separated by a comma
; Examples: 37.7749,-122.4194   or   +37.7749,-122.4194
coordinate-pair = latitude "," longitude
```

```
; -90 through +90
latitude        = [ "-" / "+"] [DIGIT08] DIGIT ["." fraction] ; 0.fraction to 89.fraction
                / [ "-" / "+"] DIGIT9 DIGIT0 ; 90 exactly only


; -180 through +180
longitude       = [ "-" / "+"] [DIGIT0] 1*2DIGIT ["." fraction]     ; 0.fraction to 99.fraction
                / [ "-" / "+"] DIGIT1 DIGIT07 DIGIT ["." fraction]  ; 100.fraction to 
179.fraction
                / [ "-" / "+"] DIGIT1 DIGIT8 DIGIT0                 ; 180 exactly


fraction        = 1*DIGIT

DIGIT0          = %x30          ; 0
DIGIT1          = %x31          ; 1
DIGIT           = %x30-39       ; 0-9
DIGIT07         = %x30-37       ; 0-7
DIGIT08         = %x30-38       ; 0-8
DIGIT8          = %x38          ; 8
DIGIT9          = %x39          ; 9
```

- If `@type="circle"`, then the Location shall define a circular area represented by a central point given as a latitude/longitude coordinate pair subject to the requirements above in "`polygon`", followed by a space character and a positive radius value in miles (indicated by a '`m`' suffix) or kilometers (indicated by a required '`km`' suffix). The value of **`Location`** shall conform to the syntax described in the following ABNF for "circle-coordinates":

```
; !syntax("abnf")
; Circle coordinates: requires one coordinate pair, a space, then a radius value.
; Examples: "45.7606,4.8351 10.5m", "17.856556,-77.047607 25km", "56.368,53.306 17.5m"
circle-coordinates = coordinate-pair " " radius

radius  = radius-miles
        / radius-km

radius-miles = 1*DIGIT ["." fraction] "m" ; miles and fraction (positive value)
radius-km = 1*DIGIT ["." fraction] ["km"] ; km and fraction (positive value) (default)

; Latitude and Longitude in decimal degrees format, separated by a comma
; Examples: 37.7749,-122.4194   or   +37.7749,-122.4194
coordinate-pair = latitude "," longitude

; -90 through +90
latitude        = [ "-" / "+"] [DIGIT08] DIGIT ["." fraction] ; 0.fraction to 89.fraction
                / [ "-" / "+"] DIGIT9 DIGIT0 ; 90 exactly only


; -180 through +180
longitude       = [ "-" / "+"] [DIGIT0] 1*2DIGIT ["." fraction]     ; 0.fraction to 99.fraction
                / [ "-" / "+"] DIGIT1 DIGIT07 DIGIT ["." fraction]  ; 100.fraction to 
179.fraction
                / [ "-" / "+"] DIGIT1 DIGIT8 DIGIT0                 ; 180 exactly


fraction        = 1*DIGIT

DIGIT0          = %x30          ; 0
DIGIT1          = %x31          ; 1
DIGIT           = %x30-39       ; 0-9
DIGIT07         = %x30-37       ; 0-7
DIGIT08         = %x30-38       ; 0-8
DIGIT8          = %x38          ; 8
DIGIT9          = %x39          ; 9
```

Note: The ABNF above is available for download at the atsc-schemas.org repository: https://atsc-schemas.org/atsc3.0/a331/20250910/

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. When the message is of type "`cancel`", there is no expectation that **`AEAText`** will be presented to the audience.

`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 [33], 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 to 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. Message of type "`cancel`" shall not contain the **`LiveMedia`** element.

`LiveMedia``@bsid` – This list of one or more unsigned 16-bit integer values shall indicate identifier(s) of the Broadcast Stream(s) which contain the essential portions of the emergency-related live A/V Service. When the value of **`LiveMedia`**`@bsid` is a list of more than one unsigned 16-bit 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., "WZYX 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 [33]. 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. It 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 BA that is subscribed to AEATs (see A/344 [54] for subscribe AEAT API) using the AppContextIdList in the Extended FDT Instance (**`EFDT.FDT-Instance`**`@appContextIdList`), see Section 9.2.10 of A/344 [54]. Messages of type "`cancel`" shall not contain the **`Media`** element.

`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 [33], 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 that provides context for a viewer to understand what a given media file contains. For example, "Evacuation map of Jones County as of 1/1/2019" or "Doppler radar image related to Hurricane Sandy", etc. The language of the **Media**@mediaDesc shall be inferred to be the 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. When 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 6.15. When **Media**@mediaType is not present, there is no default value.

Table 6.15 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. As a relative URL, the value of **Media**@url shall match the @Content-Location attribute of the corresponding **File** element in the Extended FDT Instance in the LCT [27] channel delivering the file, or the Entity header of the file, and its format complies with that of the Content-Location URI as described in Section 8.1.1.2.

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 [32]. 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/OSMN/1.0/`

The definition of this schema is in an XML schema file, *ONSCREEN-1.0-20210401.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 6.16 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 6.16 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	listOfUnsignedShort	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 space-delimited list of one or more 16-bit identifiers shall indicate the BSID(s) of the Broadcast Stream(s) 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. When @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. When @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`**. When 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. When 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. When 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 SignedMultiTable

SignedMultiTable comprises a list of LLS tables. Each iteration through the **LLS_payload_count** loop contains a single LLS table type (e.g., SLT, RRT), followed by a signature.

The syntax of this multi-table shall be according to Table 6.17.

Table 6.17 SignedMultiTable syntax

Syntax	No. of Bits	Format
SignedMultiTable() {		
LLS_payload_count	8	uimsbf
for (i=0; i<LLS_payload_count; i++) {		
LLS_payload_id	8	uimsbf
LLS_payload_version	8	uimsbf
LLS_payload_length	16	uimsbf
LLS_payload()	var	
}		
signature_length	16	uimsbf
signature()	var	uimsbf
}		

LLS_payload_count – An 8-bit unsigned integer that shall list the number of LLS tables in the loop defined in Table 6.17 above.

LLS_payload_id – An 8-bit unsigned integer that shall indicate the payload type (values of **LLS_table_id**) of the **LLS_payload()** field. Valid values are from the values defined for **LLS_table_id**. The values 0x00 and 0xFE shall not be used.

LLS_payload_version – An 8-bit unsigned integer that shall be defined as **LLS_table_version** for this payload.

LLS_payload_length – A 16-bit unsigned integer that shall indicate the length, in bytes, of the **LLS_payload()** field.

LLS_payload() – The payload as signaled by the **LLS_payload_id** value (e.g., SLT, RRT). See **LLS_table_id**.

signature_length – A 16-bit unsigned integer that shall indicate the length, in bytes, of the **signature()** field.

signature() – This field shall be CMS Signed Data as per A/360 [10], Section 5.2.2.3. The signature shall be computed over the **LLS_payload_count** field up to but not including the **signature_length** field.

6.8 UserDefined

LLS_table_id 0xFF is available for user-defined usage. This table shall be an XML document that is defined by any user of this Standard and shall have at least one namespace not defined by ATSC.

6.9 Signaling Server

The Signaling Server provides HTTPS Secure Connection (per the Security Policies of Section 5.9) access to the Service Layer Signaling (SLS) tables defined in Section 7.

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 current signaling metadata. The desired signaling metadata objects 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.

When an **SLTInetUrl** with urlType attribute "1" base URL appears (at the SLT level) and no svcInetUrl is present for the Service (at the service layer), the service_id term is appended to indicate the Service to which the requested signaling metadata objects apply. When constructing a URL path, the value of service_id shall be represented as a decimal number (with no leading zeros and no decimal point).

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 the URL is explicitly associated with the desired Service.

The response body for those HTTP requests shall include an MBMS metadata envelope per Section 7.1.6.1. The validFrom and validUntil attributes of the item elements should be present, to indicate the interval of validity of each signaling object. The absence of an expressed boundary of the validity period means that the validity period is unbounded in the corresponding direction in time. Responses should include the latest currently valid version of signaling objects.

The item element of the MBMS metadata envelope shall be extended by the addition of two optional attributes, @nextURL and @availAt.

Thus, at the @availAt time for a signaling object that was acquired via broadband, the next scheduled update of the signaling object shall be available to 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. It is expected that each receiver will initiate its request at a randomly-selected time between @availAt and shortly before @validUntil to request the @nextURL in order to smooth the transaction load on the Signaling Server. It is expected that receivers do not leave this until the very end of the validity period but allow for network and processing latencies so as to ensure that the scheduled update is received prior to @validUntil. The response to a request using a nextURL shall conform to the requirements of a response to a SvcInetUrl or SLTInetUrl request, however signaling objects in the response are not required to include currently valid signaling objects; i.e. the @validFrom element associated with all signaling objects in the response may indicate a time in the future.

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-1.0-20190122.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 6.18 describes this attribute.

Table 6.18 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.
@availAt	0..1	xs:dateTime	The time at which the URL of the next scheduled update to the signaling object described in the item element will be available.

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

@availAt – The earliest time at which the resource identified by @nextURL will be available for retrieval from the Signaling Server. When present, @availAt shall have a value that is earlier than the value of @validUntil (if @validUntil is present). When absent, if a @validUntil attribute is specified for the signaling object, the absence of @availAt shall mean a default value of @availAt that is 1 minute prior to @validUntil.

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 specification A/337 [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 specification A/332 [5].

7. SERVICE LAYER SIGNALING

For Service delivery using ROUTE, the SLS for each Service describes characteristics of the Service, such as a list of its Components and where to acquire them, 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 ROUTE the SLS shall include the tables based on Service Type and Configuration as shown in Table 7.1 through Table 7.2.

Table 7.1 SLS Table Requirements per Service Type for Broadcast Configuration

Table Service Type	S-TSID	USBD[7]	MPD	HELD App/no App	DWD App/no App	RSAT	APD
Linear A/V	M	O	M	M/na	O/na	O	O
Linear Audio	M	O	M	M/na	O/na	O	O
App Based	M	O	O	M/na	O/O	O	O
ESG	M	O	na	na/na	na/na	O	O
DRM Data	M	O	na	na/na	na/na	O	O
Data	M	O	O	O/O	O/O	O	O

M = Mandatory, O = Optional, na = Not Applicable (and no defined semantics)

Table 7.2 SLS Table Requirements per Service Type for Broadband Configuration

Table Service Type	S-TSID[7]	USBD[7]	MPD	HELD App/no App	DWD App/no App	RSAT	APD
Linear A/V	O	O	M	M/na	O/na	O	O
Linear A	O	O	M	M/na	O/na	O	O
App Based	O	O	O	M/na	O/O	O	O
ESG	O	O	na	na/na	na/na	O	O
DRM Data	O	O	na	na/na	na/na	O	O
Data	O	O	O	M/na	O/O	O	O

M = Mandatory, O = Optional, na = Not Applicable (and no defined semantics)

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 7.3 shows the elements and attributes of the ROUTE USBD that would be used in practice for ATSC 3.0 Service delivery.

For MMTP, the SLS for each MPU-formatted streaming 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 shall include the USBD fragment, the MMT Package (MP) table, and may include the HELD, and the DWD. For hybrid delivery, the MMTP-specific SLS shall include the MPD for broadband Components. Table 7.8 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].

[7] A/331:2020 required this SLS table.

The `Service.BroadcastSvcSignaling` element in the SLT shall be unique per Service (i.e., the collection of its attribute values shall not be repeated for any other Service) in a Broadcast Stream.

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. An 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 Service delivery using ROUTE, the receiver can acquire SLS fragments carried over the IP/UDP/LCT channel as indicated in the SLT, whereas for streaming Service delivery using MMTP, the receiver can acquire SLS fragments carried over the MMTP session as indicated in the SLT. The USBD/USD fragment describes basic properties of the Service such as its identifier, status, name, associated language(s) and delivery method. 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.

In broadcast delivery of SLS for a DASH-formatted streaming Service delivered using ROUTE, since SLS fragments are NRT files in nature, their carriage over the ROUTE session/LCT channel assigned by the SLT shall be in accordance to the Unsigned Packaged Mode or the Signed Package Mode as described in Section A.3.3.4 and A.3.3.5, respectively.

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 for a streaming Service, also 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 feature, or data Services such as the ESG or EAS content.

The HELD fragment 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.

The DWD fragment provides the broadcast delivery schedule of NRT files for consumption by either the receiver or broadcaster applications, and may contain additional metadata such as application context identifiers and filter codes that enable selective content reception, as well as identification of the application to which the application-related files belong.

The RSAT fragment (A/200, [55]) provides a schedule of upcoming changes to the available broadcast Services in the region. This schedule indicates when Services will move to another broadcast or will be discontinued.

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.

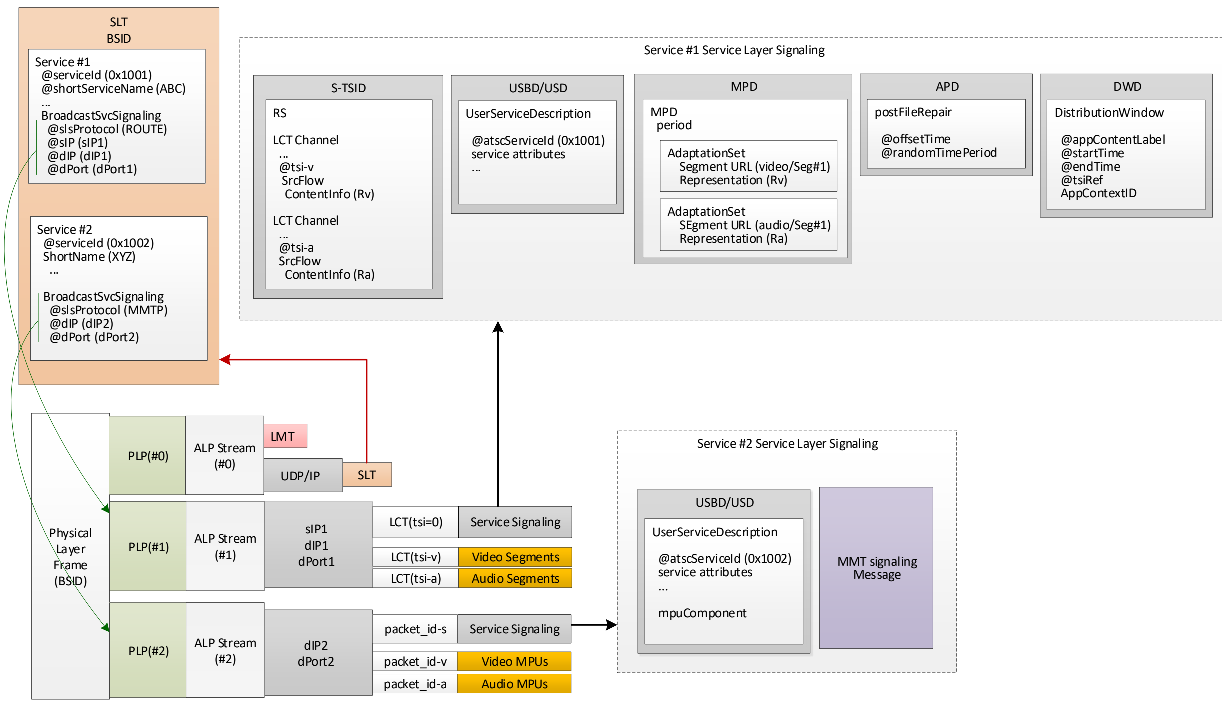

Figure 7.1 Example use of Service signaling for bootstrapping and Service discovery.

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 for the selected Service (in order to monitor for emergency alerts), Components of any given Service shall be transported in no more than four total ALP packet streams including the ALP stream transporting the LLS tables describing that Service.[8]

> Note: Each ROUTE session shall connect to a maximum of one PLP via an ALP stream. Multiple ROUTE sessions may connect to a given PLP/ALP stream. Each MMTP session shall connect to a maximum of one PLP via an ALP stream. Multiple MMTP sessions may connect to a given PLP/ALP stream.

Figure 7.2 depicts how to organize robust audio Service with one PLP/ALP stream per ROUTE session. There are two basic QoS(s) utilized in the Service. The first of these is more robust and carries signaling and audio in one ROUTE session. The second, less robust QoS, delivers video and text (for example, closed-captioning) in a second ROUTE session.

[8] With LDM, services delivered in the Enhanced Layer cause a receiver to dedicate a PLP processor to decoding the Core Layer related to that Enhanced Layer. That Core Layer processor counts as one of the four PLP processors available in the receiver whether that Core Layer carries a stream related to the desired service or not.

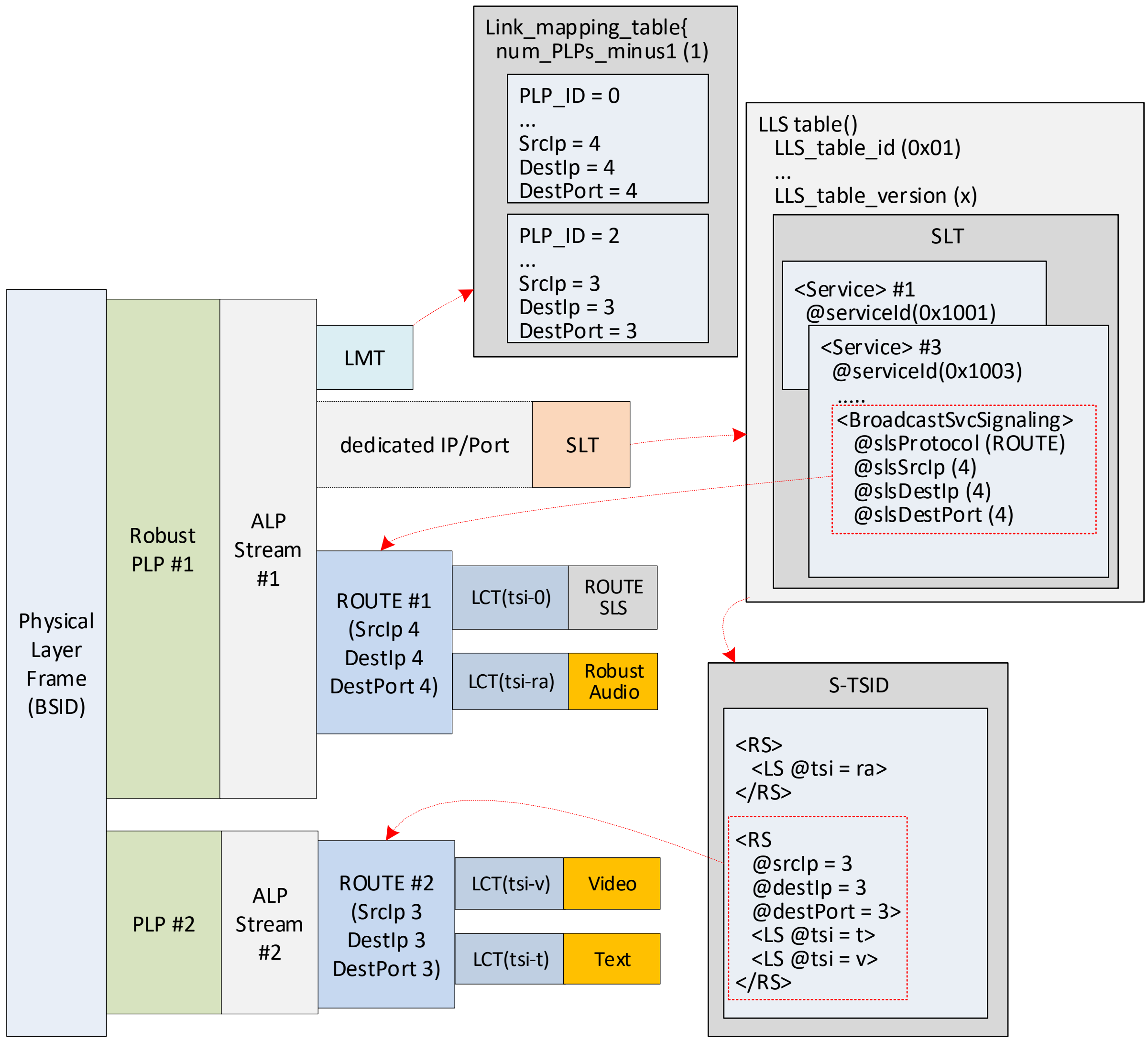

Figure 7.2 Example use of separate ROUTE sessions/PLPs to support different QoS for different data types.

7.1 ROUTE/DASH Service Layer Signaling

Service Layer Signaling provides detailed technical information to an ATSC 3.0 receiver to enable the discovery and access of ATSC 3.0 user Services and their Components. It comprises a set of XML-encoded metadata fragments. SLS 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 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.. When SLS is delivered via ROUTE, 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 7.3.

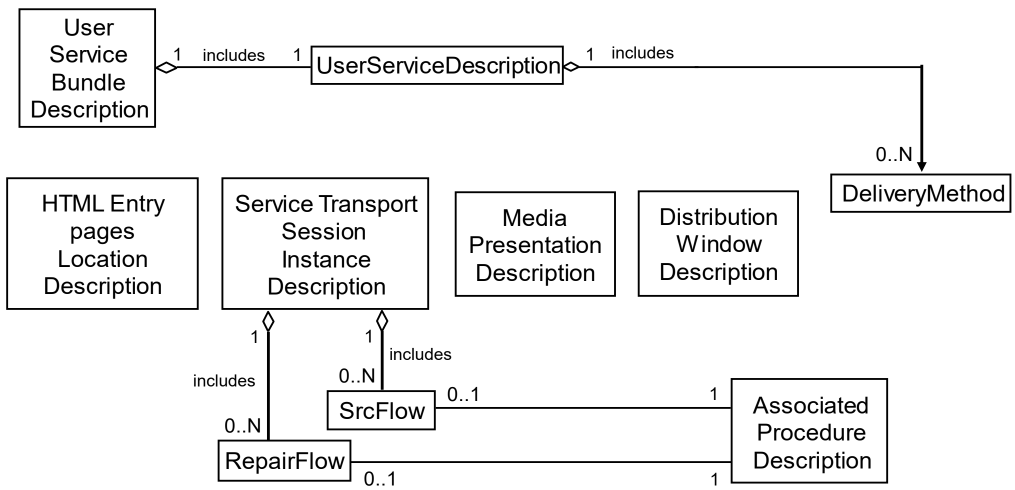

Figure 7.3 Service Layer Signaling data model for ROUTE/DASH Linear Services.

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), HTML Entry pages Location Description (HELD), and Distribution Window Description (DWD) 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 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, when 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 USBD, S-TSID, APD, HELD, DWD, and RSAT are further detailed in Sections 7.1.3, 7.1.4, 7.1.7, 7.1.8, 7.1.9, and 7.1.10 respectively.

7.1.1 Streaming Content Signaling

The streaming content signaling element 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 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 Application Signaling

7.1.2.1 App-based Feature Signaling

App-based feature signaling element 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 feature elements, 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 Section 7.1.8.

7.1.2.2 Signaling of Broadcaster Application Properties

A Service can contain zero or more app-based features. For example, a linear Service could contain one app-based feature consisting of a Broadcaster Application that manages the insertion of targeted ads, and another Broadcaster Application that contains a collection of Broadcaster Applications that provide an interactive viewing experience to enhance the audio/video program. Each Broadcaster Application is separately signaled, so that the creators of diverse Broadcaster Applications do not need to coordinate their signaling.

The URL of an Entry Package containing an Entry Page can be signaled for a Service, i.e., the `@bcastEntryPackageUrl` attribute or the `@bbandEntryPageUrl` attribute in HELD defined in Section 7.1.8. This entry URL can change from time to time. Changes to the entry URL can be very infrequent. For example, an Entry Page contains a container (e.g., an iFrame) into which it can load different sub-pages sequentially (usually driven by Events in the broadcast). In this case the entry URL would typically change only when a new version of the Entry Page is desired. Alternatively, changes to the entry URL can be frequent. For example, an Entry Page routinely changes at show and/or interstitial boundaries. The approach to changes of the entry URL is determined by the broadcaster.

7.1.2.3 Broadcaster Application Lifecycle

For the Broadcaster Application Lifecycle, see A/344, Section 6.3 [54].

7.1.2.4 HTTP User-Agent Header Field

In order for HTTP servers to customize the delivery of apps, HTTP requests from the receiver on behalf of a Broadcast Application, including the launch URL request, shall provide a User-Agent header field as defined in HTTP [32] and this section.

The syntax shall conform to HTTP [32] Section 5.5.3 and the syntax defined below:

```
"ATSC3/"<yyyy>[-mm] "("<capabilities>; <vendorName>; <modelName>;
   <softwareVersion>; <familyName>;[ hardwareVersion]")" <HTML>
```

where:

`yyyy` shall be the required year of publication of A/300

`mm` shall be the optional month of publication of A/300, which is required if it is the second or more publication in that year

`capabilities` shall be a required string conforming to the syntax of sa:Capabilities as defined in A/332 [5] Section 5.2.2.3.3

`vendorName` shall be a required string of the Receiver vendor's name

`modelName` shall be a required string of the Receiver vendor's model name (vendor-dependent)

`softwareVersion` shall be a required string of the Receiver vendor's software version (vendor-dependent)

`familyName` shall be a required string of the Receiver vendor's family name, e.g. `org,vendor.x` (vendor-dependent) `hardwareVersion` shall be an optional string of the Receiver vendor's model name (vendor-dependent)

`HTML` shall be the receiver's HTML User-Agent information

Although the full syntax of "`capabilities`" is permitted, it is expected that supported capabilities are ANDed together (e.g., "`509 50a &`"), and it is expected that the OR ("|") is not used.

The ABNF [23] below is authoritative over the pseudocode and description above.

```
; !syntax("abnf")
ATSCUser-Agent  = ATSCproduct RWS ATSCcomment RWS *User-Agent
ATSCproduct     = "ATSC3/" date
ATSCcomment     = "(" SAcapabilities vendorName modelName softwareVersion
familyName [hardwareVersion] ")"
date            = year ["-" month]
year            = 4DIGIT
month           = 2DIGIT
SAcapabilities  = capabilities
vendorName      = commentBody
modelName       = commentBody
softwareVersion = commentBody
familyName      = commentBody
hardwareVersion = commentBody
commentBody     = ";" RWS 1*(ATSCctextNSP / quoted-pair)
ATSCctext       = HTAB / SP / ATSCctextNSP
ATSCctextNSP    = %x21-27 / %x2A-3A / %x3C-5B / %x5D-7E / obs-text ;except
semicolon

; A/332
ccode           = 1*4HEXDIG
string-code     = ubyte "=" 1*UTF8-char
ubyte           = 1*3DIGIT
boperator       = "&" / "|"
expr            = ccode / string-code
                        / expr WSP expr WSP boperator
capabilities    = expr

; RFC 7231
User-Agent      = product *( RWS ( product / comment ) )
product         = token ["/" product-version]
product-version = token

; RFC 7230
token           = 1*tchar
tchar           = "!" / "#" / "$" / "%" / "&" / "'" / "*"
                  / "+" / "-" / "." / "^" / "_" / "`" / "|" / "~"
                  / DIGIT / ALPHA
comment         = "(" *( ctext / quoted-pair / comment ) ")"
```

```
ctext           = HTAB / SP / %x21-27 / %x2A-5B / %x5D-7E / obs-text
quoted-pair     = "\" ( HTAB / SP / VCHAR / obs-text )
obs-text        = %x80-FF
RWS             = 1*( SP / HTAB )

; RFC 5234
ALPHA           = %x41-5A / %x61-7A   ; A-Z / a-z
DIGIT           = %x30-39
HEXDIG          =  DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
HTAB            = %x09
SP              = " "
VCHAR           = %x21-7E
WSP             = " " / %x09

;RFC 3629
UTF8-char   = UTF8-1 / UTF8-2 / UTF8-3 / UTF8-4
UTF8-1      = %x00-7F
UTF8-2      = %xC2-DF UTF8-tail
UTF8-3      = %xE0 %xA0-BF UTF8-tail / %xE1-EC 2( UTF8-tail ) /
                 %xED %x80-9F UTF8-tail / %xEE-EF 2( UTF8-tail )
UTF8-4      = %xF0 %x90-BF 2( UTF8-tail ) / %xF1-F3 3( UTF8-tail ) /
                 %xF4 %x80-8F 2( UTF8-tail )
UTF8-tail   = %x80-BF
```

An example for a receiver conforming to A/300:2022 that signals SHEVC and AC-4 with "HTML" strings from a popular web browser is as follows:

```
ATSC3/2022-06 (50B 512 &; TVco; Model55; R4.2; com.tvco.55; 5.234)
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/75.0.3770.142 Safari/537.36
```

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 definition of this schema is in an XML schema 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 7.3 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 7.3 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 7.3 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** element 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 DRM Data Services.
BroadcastAppService	0..N		A DASH Representation delivered over broadcast containing the corresponding 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 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 Service named in the parent USBD fragment. 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 DRM Data Service.

@lang – This attribute shall indicate the language for the name of this ATSC 3.0 Service, and shall be represented by formal natural language identifiers as defined by BCP 47 [33].

ServiceLanguage – This element is for backwards compatibility and should not be present in XML encodings that conform to this Standard. Decoders should ignore it when present.

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 Components. This element is not applicable to and therefore shall be absent for the ESG and DRM Data 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 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 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 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 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 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 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 7.4 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-1.0-20230714.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 7.4 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 7.4 Semantics of the Service-based Transport Session Instance Description Fragment

Element and Attribute Names	Use	Data Type	Description
S-TSID			Service-based 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 7.4.

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 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 when this ROUTE session, whose LCT channel carries 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 when this ROUTE session, whose LCT channel carries 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 when this ROUTE session, whose LCT channel carries 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 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 [21]. 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 [21], and representing the "t=<start-time>" value, The only difference from RFC 4566 [21] 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 [44]. 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 [21], and representing the "t=<end-time>" value, The only difference from RFC 4566 [21] 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 [44]. 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 shall contain 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 only applicable to a Service containing DASH-formatted content. It 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 identical to the data structure and semantics of the DASH Media Presentation Description as defined in DASH-IF [12].

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.

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 fragments 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 uniformly-distributed 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 7.5 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 7.5 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 7.5 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. When 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 uniformly-distributed 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 of the following events:
  - Presence of the Close Object flag (B) in the LCT header [27] for the file of interest;
  - Presence of the Close Session flag (A) in the LCT header [27] before the nominal expiration of the Extended FDT Instance as defined by the @Expires attribute.

7.1.8 HTML Entry pages Location Description (HELD)

7.1.8.1 Semantics of HELD

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

tag:atsc.org,2016:XMLSchemas/ATSC3/AppSignaling/HELD/1.0/

The XML schema "xmlns" short name should be "held".

Table 7.6 provides an informative description of the semantics of the HELD. The normative XML schema for HELD shall be as specified in the file *HELD-1.0-20210312.xsd*. The normative semantics of the elements and attributes of the HELD follow Table 7.6.

Table 7.6 HTML Entry pages Location Description (HELD) Semantics

Element Name	Use	Data Type	Description
HELD	1		Includes HTML entry page collection elements.
HTMLEntryPackage	1..N		Contains properties of the Entry Package.
@appContextId	1	anyURI	Defines the Broadcaster Application Context Identifier for this Entry Package.
@requiredCapabilities	0..1	sa:CapabilitiesType	Device capabilities needed for meaningful rendition of the Entry Page (as defined in A/332, "Service Announcement").
@appRendering	0..1	boolean	For a linear Service, indicates a broadcaster request that the broadcaster application be allowed to render the presentable Component(s) of the Service.
@clearAppContextCacheDate	0..1	dateTime	Specifies that all files associated with the Broadcaster Application context before the indicated date and time can be deleted.
@bcastEntryPackageUrl	0..1	anyURI	URL of an Entry Package containing the Entry Page of a Broadcaster Application delivered by broadcast.
@bcastEntryPageUrl	0..1	anyURI	URL of an Entry Page of a broadcast-delivered Broadcaster Application.
@bbandEntryPageUrl	0..1	anyURI	URL of an Entry Page of a broadband-delivered Broadcaster Application.
@validFrom	0..1	dateTime	Indicates that the Entry Page contained in the Entry Package is to be loaded at the date and time of @validFrom, or at any time after the date and time of @validFrom and before the date and time of @validUntil when the Service is selected.
@validUntil	0..1	dateTime	Indicates that the Entry Page contained in the Entry Package is to be unloaded at the date and time of @validUntil.
@coupledServices	0..1	held:listOfUnsignedShort	Provides a space-separated list of linear Services sharing a common broadcaster application.
@lctTSIRef	0..1	held:listOfUnsignedInt	A list of one or more TSI values for an LCT channel which carries a Broadcaster Application Entry Package.
@default	0..1	boolean	This attribute, when present and set to "true" indicates the default Broadcaster Application to run on service acquisition and other events.
@appId	0..1	anyUri	Defines an identifier for the Broadcaster Application when @appContextId is not sufficient.

HELD – This root element contains one or more **HTMLEntryPage** elements.

HTMLEntryPackage – Each instance of this element contains information about one Entry Package that contains an HTML Entry Page.

@appContextId – This required xs:anyURI attribute represents a Broadcaster Application Context Identifier as a URI that indicates possible shared use of Broadcaster Application resources among multiple Broadcaster Applications. Resources associated with a Broadcaster Application and hence an Application Context Identifier shall be made available to another Broadcaster Application if and only if each has the same Application Context Identifier. Refer to A/344, "Interactive Content" [54] for the definition of a Broadcaster Application and how absolute URLs are formed from the relative URLs associated with broadcast-delivered resources based on the value of the @AppContextId.

Resources are delivered over the LCT channel(s) identified by the @lctTSIRef attribute and are made available to only the Broadcaster Applications associated with the particular Application Context Identifier.

For a given **HTMLEntryPackage** element present in the HELD, any files delivered over the LCT channels identified by the @lctTSIRef attribute for that **HTMLEntryPackage** element instance shall be associated with the Application Context Identifier defined for the same **HTMLEntryPackage** instance.

@requiredCapabilities – When present, this optional sa:CapabilitiesType attribute shall describe additional device capabilities needed for meaningful rendition of the Entry Page contained within this Entry Package (beyond those defined in A/344, "Interactive Content" [54]). The syntax and semantics of the @requiredCapabilities attribute shall be the same as the sa:Capabilities element specified under the Content fragment in A/332, "Service Announcement" [5]. When the @requiredCapabilities attribute is not present, the capabilities required to provide a meaningful rendition of the Entry Page are as defined in A/344, "Interactive Content" [54].

@appRendering – This optional xs:boolean attribute shall indicate, when present and set to "true", that for the associated linear Service, a broadcaster's request that the broadcaster application, via the Application Media Player as defined in A/344, "Interactive Content" [54], be allowed to render the Component(s) of the Service. The @appRendering attribute shall only be present for Services for which the value of the **SLT.Service**@serviceCatgory is equal to "1" (indicating a linear A/V Service with a Broadcaster Application enhancement). At the manufacturer's discretion, Receivers may choose to disregard this flag and upon initial acquisition of the Service, always begin rendering such Services using the Receiver Media Player as defined in A/344, "Interactive Content" [54]. When the @appRendering attribute is "false" or not present, the broadcaster has not requested that the broadcaster application be allowed to render the Component(s) of the Service. When this attribute is not present, the default value is "false".

@clearAppContextCacheDate – This optional xs:dateTime attribute shall indicate to receivers that the broadcaster wishes all files associated with the Broadcaster Application context with a date and time before the specified date and time to be deleted from the cache used for the Application Context identified by @appContextId. The date and time of each file are reflected in their Last-Modified HTTP header parameters. The Last-Modified value of a file is defined using signaling as described in A/344, "Interactive Content" [54]. Setting this attribute overrides any hints that may have been provided to extend the availability of a file, such as the Expires parameter. When this attribute not present, there is no default value.

@bcastEntryPackageUrl – This optional xs:anyURI attribute shall reference an Entry Package containing the Entry Page of a Broadcaster Application delivered by broadcast. The package referenced by the @bcastEntryPackageUrl attribute shall be present in the broadcast emission. For broadcast-delivered content, the @bcastEntryPackageUrl attribute shall point to an object delivered via ROUTE, and shall be a relative URL. Specifically, the object delivered in ROUTE is a signed package and has an EFDT-Instance.File@Content-Location that matches the value provided in the @bcastEntryPackageUrl. Note that while the syntax requires a URL, it is sufficient that the strings of these two elements match. Either the @bcastEntryPackageUrl attribute or the @bbandEntryPageUrl attribute or both shall be present. When this attribute is not present, there is no default value.

@bcastEntryPageUrl – This optional xs:anyURI attribute shall indicate the URL of the broadcast-delivered Entry Page; e.g., the file in the package with a Content-Location value matching the value given in the @bcastEntryPageUrl attribute. When the @bcastEntryPackageUrl attribute is present, the @bcastEntryPageUrl attribute shall also be present.

@bbandEntryPageUrl – This optional xs:anyURI attribute shall be a URL of an Entry Page of a broadband-delivered Broadcaster Application. The @bbandEntryPageUrl attribute shall reference a file that shall be available for download from the referenced broadband server and shall be an absolute URL. Either the @bcastEntryPackageUrl attribute or the @bbandEntryPageUrl attribute or both shall be present. When both @bcastEntryPackageUrl and @bbandEntryPageUrl are present, the Receiver is expected to attempt to acquire the broadband reference before attempting to acquire the broadcast resource. When this attribute is not present, there is no default value.

A change in either of these two URLs shall be an indication that a new Entry Page is available.

@validFrom – This optional xs:dateTime attribute shall indicate that, when the Service is selected, the Entry Page (the page referenced by the @bcastEntryPageUrl attribute or the @bbandEntryPageUrl attribute) is intended to be loaded starting at the indicated date and time, or at any time after the indicated date and time and before the date and time indicated by @validUntil. When this attribute is not present, the default value is upon receipt of the HELD, the Entry Page is valid now and the Entry Page is intended to be loaded at that time. The value of the @validFrom attribute may indicate a time in the future.

@validUntil – This optional xs:dateTime attribute shall indicate that the Entry Page (the page referenced by the @bcastEntryPageUrl attribute or the @bbandEntryPageUrl attribute) is intended to be unloaded starting at the indicated date and time. When the @validUntil attribute is not specified, the Entry Page indicated by the @bcastEntryPageUrl attribute or the @bbandEntryPageUrl attribute is intended to stay loaded for the indefinite future. The value of the @validUntil attribute, when present, shall indicate a time in the future relative to the delivery time of this instance of the HELD, and shall represent a time farther in the future than the time given in the @validFrom attribute, when the @validFrom attribute is present. When this attribute is not present, the default value is that the Entry Page is valid indefinitely.

@coupledServices – This optional held:listOfUnsignedShort attribute shall consist of a space-separated list of linear Services sharing a common broadcaster application. Each Service shall be represented as a 16-bit unsigned integer value of the **SLT.Service**@serviceId The Receiver may use the information in the @coupledServices attribute as a caching hint, to preferentially retain the Application Content in cache upon a Service change, since a tuned-to Service may employ the same Broadcaster Application. When this attribute is not present, there is no default value.

@lctTSIRef – This optional held:listOfUnsignedInt attribute shall consist of a space-separated list of TSI values identifying an LCT channel which carries Broadcaster Application-related files, when they are delivered over broadcast. When this attribute is not present, there is no default value.

@default – This optional attribute shall signal the Broadcaster Application to start upon Service acquisition and other events that trigger the Receiver to examine and start a Broadcaster Application from the HELD. When multiple **HTMLEntryPackage** elements are present the @default attribute shall be present on exactly one of those elements. When this attribute is not present, the default value is "false".

@appId – This optional attribute shall uniquely identify the Broadcaster Application when the HELD contains two or more **HTMLEntryPackage** elements. The syntax of this attribute shall be the same as the appContextId. When present at all, this attribute shall be present on all **HTMLEntryPackage** elements. When this attribute is not present, there is no default value.

7.1.8.2 HELD Broadcaster Application Distribution Timing

All application files referenced in the HELD shall be available for retrieval from the Broadcast Stream or accessible from a broadband server (as appropriate to the delivery path). For example, if the HELD is updated to indicate an Entry Package that is valid at a time six hours from now, the files associated with that Broadcaster Application (as indicated by the @bcastEntryPackageUrl attribute) shall be available to the Receiver at the time of the HELD update, even though the launch time is six hours in the future.

7.1.8.3 HELD Examples

This section provides illustrative examples of HELD instances.

The first example is the simplest. This HELD only signals one Entry Page delivered via broadcast, and gives no indication of any future expiration time.

```
<HELD>
    <HTMLEntryPackage appId="http://kids.pbs.org" appContextId="http://kids.pbs.org"
        bcastEntryPackageUrl="app" bcastEntryPageUrl="index.html"/>
</HELD>
```

The second example illustrates several principles:

- The ability to distribute and signal Broadcaster Applications targeted at Receivers with different capabilities (a2).
- The ability to pre-announce the timing of a URL switch, and the Broadcaster Application resources that will be needed at the time of the switch (a1-a2 – see more below examples).
- The ability to announce the Entry Page via broadband-downloadable only (a5), or both broadcast-delivered and broadband-downloadable (a6).
- Identifying the default Broadcaster Application to launch (a1).
- Signaling different appContextIDs (a1-a3 versus a4-a6).
- Unique appId values (a1-a6).

```xml
<HELD>
    <HTMLEntryPackage appId="http://kids.pbs.org/a1" appContextId="http://kids.pbs.org"
       bcastEntryPackageUrl="app" bcastEntryPageUrl="p1/index.html" validUntil="2016-07-
       17T09:30:47Z" default="true"/>
    <HTMLEntryPackage appId="http://kids.pbs.org/a2" appContextId="http://kid.pbs.org"
       bcastEntryPackageUrl="app" requiredCapabilities="050E 058E |"
       bcastEntryPageUrl="p1a/index.html" validUntil="2016-07-17T09:30:47Z"/>
    <HTMLEntryPackage appId="http://kids.pbs.org/a3" appContextId="http://kids.pbs.org"
       bcastEntryPackageUrl="app" bcastEntryPageUrl="p2/index.html" validFrom="2016-07-
       17T09:30:47Z" validUntil="2016-07-17T12:00:47Z"/>
    <HTMLEntryPackage appId="http://kids.pbs.org/a4" appContextId="http://kids.pbs.org/alt"
       bcastEntryPackageUrl="app" bcastEntryPageUrl="p2a/index.html" validFrom="2016-07-
       17T09:30:47Z" validUntil="2016-07-17T12:00:47Z"/>
    <HTMLEntryPackage appId="http://kids.pbs.org/a5" appContextId="http://kids.pbs.org/alt"
       bbandEntryPageUrl="http://kids.pbs.org/a5/index.html" validFrom="2016-07-17T09:30:47Z"
       validUntil="2016-07-17T12:00:47Z"/>
    <HTMLEntryPackage appId="http://kids.pbs.org/a6" appContextId="http://kids.pbs.org/alt"
       bcastEntryPackageUrl="app" bcastEntryPageUrl="p2a/index.html"
       bbandEntryPageUrl="http://kids.pbs.org/a6/index.html" validFrom="2016-07-17T12:30:47Z"
       validUntil="2016-07-17T13:00:47Z"/>
</HELD>
```


Note that the first **HTMLEntryPackage** element contains a URL and a @validUntil attribute indicating a future time after which the Broadcaster Application at p1/index.html is no longer valid. The second instance of the **HTMLEntryPackage** element contains a URL referencing a different Broadcaster Application at p1a/index.html and a @requiredCapabilities attribute indicating that the Receiver must support a runtime environment capability not specified in the version of the ATSC Standards the Receiver was built to support (code 050E or 058E). Receivers that recognize and support the capabilities identified with the 050E or 058E code are expected to execute the Broadcaster Application at p1a/index.html as opposed to p1/index.html.

7.1.9 Distribution Window Description (DWD)

The Distribution Window Description provides the broadcast delivery schedule of NRT files targeted for use either directly by the receiver, or by broadcaster applications. A broadcaster application may pertain to app-based features for a linear Service, or constitute a standalone, app-based Service. The DWD may contain 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.

The DWD fragment comprises one or more instances of the **DistributionWindow** element as an indication that one or more NRT content items or files are scheduled to be delivered by ROUTE. A receiver can then tune to/join the appropriate Broadcast Stream and LCT channel over which the NRT files are broadcast during the distribution window time frame to download and store that content. These files could be directed to the receiver, or a broadcaster application, and comprise any combination of an HTML5 entry page and/or other files such as JavaScript, CSS, XML and media files). The files may also serve other purposes, such as conveying DRM message data.

It might be desirable for a collection of NRT files to be broadcast during multiple distribution windows, to increase the likelihood of successful reception by a receiver having interest in those files, since the receiver may be unable to tune to the appropriate broadcast stream/LCT channel during any given distribution window instance. For example, a single-tuner receiver may be in active use and tuned to a different Service during a given distribution window, but the receiver may not be in active use during a later instance of a distribution window that delivers the same content.

The specific set of NRT files to be delivered during the time described by an instance of the **DistributionWindow** element (in the interval from the **DistributionWindow**@startTime attribute to the **DistributionWindow**@endTime attribute) is uniquely associated with the value of **DistributionWindow**@contentLabel attribute. In the delivery of NRT files (either application-related or not), the files sent during the one or more instances of the **DistributionWindow** element with the same label (i.e., with the same value of @contentLabel but different time windows) shall be identical, i.e. contain the same objects. Furthermore, a DWD fragment may contain distribution window instances with multiple @contentLabel attribute values. In this case, for all time intervals (t_1, t_2), (t_3, t_4),... t_i shall be less than t_j for j>i. This enables a broadcaster to offer the content in multiple different distribution windows, and also allows a receiver to avoid participating in multiple distribution windows delivering the same content.

Each distribution window instance may include one or more instances of the **AppContextID** element. When present, each **AppContextID** instance may include the @dwFilterCode attribute which, when present, shall contain one or more Filter Codes. Filter Codes are associated with application-related files and are represented by integers that are unique within a given instance of **AppContextID**. They represent personalization categories as defined by individual broadcaster entities. For example, different Filter Code values may be assigned to categories such as truck owner, sustaining member, or a zip code. Filter Codes can be associated with application-related files. In ROUTE delivery, identification of application-related files associated with Filter Codes is provided by the Extended FDT's @fileFilterCode attribute (see Annex A.3.3.2.3). In addition, the receiver can have internally-stored Filter Code values provided by broadcaster applications (using the Set Filters API as described in A/344, "Interactive Content" [54]).

Filter Codes associated with files can be compared with internally-stored Filter Codes to help determine whether a given file is relevant to a receiver. The @dwFilterCode attribute is a concatenated list of all Filter Codes for application-related files that shall be available during the containing distribution window. The Filter Codes in the @dwFilterCode attribute can be compared with the Filter Codes stored in the receiver to help the receiver determine whether to tune to (i.e. activate content reception on the corresponding BS/LCT channel during) a given distribution window instance. When one or more Filter Codes in the @dwFilterCode attribute match one or more internally-stored Filter Codes, then the receiver can determine that at least one file is relevant with respect to personalization. When no Filter Codes are associated with a distribution window instance (i.e., absence of @dwFilterCode attribute and/or there are no internally-stored Filter Codes available in the receiver) the receiver can nonetheless participate in the distribution window and download application-related files. However, the use of Filter Codes can optimize receiver memory space by avoiding storing irrelevant material, thus allowing more space for relevant material.

NRT files that might be consumed by a BA shall be associated with an AppContextId. Other NRT files may be associated with an AppContextId.

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

```
tag:atsc.org,2016:XMLSchemas/ATSC3/AppSignaling/DWD/1.0/
```

The XML schema xmlns short name shall be "dwd".

Table 7.7 provides an informative description of the semantics of the Distribution Window Description. The normative XML schema for DWD shall be as specified in the file *DWD-1.0-20180830.xsd*. The normative semantics of the elements and attributes of the DWD follow Table 7.7.

Table 7.7 Distribution Window Description (DWD) Semantics

Element Name	Use	Data Type	Description
DWD	1		
DistributionWindow	1..N		Broadcast transmission time frame(s) of NRT files.
@startTime	1	dateTime	Start time of the parent instance of **DistributionWindow** element.
@endTime	1	dateTime	End time of the parent instance of **DistributionWindow** element.
@lctTSIRef	1	dwd:listOfUnsignedInt	A space-separated list of TSI value of the LCT channel in which the NRT files are delivered, during the parent instance of **DistributionWindow** element.
@contentLabel	0..1	unsignedInt	A label or alias for the NRT file(s) delivered during the parent instance of **DistributionWindow** element. Distribution Window instances identified by the same value of the @contentLabel attribute shall transmit the same file(s).
AppContextId	0..N	anyURI	Defines the Application Context Identifier for the NRT file(s) delivered during the parent instance of the Distribution Window.
@dwFilterCode	0..1	dwd:listOfUnsignedInt	A space-separated list of unsigned integers associated with application content item(s) to be broadcast during the affiliated instance of **DistributionWindow** element.

DWD – This root element contains one or more **DistributionWindow** elements.

DistributionWindow – Each instance of this element shall define a single time interval during which NRT content files will be sent, in the LCT channel as identified by @lctTSIRef attribute.

@startTime – This required xs:dateTime attribute shall specify the start time of the associated distribution window.

@endTime – This required xs:dateTime attribute shall specify the end time of the associated distribution window. The @endTime attribute should indicate a date/time in the future relative to the time this instance of the DWD is inserted into the signal. (Time-shifted content, e.g., played back from a DVR, may have an @endTime attribute that is in the past.)

@lctTSIRef – This required dwd:listOfUnsignedInt attribute shall specify the space-separated list of TSI values of the LCT channel in which the NRT files shall be delivered, during the parent instance of **DistributionWindow** element.

@contentLabel – This optional xs:unsignedInt attribute, when present, indicates that each instance of **DistributionWindow** element identified by the same value of the @contentLabel attribute shall transmit the same NRT files between its start and end times. For files that are not associated with a broadcaster application, the value of this attribute shall differ for each unique set of files delivered during the interval spanning the entire set of Distribution Windows as defined by the DWD fragment. For files that are associated with a broadcaster application, one or more **AppContextId** shall be present, and the concatenation of the values of the @contentLabel attribute and **AppContextId** element shall be unique across the **DistributionWindow** elements in the DWD fragment. In this example of employing RF channel bonding for Service delivery, the @contentLabel shall be unique for each BSID value as conveyed in **SLT**@bsid. In the event that multiple broadcasters share a broadcast stream, it is expected that there will be coordination among them to ensure the uniqueness of the values of this attribute for use in the DWD fragment.

AppContextId – This conditionally optional xs:anyURI element, when present, corresponds to an Application Context Identifier as a URI that indicates possible shared use of the content to be delivered during the distribution window among multiple Broadcaster Applications. Resources associated with a Broadcaster Application and hence an Application Context Identifier shall be made available to another Broadcaster Application if and only if each has the same Application Context Identifier. Refer to A/344, "Interactive Content" [54] for the definition of a Broadcaster Application. This element shall be present for NRT files that may be consumed by a BA.

@dwFilterCode – This optional dwd:listOfUnsignedInt attribute is a space-separated list of broadcaster application specific 32-bit unsigned integers associated with the content item(s) to be broadcast during the affiliated instance of **DistributionWindow** element that are scoped by the parent **AppContextId** element. The @dwFilterCode attribute may be used by the receiver platform to determine whether content available during the given Distribution Window will be of interest to the device.

7.1.9.1 DWD Examples

This section provides illustrative examples of DWD instances.

The first example is the simplest. This DWD only signals one distribution window with its start time, its end time, one or more LCT channel(s), and application context identifier(s).

```
<DWD>
    <DistributionWindow startTime="2016-07-17T12:00:47Z" endTime="2016-07-18T12:00:47Z"
        lctTSIRef="43 44">
        <AppContextId>http://kids.pbs.org</AppContextId>
    </DistributionWindow>
</DWD>
```

The second example illustrates several principles:

- The ability to signal multiple distribution windows with different time slot information;
- The ability to signal the labels for the content objects scheduled to be delivered in the different time slots.

```xml
<DWD>
    <DistributionWindow contentLabel="1" startTime="2016-07-17T12:00:47Z" endTime="2016-07-
        18T12:00:47Z" lctTSIRef="43 44">
        <AppContextId>http://kids.pbs.org</AppContextId>
    </DistributionWindow>
    <DistributionWindow contentLabel="2" startTime="2016-07-18T12:00:47Z" endTime="2016-07-
        19T12:00:47Z" lctTSIRef="43 44">
        <AppContextId>http://kids.pbs.org</AppContextId>
    </DistributionWindow>
    <DistributionWindow contentLabel="1" startTime="2016-07-19T12:00:47Z" endTime="2016-07-
        20T12:00:47Z" lctTSIRef="43 44">
        <AppContextId>http://kids.pbs.org</AppContextId>
    </DistributionWindow>
        <DistributionWindow contentLabel="2" startTime="2016-07-20T12:00:47Z" endTime="2016-
        07-21T12:00:47Z" lctTSIRef="43 44">
        <AppContextId>http://kids.pbs.org</AppContextId>
    </DistributionWindow>
</DWD>
```

The third example illustrates several principles:

- The ability to signal multiple distribution windows with different time slot information;
- The ability to signal the labels for the content objects scheduled to be delivered in the different time slots.
- The ability to signal the Filter Codes of app-related files distributed during available time slots for each distribution window instance.

```xml
<DWD>
    <DistributionWindow contentLabel="1" startTime="2016-07-17T12:00:47Z" endTime="2016-07-
        18T12:00:47Z" lctTSIRef="43 44">
        <AppContextId dwFilterCode="1 2 3" >http://kids.pbs.org</AppContextId>
    </DistributionWindow>
    <DistributionWindow contentLabel="1" startTime="2016-07-18T12:00:47Z" endTime="2016-07-
        19T12:00:47Z" lctTSIRef="43 44">
        <AppContextId dwFilterCode="1 2 3" >http://kids.pbs.org</AppContextId>
    </DistributionWindow>
    <DistributionWindow contentLabel="2" startTime="2016-07-18T12:00:47Z" endTime="2016-07-
        19T12:00:47Z" lctTSIRef="45 46">
        <AppContextId dwFilterCode="4 5 6" >http://kids.pbs.org</AppContextId>
    </DistributionWindow>
</DWD>
```

7.1.10 Regional Service Availability Table (RSAT)

The Regional Service Availability Table provides a schedule of upcoming changes to the available broadcast Services. The schedule information may be restricted to only the related Service or may contain information pertaining to many Services. More details of the usage of the RSAT are specified in A/200, Regional Service Availability [55].

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, the HELD fragment and the RSAT fragment (see A/200 [55]). The USBD fragment contains Service identification, references to other SLS information required to access the Service and constituent 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 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 targeted to either receivers or broadcaster applications. Details about the DWD fragment can be found in Section 7.1.9, and details of the HELD fragment are specified in Section 7.1.8. 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 7.4.

The streaming content signaling element of the SLS for MPU Components shall correspond to the MP table defined in subclause 10.3.9 of ISO/IEC 23008-1 [38]. 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.

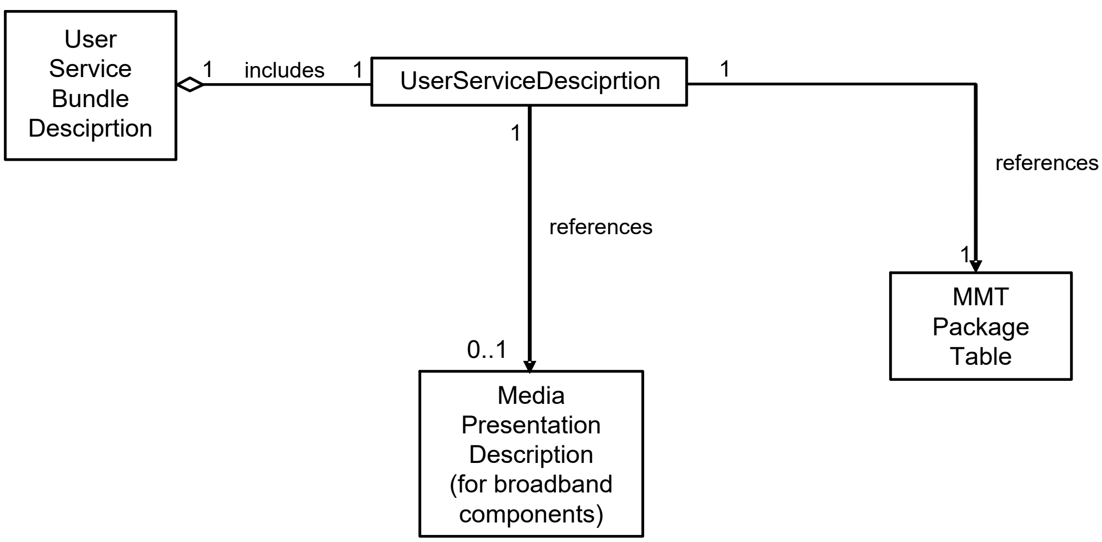

Figure 7.4 Service Layer Signaling data model for MMTP/MPU Linear Services.

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, @contentIdSchemdIdUri, @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**, its child attribute @fullMPDUri, and its child element **BroadbandComponentInfo** 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 7.8 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-1.0-20210401.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 7.8 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 7.8 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 Components of ATSC 3.0 Service delivered as MPUs.
@mmtPackageId	1	string	Reference to a MMT Package for 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 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 of the Transport sessions (i.e., LCT channel(s) of ROUTE session(s)) 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 Components of ATSC 3.0 Service delivered by broadband.
@fullMPDUri	1	anyURI	Reference to an MPD fragment which contains descriptions for Components of the ATSC 3.0 Service delivered over broadband.
BroadbandComponentInfo	0..N		A description of the relationship among Components of ATSC 3.0 Service delivered by broadcast and broadband, respectively.
@repId	1	stringNoWhitespace	Representation ID of the DASH Representation delivered by broadband.
@complementaryAssetId	0..1	listOfStringType	All complementary MMT Assets that the DASH Representation depends upon in the decoding and/or presentation process as a whitespace-separated list of values of complementary asset ID.
@dependentAssetId	0..1	listOfStringType	All dependent MMT Assets that depend upon the DASH Representation in the decoding and/or presentation process as a whitespace-separated list of values of dependent asset ID.
@simulcastAssetId	0..1	string	Asset ID of an MPU Component carrying the same content with the DASH Representation.
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 7.3.)

`@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 7.3.)

`Name` – Name of the ATSC 3.0 Service in the language specified by `@lang` attribute. (Same as given in Table 7.3.) 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 [33]. (Same as given in Table 7.3.)

`ServiceLanguage` – This element is for backwards compatibility and should not be present in XML encodings that conform to this Standard. Decoders should ignore it when present.

`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]. The content rating information is usually assigned to content during production and distribution. Content might have a rating, or it might not have a rating (and if it has a rating, that rating might be "None", "NR", or "E", see CTA-766 [11]). This element is required except when the content in the service does not have a rating assigned, then it shall not be present. When the program in the service is rated "exempt", the element contents shall be empty. This element is authoritative over the Service Announcement (SA, ESG) rating. For North America, the terms "not rated" and "exempt" shall be as defined in CTA 766 [11].

`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 Components of ATSC 3.0 Service delivered as MPUs.

`@mmtPackageId` – Reference to a MMT Package for 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 Content ID; and
- the "Designator" for either the "full" or "compact" encoding as defined in SMPTE 2092-1 [42].

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 Content ID system identified by `@contentIdSchemeIdUri` attribute. The "EIDR Content ID" shall be a valid canonical EIDR entry as defined in [43]. The "Ad-ID Content ID" shall be a valid Ad-ID entry as defined in [42].

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 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 Content ID; and
- the "Designator" for either the "full" or "compact" encoding as defined in SMPTE 2092-1 [42].

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 Content ID system identified by `@nextContentIdSchemeIdUri` attribute. The "EIDR Content ID" shall be a valid canonical EIDR entry as defined in [43]. The "Ad-ID Content ID" shall be a valid Ad-ID entry as defined in [42].

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 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 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 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 Components of the ATSC 3.0 Service delivered over broadband.

`BroadbandComponentInfo` – A description of the relationship between Components of the ATSC 3.0 Service delivered by broadcast and broadband, respectively. This element shall be present when there is a dependency between the DASH representation delivered over broadband and the MMT Asset over broadcast or when the DASH representation delivered over broadband carries the same content as the MMT Asset over broadcast.

`@repId` – This attribute indicates the representation ID of DASH Representation delivered by broadband.

`@complementaryAssetId` – This attribute indicates all complementary MMT Assets that the DASH Representation depends on in the decoding and/or presentation process as a whitespace-separated list of values of complementary asset ID. This attribute is used when the DASH representation delivered over broadband depends on the MMT Asset(s) over broadcast. When `@complementaryAssetId` is not present, there is no default value. Dependent assets depend on complementary assets. For example, a complementary asset might be a base layer while a dependent asset might be the related enhancement layer.

@dependentAssetId – This attribute indicates all dependent MMT Assets that depend on the DASH Representation in the decoding and/or presentation process as a whitespace-separated list of values of dependent asset ID. This attribute is used when the MMT Asset(s) delivered over broadcast depends on the DASH representation over broadband. When @dependentAssetId is not present, there is no default value. For the definition of a dependent asset, see subclause 3.1 of ISO/IEC 23008-1 [38]. For example, a complementary asset might be a base layer while a dependent asset might be the related enhancement layer.

@simulcastAssetId – This attribute indicates the asset ID of an MPU Component carrying the same content as the DASH Representation. When @simulcastAssetId is not present, there is no default value.

ComponentInfo – Contains information about Components, available in the Service and delivered as MPUs. For each Component, this includes information about the Component type, Component role, Component name, Component identifier, and the 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 ATSC 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= ATSC 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 = ATSC 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 = ATSC 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 "true" this Component is protected (e.g., encrypted). When this flag is set to a value of "false" this Component is not protected (e.g., encrypted). When not present the value of componentProtectedFlag attribute is inferred to be equal to "false".

@componentId – This attribute indicates the identifier of this Component. The (string) value of this attribute shall be either a URI, or an RFC 4122 [20] canonical string representation of a UUID (e.g., "11111111-1111-1111-1111-111111111111"). If this value is a UUID, the "urn:uuid:" shall be omitted. The value of this attribute shall be the same as the asset_id in the MP table corresponding to this Component. As a result, the asset_id in the MP table corresponding to this Component is required to be a string value (whether URI or UUID), and if the asset_id value is a UUID, it is required to be formatted in the RFC 4122 [20] canonical string representation.

@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 a 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 identical to the data structure and semantics of the DASH Media Presentation Description as defined in DASH-IF [12].

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 [38] are delivered in binary format by MMTP packets according to Signaling Message Mode specified in subclause 9.3.4 of ISO/IEC 23008-1 [38]. 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 to 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 7.8. 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) [38].
- 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) [38].

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 [38]);
- 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) [38].

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 [38]. The syntax of this message shall be as provided in Table 7.9.

The semantics of each field in mmt_atsc3_message() shall be as described in the text following Table 7.9.

Table 7.9 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 wrap around 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 7.10.

Table 7.10 Code Values for atsc3_message_content_type

atsc3_message_content_type	Meaning
0x0000	ATSC Reserved
0x0001	`UserServiceDescription` as given in Table 7.8.
0x0002	MPD as given in DASH-IF [12].
0x0003	HELD
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
0x000B	RSAT as given in A/200, Regional Service Availability [55].
0x000C	Security Properties Descriptor (Sec. 7.2.4.2.1.1)
0x000D~0xFFFF	Industry Reserved. See ATSC Code Point Registry [71].

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 atsc3_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 coded per Table 7.11 that shall identify the type of compression applied to the data in atsc3_message_content_byte.

Table 7.11 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	(deprecated)
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. When 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 7.12. The semantics of the fields in the video_stream_properties_descriptor() shall be as given immediately below the table.

Table 7.12 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 7.14

}		
if (multiview_info_present) {		
multiview_info()	40	Table 7.16
}		
if (res_cf_bd_info_present) {		
res_cf_bd_prop_info()	48	Table 7.17
}		
if (pr_info_present) {		
if (sub_layer_profile_tier_level_info_present) {		
pr_info(max_sub_layers_instream-1)	var	Table 7.18
} else {		
pr_info(0)	var	Table 7.18
}		
}		
if (br_info_present) {		
if (sub_layer_profile_tier_level_info_present) {		
br_info(max_sub_layers_instream-1)	32*(s-1)	Table 7.19
} else {		
br_info(0)	32	Table 7.19
}		
}		
if (color_info_present) {		
color_info()	var	Table 7.20
}		
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', 'lhe1', 'vvc1', 'vvc2', 'vvi1', or 'vvi2', with semantic meaning for these codes as specified in ISO/IEC 14496-15 [36] 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. The values 0x00 and 0x08 to 0x3F are reserved.

sub_layer_profile_tier_level_info_present – This 1-bit Boolean flag shall indicate, when set to '1', that profile, tier, and level information is 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 [39]) 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. The value 7 is reserved.

tid_min – This 3-bit field shall indicate the minimum value of TemporalId (as defined in Rec. ITU-T H.265 [39]) of all access units for this video asset. tid_min shall be in the range of 0 to 6, inclusive. The value 7 is reserved.

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 7.13. 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 7.13. 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 7.13.

Table 7.13 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 [39].

Scalability Information

Table 7.14 Bit Stream Syntax for Scalability Information

Syntax	No. of Bits	Format
scalability_info() {		
asset_layer_id	6	uimsbf
scalable_type_id	2	uimsbf
}		

asset_layer_id – This 6-bit unsigned integer field shall specify the nuh_layer_id [39] for this asset. The value of asset_layer_id shall be in the range of 0 to 62, inclusive. The value 63 is reserved.

scalable_type_id – This 2-bit unsigned integer field shall specify the scalable profile or codec utilized for this asset. The value of scalable_type_id shall be in the range of 0 to 2, inclusive. The value 3 is reserved.

Table 7.15 Values of scalable_type_id

scalable_type_id	Meaning
'00'	HEVC Scalable Main 10 Profile (SHVC) [51]
'01'	MultiLayer VVC Main Profile [39]
'10'	MPEG-5 LCEVC Main Profile [52]
'11'	reserved

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 [38] 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.

MultiView Information

Table 7.16 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 [39] for the view represented by this asset. The value of view_nuh_layer_id shall be in the range of 0 to 62, inclusive. The value 63 is reserved.

view_pos – This 6-bit unsigned integer field shall specify the order of the view with nuh_layer_id [39] 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. The value 63 is reserved.

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

min_disp_with_offset – The value of this 11-bit unsigned integer field minus 1024 shall specify 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 – The value of 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.

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

Table 7.17 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) [39].

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 [34]. When set to '0', the flag shall indicate that the video asset does not include HEVC still pictures as defined in ISO/IEC 13818-1 [34].

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 [34]. When set to '0', the flag shall indicate that the video asset does not include any HEVC 24-hour picture as defined in ISO/IEC 13818-1 [34].

Picture Rate Information

Table 7.18 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 i^th^ temporal sub-layer of this video asset. The picture_ rate_code[i] code indicates the 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 = ATSC 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 pictures 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) [39] 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 these cases the picture_rate_code[i] shall be used to indicate the picture rate.

Bit Rate Information

Table 7.19 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) [39] 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) [39] shall apply.

Color Information

Table 7.20 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 as the elements with the same name in H.265 (10/2014) HEVC specification Section E.3.1 (VUI Parameter Semantics) [39].

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 [41] 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 [41] 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 0 and Table 7.21.

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.

Transfer Function Information

Table 7.21 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 [39] NAL unit [39] 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 [39]. 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 7.22) 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 7.22 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(), an ATSC 3.0 client is expected to not select such Asset for regular playout. If an 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 7.10.

7.2.3.4.1 Syntax and Semantics

The syntax for the audio_stream_properties_descriptor() shall conform to Table 7.23. The semantics of the fields in the audio_stream_properties_descriptor() shall be as given immediately below the table.

Table 7.23 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 7.32
}		
} /* end of for num_presentations loop*/		
if (multi_stream_info_present) {		
multi_stream_info()		Table 7.31
}		
if (emergency_info_time_present) {		
emergency_information_time_info()		Table 7.30
}		
}		
}		

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 this element shall be created according to the syntax described in RFC6381 [30] 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 [40].

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. This field, when set to '0', shall be an indication 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 [40].

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 [40].

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 '1' if the CompatibleProfileLevelSet() config extension element specified in ISO/IEC 23008-3, Clause 4.8.2.7 [40] is present, or to '0' otherwise.

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 7.24.

Table 7.24 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'
}		

In the case of MPEG-H Audio [9], this field shall contain the values of the 'mpegh3daProfileLevelIndication' and 'CompatibleSetIndication' fields as specified in ISO/IEC 23008-3 subclause 5.3.2 [40]. The syntax of the profile_level_indication field shall be as specified in Table 7.25.

Table 7.25 Syntax for MPEG-H Audio profile_level_indication()

Syntax	No. of Bits	Format
profile_level_indication() {		
mpegh3daProfileLevelIndication	8	uimbsf
CompatibleSetIndication	8	uimbsf
reserved	8	'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 [40].

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

When 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.

When 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 7.26.

Table 7.26 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 [37]. 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 7.27.

Table 7.27 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 greater than 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 [33].

In the case of AC-4 Audio [8], the language indicated by this field shall 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 shall 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 [40].

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 the k-th language in the language "for" loop above. Table 7.28 specifies the bit used to indicate if the Presentation contains support for a particular audio accessibility Service. When the bit specified in Table 7.28 is set to '1' it indicates the Presentation contains the corresponding audio accessibility Service. When the bit specified in Table 7.28 is set to '0' it indicates the Presentation does not contain the corresponding audio accessibility Service.

Table 7.28 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 is 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 [40]. 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 7.29. The role values shall correspond to the role scheme schemeIdUri="urn:mpeg:dash:role:2011" defined in ISO/IEC 23009-1:2014 MPEG DASH [65]. For a description of the role values, see Section 5.8.4.4 in ISO/IEC 23009-1:2014.

Table 7.29 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 7.30 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 the 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 7.31 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 7.32 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 7.33. 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 7.33 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 [33].

role – A 4-bit field that shall specify the purpose of the closed caption text as given in Table 7.34.

Table 7.34 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 7.35.

Table 7.35 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 ATSC Reserved.

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 [70].

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 [70].

7.2.4 DRM for MMTP

7.2.4.1 General

Content protection in MMT relies on ISO/IEC23001-7 Common encryption in ISO base media file format files [45]. Protection system specific and proprietary signaling information are delivered in two ways:

- by the MMT signaling messages, and
- carried in MPU in designated metadata boxes defined by ISO BMFF

Additional information is available in ATSC A/362 [76].

7.2.4.1.1 Service Signaling

When a Service is encrypted, this is signaled in the SLT by the Service@protected attribute and the Service@drmSystemID attribute as specified in Section 6.3.1. For MMT, this is signaled when the ComponentInfo@componentProtectedFlag of at least one component listed in BundleDescriptionMMT is set to true (See Section 7.2.1).

7.2.4.1.2 Use of the Content Protection Scheme

The scheme_code field of the security_properties_descriptor shall indicate the encryption scheme as specified by A/360, Section 5.7 [10]. The "`urn:mpeg:cenc:2013`" extension namespace is assumed for this field. The default_KID signaled in security_properties_descriptor for each component may be sufficient for the receiver to acquire a DRM license, or identify a previously acquired license that can be used to decrypt the component.

As the default_KID is signaled by security_properties_descriptor for each component, it allows a player to determine if a new license needs to be acquired for each component by comparing their default_KID attributes with each other, and with the default_KID attributes of stored licenses. A player can simply compare these KID strings and determine what unique licenses are necessary without interpreting license information specific to each DRM system.

7.2.4.1.3 Signaling of Information from Protection System Specific Header Box

A 'pssh' box (as specified in ISO/IEC 23001-7 [45]) is used by each DRM system with its registered SystemID, and is nominally stored in the movie box ('moov') and additionally may be present in the movie fragment box ('moof'). The information of the box shall be identical to that of the pssh data of the security_properties_descriptor carried in the MMT ATSC 3.0 Signaling Message by the broadcast server. As specified in Section6.3.2, initialization segments do not contain any pssh boxes as per DASH-IF IOP V5, Part 6, 6.3 [47] and Receivers can ignore such boxes when encountered.

7.2.4.1.4 Signaling of License Acquisition URL by the MMT Signaling Messages

Information to access a server for the ATSC receive to acquire license, the type of license and the URL to access the server for license acquisition may be added to the security_properties_descriptor carried in the MMT ATSC 3.0 Signaling Message by the broadcast server. More than one of following methods to acquire a valid license may be signaled.

- `license-1.0`. Direct license acquisition by ATSC3 receivers and the URI scheme is a valid endpoint for access.
- `groupLicense-1.0.` Provides a path for a group-based license. The URI may need to be parsed specific to that DRM System by the ATSC3 Receiver to access any local group licenses.
- `contentId-1.0.` Provides information for the DRM specific content identifier used to generate the KIDs. The URI should be parsed to extract relevant information using REST based notation.

7.2.4.1.5 Additional Signaling for Out-of-Order Delivery

In case of out of order delivery is used, i.e., media data from 'mdat' box is delivered before other boxes are delivered, encryption metadata is delivered as a signaling message so that the ATSC3 Receiver can start decryption of media data immediately without any further delay. The receiver uses the signaling message to acquire the information about the number of bytes of clear data and protected data of each subsamples of each sample, and the count of the encrypted blocks and the unencrypted blocks in the protection pattern for each sample groups are before it receives 'moof' box or 'moov' box.

The low_delay_decryption_information_descriptor() is carried by the MMT ATSC 3.0 Signaling Message, mmt_atsc3_message(), contains encryption metadata for a specific sample. The descriptor uses MPU_sequence_number, movie_fragment_sequence_number and sample_number to identify the specific sample the information in the descriptor is applied to.

7.2.4.2 MMT Signaling Message Support for Encryption and DRM Signaling

7.2.4.2.1 Descriptors

7.2.4.2.1.1 Security Properties Descriptor

For delivery of DRM-related information to prepare the receiver before the content is delivered, some information is sent as MMT signaling messages in addition to the ISOBMFF defined metadata boxes of the MPUs. A Security Properties Descriptor security_properties_descriptor() is carried by the MMT ATSC 3.0 Signaling Message, mmt_atsc3_message(), containing information for CENC and standardized license acquisition information. The syntax of security_properties_descriptor() shall be as provided in Table 7.36.

Table 7.36 Bit Stream Syntax for Security Properties Descriptor

Syntax	No. of Bits	Format
security_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
}		
scheme_code_present	1	bslbf
default_KID_present	1	bslbf
reserved	6	'111111'
if (scheme_code_present) {		
scheme_code	4*8	uimsbf
}		
if (default_KID_present) {		
default_KID	16*8	uimsbf
}		
number_of_systems	8	uimsbf
for (j=0; j<number_of_systems; j++) {		
system_UUID_present	1	bslbf
license_info_present	1	bslbf
pssh_present	1	bslbf
reserved	5	'11111'
if (system_UUID_present) {		
system_UUID	16*8	uimsbf
}		
if (license_info_present) {		
number_of_license_info	8	uimsbf
for (k=0; k<number_of_license_info; k++) {		
license_type	8	uimsbf
LA_URL_length	8	uimsbf
for (m=0; m<LA_URL_length; m++) {		
LA_URL_byte	8	bslbf
}		
}		
}		
if (pssh_present) {		
pssh_length	4*8	uimsbf
for (k=0; k<pssh_length; k++) {		
pssh_byte	8	bslbf
}		

} } } }		

descriptor_tag – A 16-bit unsigned integer field that shall have the value 0x000C, identifying this descriptor as the security_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 DRM protected assets described by this descriptor.

asset_id_length – A 32-bit unsigned integer field that shall specify the length in bytes of the DRM protected asset id.

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

scheme_code_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the element scheme_code is present. When set to '0', the flag shall indicate that the element scheme_code is not present.

default_KID_present – A 1-bit Boolean flag that shall indicate, when set to '1', that default KID for this DRM protected asset is present. When set to '0', this flag shall indicate that no default KID for this DRM protected asset is present.

scheme_code – A 32-bit unsigned integer field that shall specify a 4-character code for a protection scheme. The value of these four characters shall be as specified in [45].

default_KID_length – An 8-bit unsigned integer field that shall specify the length in bytes of the default KID for this DRM protected asset. The value of this field shall be set to 16.

default_KID_byte – An 8-bit unsigned integer field that shall contain a byte of the default KID.

number_of_systems – An 8-bit unsigned integer field that shall specify the number of protection systems represented.

system_UUID_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the protection system UUID is present. When set to '0', the flag shall indicate that the protection system UUID information is not present and the protection system UUID is not specified.

license_info_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the URL information for direct license server access is present.

pssh_present – A 1-bit Boolean flag that shall indicate, when set to '1', that the pssh box data is present.

system_UUID – Sixteen unsigned integer bytes that shall contain the UUID of the protection system as specified in DASH-IF Identifiers/ Content Protection [46].

number_of_license_info – An 8-bit unsigned integer field that shall specify the number of license information blocks signaled for the current asset.

license_type – An 8-bit unsigned integer field that shall specify a type of applicable license as specified in Table 7.37.

LA_URL_length – An 8-bit unsigned integer field that shall specify the length in bytes of the LA_URL (the URL for license acquisition).

LA_URL_byte – An 8-bit field that shall contain a byte of the LA_URL. The LA_URL shall be a dashif:Laurl as specified by DASH-IF IOP V5 [47].

pssh_length – A 32-bit unsigned integer field that shall specify the length in bytes of the pssh.

pssh_byte – An 8-bit field that shall contain a byte of the encoded pssh. The pssh shall be encoded as base64 as specified in RFC 4648 [48], Section 4.

Table 7.37 Code Values for license_type

license_type	Meaning
0x00	ATSC Reserved
0x01	`license-1.0`, Direct license acquisition by ATSC3 receivers and the URI scheme is a valid endpoint for access.
0x02	`groupLicense-1.0`, Provides a path for a group-based license. The URI may need to be parsed specific to that DRM System by the ATSC3 Receiver to access any local group licenses.
0x03	`contentId-1.0`, Provides information for the DRM specific content identifier used to generate the KIDs. The URI should be parsed to extract relevant information using REST based notation.
0x04~0xFF	Industry Reserved. See ATSC Code Point Registry.

7.2.4.2.1.2 Low Delay Decryption Information Descriptor

This message supports two modes of operation, MPU-based operation mode and sample-based operation mode, which is distinguished by operation_mode. In MPU-based operation mode, descriptor contains information about all samples in an MPU with a single descriptor. CencSampleEncryptionInformationGroupEntry sample group description structures for all sample groups defined in MPU is provided by a single descriptor in this mode. In addition, a list of samples associated to each sample groups and CencSampleAuxiliaryDataFormat for all of them are also provided by a single descriptor in MPEG-based operation mode. In sample-based operation mode, one descriptor provides information about a single sample. A descriptor carries SampleEncryptionBox associated with this sample. In addition, the descriptor in this mode carries CencSampleEncryptionInformationGroupEntry sample group description structures for the sample group the sample this descriptor is associated with, whenever needed. If the sample group description structure for the same sample group has been already delivered, it can be skipped. See Table 7.38.

Table 7.38 Low Delay Decryption Information Descriptor

Syntax	No. of Bits	Format
low_delay_decryption_information_descriptor() {		
descriptor_tag	16	uimsbf
descriptor_length	16	uimsbf
MPU_sequence_number	32	uimsbf
operation_mode	3	uimsbf
sample_group_info_present	1	uimsbf
reserved	4	1111
if (**operation_mode** == 1) {		
movie_fragment_sequence_number	32	uimsbf
sample_ number	8	uimsbf
sample_group_number	8	uimsbf
if (**sample_group_info_present** ==1) {		
size_of_seig_box	32	uimsbf
for (k=0; k<bytes_of_seig_box; k++) {		

seig_box_byte	8	uimsbf
}		
size_of_senc_box	32	uimsbf
for (n=0; n<size_of_senc_info; n++) {		
senc_box_byte	8	uimsbf
}		
}		
if (**operation_mode** == 2) {		
number_of_movie_fragments	8	uimsbf
for (i=0; i<number_of_movie_fragments; i++) {		
movie_fragment_sequence_number	32	uimsbf
number_of_sample_groups	8	uimsbf
for (j=0; j<number_of_sample_groups; j++) {		
size_of_seig_box	32	uimsbf
for (k=0; k<bytes_of_seig_box; k++) {		
seig_box_byte	8	uimsbf
}		
number_of_samples	8	uimsbf
for (l=0; l<number_of_samples; l++) {		
sample_ number	8	uimsbf
size_of_aux_info	32	uimsbf
for (m=0; m<size_of_aux_info; m++) {		
aux_info_byte	8	uimsbf
}		
}		
}		
}		
}		

descriptor_tag – A 16-bit unsigned integer field that shall have the value 0x000E, identifying this descriptor as the low_delay_decryption_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.

MPU_sequence_number – An 32-bit unsigned integer field that shall specify sequence number of the MPU this descriptor is applied to.

operation_mode – An 3-bit unsigned integer field that shall specify the mode of operation this descriptor is used for as specified in Table 7.39.

Table 7.39 Code Values for operation_mode

license_type	Meaning
0	ATSC Reserved
1	Sample-based operation mode. The descriptor carries decryption information for the sample identified by the combination of MPU_sequence_number, movie_fragment_sequence_number and sample_number.
2	MPU-based operation mode. The descriptor carries decryption information for all samples of the MPU identified by MPU_sequence_number.
3~7	Industry Reserved. See ATSC Code Point Registry.

sample_group_info_present – A 1-bit Boolean flag that shall indicate whether CencSampleEncryptionInformationGroupEntry is carried or not. When set to '1', that CencSampleEncryptionInformationGroupEntry is present.

movie_fragement_sequence_number – A 32-bit unsigned integer field that shall specify the sequence number of the movie fragment to which the information is applied.

sample_number – A 32-bit unsigned integer field that shall specify the sequence number of a sample within a movie fragment to which the information is applied.

size_of_seig_box – A 32-bit unsigned integer field that shall specify the size of CencSampleEncryptionInformationGroupEntry box for a sample group. When the operation_mode is equal to '1', this field will indicate the size of CencSampleEncryptionInformationGroupEntry box for a sample group the sample this descriptor is associated with. When the operation_mode is equal to '2', this field will indicate the size of CencSampleEncryptionInformationGroupEntry box for a sample group whose group_description_index is equal to j.

seig_box_byte – An 8-bit unsigned integer field that shall contain the kth byte of the CencSampleEncryptionInformationGroupEntry box for a sample group. When the operation_mode is equal to '1', this field will carry the kth byte of CencSampleEncryptionInformationGroupEntry box for a sample group the sample this descriptor is associated with. When the operation_mode is equal to '2', this field will carry the kth byte of CencSampleEncryptionInformationGroupEntry box for a sample group whose group_description_index is equal to j.

size_of_senc_box – A 32-bit unsigned integer field that shall specify the size of SampleEncryptionBox for the sample with which this descriptor is associated.

senc_box_byte – An 8-bit unsigned integer field that shall contain the *n*th byte of the SampleEncryptionBox box for the sample with which this descriptor is associated.

number_of_movie_fragments – An 8-bit unsigned integer field that shall specify the number of movie fragments contained in the MPU with which this descriptor is applied.

number_of_sample_groups – An 8-bit unsigned integer field that shall specify the number of sample groups defined in the movie fragment information in the current loop.

number_of_samples – An 8-bit unsigned integer field that shall specify the number of samples that belong to the sample group whose group_description_index is equal to j.

size_of_aux_info – A 32-bit unsigned integer field that shall specify the size of the CencSampleAuxiliaryDataFormat data structure applied to the sample currently being described.

aux_info_byte – An 8-bit unsigned integer field that shall contain the *m*th byte of the CencSampleAuxiliaryDataFormat.

7.2.4.2.2 MMT Hint Sample for Low Delay Decryption

7.2.4.2.2.1 General

When out of order delivery mode is used, i.e., media data from 'mdat' box is delivered before other boxes are delivered, metadata required for decryption of that media data is delivered as hint samples in order to reduce the number of signaling messages to be processed. The sender copies the information for decryption to LowDelayProcessingInfo for each hint sample from the SampleEncryptionBox of the corresponding media samples.

The semantics of each field of LowDelayProcessingInfo shall be identical to the fields of the SampleEncryptionBox with the identical name. As the LowDelayProcessingInfo only contains the information for a single sample, the field about the number of samples of the SampleEncryptionBox is not copied.

The sender may deliver such LowDelayProcessingInfo together with media samples. The receiver may use such LowDelayProcessingInfo instead of SampleEncryptionBox for decryption of the media sample without waiting for a 'moof' box or 'moov' box containing a SampleEncryptionBox to be delivered. Note that a complete SampleEncryptionBox corresponding the media samples of a movie fragment is not replaced by LowDelayProcessingInfo and is delivered as a part of a 'moof' box, regardless of the use of LowDelayProcessingInfo.

7.2.4.2.2.2 Sample Description Format

A sample entry type of 'mtha' shall be used for MMT hint tracks. This sample entry type shall include an indication of the version and includes reserved bits for the introduction of new flags for future extensions.

7.2.4.2.2.2.1 Syntax

Note that the following syntax follows the MPEG style guide.

```
aligned(8) class MMTHintATSC3SampleEntry() extends SampleEntry('mtha') {
   unsigned int(16) hinttrackversion = 1;
   unsigned int(16) highestcompatibleversion = 1;
   unsigned int(1) has_mfus_flag;
   unsigned int(1) is_timed;
   unsigned int(1) decryption_info_flag;
   unsigned int(13) reserved;
}
```

7.2.4.2.2.2.2 Semantics

`hinttrackversion` – Specifies the version of this hint track. The current version is 1.

`highestcompatibleversion` – Specifies the oldest version with which this hint track is backward-compatible.

`has_mfus_flag` – A flag indicating whether the MPUs are fragmented into MFUs or not. If this flag is set to FALSE, the hint track applies to the complete MPU, i.e., each track fragment will have a single sample. Otherwise, each hint sample applies to an MFU.

`is_timed` – Indicates whether the media data hinted by this track is timed data or non-timed data.

`low_delay_decryption_info_flag` – Indicates whether the low delay decryption information of the media data hinted by this track is included in the hint samples or not. If the value of this field is set to '1', then the information is included in the samples. If the value of this field is set to '0', then the information is not included in the samples.

7.2.4.2.2.3 Sample Format

The sample structure of MMT hint track for ATSC 3 varies by the value of `hinttrackversion` field set by the sample entry box. According to the value of `hinttrackversion` field, value of the different set of flags set by the sample entry box are evaluated and additional data structure are included in the end of the hint sample.

7.2.4.2.2.3.1 Syntax

```
aligned(8) class MMTHSampleATSC3 extends Full Box ('mhas', version, 0)
{
   unsigned int(32) sequence_number;
   if (is_timed) {
        signed int(8) trackrefindex;
        unsigned int(32) movie_fragment_sequence_number
        unsigned int(32) samplenumber;
        unsigned int(8)  priority;
        unsigned int(8)  dependency_counter;
```

```
        unsigned int(32) offset;
        unsigned int(32) length;
    } else {
        unsigned int(16) item_ID;
    }
    if(version == 1){
        if(low_delay_decryption_info_flag){
                unsigned int(24) SampleEnctryptionBoxflags;
                LowDelayDecryptionInfo()
        }
    }
}
```

7.2.4.2.2.3.2 Semantics

`hinttrackversion` – Contains the value set by the sample entry this sample belongs to.

`sequence_number` – An integer number that indicates the sequencing order of this MFU within the MPU. Discontinuity of sequence numbers in an MPU is allowed to indicate that certain MFUs (whose sequence number was not in the sequence) were not processed after packetization of MPU. Examples of MFU processing are delivery and caching by underlying network entity.

`movie_fragment_sequence_number` – The sequence number of the movie fragment to which the media data of this MFU belongs (see ISO/IEC 14496-12 [35], Section 8.8.5). The `movie_fragment_sequence_number` in an MPU shall start by "1" for the first movie fragment of the MPU and shall be incremented by "1" for every succeeding movie fragment in that MPU.

`trackrefindex` – The ID of the media track from which the MFU data is extracted.

`samplenumber` – The number of the sample from which this MFU is extracted. Sample number `n` points to the *n*th sample from accumulated samples of the current movie fragment. The sample number of the first sample in the movie fragment is set to "1" (see ISO/IEC 14496-12 [35], Section 8.8.8).

`priority` – Indicates the priority of the MFU relative to other MFUs within an MPU.

`dependency_counter` – Indicates the number of MFUs whose decoding is dependent on this MFU. The value of this field is equal to the number of subsequent MFUs in the order of `sequence_number` that may not be correctly decoded without this MFU. For example, if the value of this field is equal to `n`, then `n` subsequent MFUs may not be correctly decoded without this MFU.

`offset` – The offset of the media data contained in this MFU. The offset base is the beginning of the containing "`mdat`" box.

`length` – The length of the data corresponding to this MFU in bytes.

`item_ID` – For non-timed media data, this is the ID of the item that is contained in this MFU.

`hinttrackversion` – The version of this hint track indicated by the sample entry of this track.

`SamepleEncryptionBoxflags` – Carries the same value of the flags field of `SampleEncryptionBox` to which this sample belongs.

`low_delay_decryption_info_flag` – Contains the value set by the sample entry this sample belongs to.

7.2.4.2.2.4 Low Delay Decryption Info

7.2.4.2.2.4.1 Syntax

```
aligned(8) class LowDelayDecryptionInfo()
```

```
{

   unsigned int(8) Per_Sample_IV_Size
   unsigned int(Per_Sample_IV_Size*8)  InitializationVector;
   if (flags & 0x000002)
   {
        unsigned int(16)  subsample_count;
        {
                unsigned int(16)  BytesOfClearData;
                unsigned int(32)  BytesOfProtectedData;
        }[ subsample_count ]
   }
}
```

7.2.4.2.2.4.2 Semantics

`Per_Sample_IV_Size` – Same as the semantics of `Per_Sample_IV_Size` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `Per_Sample_IV_Size` field of `SampleEncryptionBox` corresponding the same sample.

`InitializationVector` – Same as the semantics of `InitializationVector` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `InitializationVector` field of `SampleEncryptionBox` corresponding the same sample.

`subsample_count` – Same as the semantics of `subsample_count` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `subsample_count` field of `SampleEncryptionBox` corresponding the same sample.

`BytesOfClearData` – Same as the semantics of `BytesOfClearData` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `BytesOfClearData` field of `SampleEncryptionBox` corresponding the same sample.

`BytesOfProtectedData` – Same as the semantics of `BytesOfClearData` field of `SampleEncryptionBox` in ISO/IEC 23001-7. This field carries the same value of the `BytesOfProtectedData` field of `SampleEncryptionBox` corresponding the same sample.

7.2.4.2.3 MPU Fragment Type for MMT Hint Sample for ATSC 3

As hint samples are not an independent media but an integral part of an MPU, they shall be delivered in the same MPU as the media data they are describing, which means that they shall be delivered by the MMTP packets whose value of the packet_id field is the same as that of the MMTP packet delivering media data they are describing. The hint samples shall be delivered immediately before their associated MFUs.

An MPU is delivered with the MMTP packets the value of whose packet_id field of the MMTP packet header is same. When an MPU is delivered, the MPU Fragment Type, FT, field of the MMTP payload header is used to distinguish the types of data from a MPU. To differentiate the packets containing hint sample data from the packets containing media data new fragment type value is defined for hint sample as shown in Table 7.39.

Table 7.40 Data Type and Definition of Data Unit

FT	Description	Content
0	MPU metadata	Contains the ftyp, mmpu, moov, and meta boxes, as well as any other boxes that appear in between.
1	Movie fragment metadata	Contains the moof box and the mdat box, excluding all media data inside the mdat box but including any chunks of auxiliary sample information.
2	MFU	Contains a sample or subsample of timed media data or an item of non-timed media data.
3	MMT hint sample for ATSC 3	Contains a sample data of MMT hint sample for ATSC 3 type describing media data delivered by the MMT packets with same packet_id.
4 ~ 15	Reserved for private use	reserved

The DU header for timed media is used for the hint samples. The value of the fields of the DU header for hint sample are set to the same value of the DU header for media sample which is described by that hint sample except the offset field.

7.2.5 SIGNED_MMT_MESSAGE

All MMTP messages with MMTP packet type == 0x2, except those with a message_id of 0x8101 (signed_mmt_message()), shall be carried as a message_instance() within the signed_mmt_message() structure as specified in this section. When any MMTP-specific signaling message as specified in Clause 10 of 23008-1 [38] and when any MMT ATSC3 (MA3) messages as specified in Section 7.2.3.1 are transmitted, such messages shall be transmitted as a message_instance() in a signed_mmt_message().

Table 7.41 Bitstream Syntax for signed_mmt_message()

Syntax	No. of Bits	Format
signed_mmt_message() {		
message_id	16	uimsbf
version	8	uimsbf
length	32	uimsbf
message_payload {		
message_instance()		
atsc3_signature_length	16	uimsbf
for (i=0;i<atsc3_signature_length;i++) {		
atsc3_signature_byte	8	uimsbf
}		
}		
}		

Signing only MMTP packets with a type field equal to 0x2 limits signing to MMTP packets carrying signaling messages and MA3 messages and excludes signing from MMTP packets carrying Assets.

message_id – A 16-bit unsigned integer field that shall uniquely identify the signed_mmt_message(). The value of this field shall be 0x8101.

version – An 8-bit unsigned integer field that shall indicate the version of the signed_mmt_message(). When the message_instance() with a particular message_id attribute changes, the version of the signed_mmt_message() shall change. When there are two or more message_instance()s with different message_id attributes, the version of the signed_mmt_message() associated with those message_id attributes shall differ. The version attribute value shall not be reused until after a signed_mmt_message() with the associated message_id and an updated version attribute has been emitted.

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

message_payload – The message payload includes the message_instance() that is signed along with its signature.

message_instance() – The signed MMT signaling message instance which shall be the mmt_atsc3_message() or a message specified in ISO/IEC 23008-1 [38] Clause 10.

atsc3_signature_length – A 16-bit unsigned integer field that shall provide the length of the signature data contained in this message.

atsc3_signature_byte – An 8-bit unsigned integer field that shall contain a byte of the signature of this message. The signature content shall be based on the message_instance() data only. The signature shall be as described in A/360 [10] Section 5.2.2.5, "Signatures for MMT Messages".

The following sequence is an example of the usage of the **version** attribute:

1) message_id attribute of the message_instance() = A; version attribute of the message_instance() = 1; version attribute of the signed_mmt_message() = 1;
2) message_id attribute of the message_instance() = B; version attribute of the message_instance() = 1; version attribute of the signed_mmt_message() = 2;
3) message_id attribute of the message_instance() = A; version attribute of the message_instance() = 1; version attribute of the signed_mmt_message() = 1;
4) message_id attribute of the message_instance() = B; version attribute of the message_instance() = 1; version attribute of the signed_mmt_message() = 2;
5) message_id attribute of the message_instance() = A; version attribute of the message_instance() = 2; version attribute of the signed_mmt_message() = 3;
6) message_id attribute of the message_instance() = B; version attribute of the message_instance() = 2; version attribute of the signed_mmt_message() = 4;

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 ABNF [23] rules and found in ContentAdvisoryRatings-20210420.abnf. When the content in the service is rated "exempt" this shall be the null string. For North America, "exempt" shall be as defined in CTA 766 [11].

```
ContentAdvisoryDescriptor =  RegionRating

RegionRating = [ RegionIdentifier ] "," RatingDescription "," Rating

RegionIdentifier = ByteNotZero

RatingDescription = "'" *ACHAR "'"

Rating = 1*RatingPair

RatingPair = "{" [ RatingDimension SP ]  RatingLevel "}"

RatingDimension = Byte

RatingLevel = "'" 1*BCHAR "'"

Byte = "0" / ByteNotZero

ByteNotZero  = %x31-39          ; 1-9
      / %x31-39 DIGIT    ; 10 - 99
      / "1" DIGIT DIGIT  ; 100 - 199
      / "2" %x30-34 DIGIT ; 200 - 249
      / "25" %x30-35     ; 250 - 255

ACHAR = %x20-21    ; ascii characters excluding single and double quote marks.
        / %x23-26
        / %x28-7E

BCHAR = %x21; ascii characters excluding space, single and double
        / %x23-26  ; quote marks.
        / %x28-7E
```

The content advisory information XML data as specified in each ContentAdvisoryRatings element may be transformed into a content advisory ratings string using the XSL transformation found in MakeContentAdvisoryString-20210402.xsl. This script is informational and does not override the above ABNF.

Table 7.42 illustrates some example cases.

Table 7.42 Example Content Advisory Rating Strings

<table>
<tr><th>Rating Region</th><th>Program Rating</th><th colspan="2">RRT Reference</th><th>Content Advisory Rating String</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">TV-PG-D-L-S-V</td><td>Dimension</td><td>Level</td><td rowspan="6">"1,'TV-PG D-L-S-V',{0 'TV-PG'}{1 'D'}{2 'L'}{3 'S'}{4 'V'}"</td></tr>
<tr><td>0</td><td>3</td></tr>
<tr><td>1</td><td>1</td></tr>
<tr><td>2</td><td>1</td></tr>
<tr><td>3</td><td>1</td></tr>
<tr><td>4</td><td>1</td></tr>
<tr><td rowspan="3">2</td><td rowspan="2">Canadian English Language Rating: 14+</td><td>Dimension</td><td>Level</td><td rowspan="3">"2,'13+(fr)/14+(en)',{0 '14+'}{1 '13+'}"</td></tr>
<tr><td>0</td><td>5</td></tr>
<tr><td>Canadian French Language Rating: 13+</td><td>1</td><td>3</td></tr>
</table>

In the case that a single program is rated for both rating regions 1 and 2, for the example in Table 7.42, the string would be captured in two ContentRating elements as follows:

```
<Rating schemeIdUri="http://dashif.org/guidelines/dash-atsc-RRTrating:1
  value="1,'TV-PG D-L-S-V',{0 'TV-PG'}{1 'D'}{2 'L'}{3 'S'}{4 'V'}”/>
<Rating schemeIdUri="http://dashif.org/guidelines/dash-atsc-RRTrating:1
         value="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 7.8. 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]. Note that A/332 includes language specification and unconstrained Unicode text, which is not supported in CTA-766 [11] or the Content Advisory Rating Strings defined in the previous section.

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 7.8. 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 or repair flows on LCT channels. Delivery of Service content or NRT content via the repair flow, without an associated source flow, could be used for certain deployment scenarios for which the broadcaster expects all targeted receivers to be AL-FEC capable. Such scenarios might include delivery of Services a) intended strictly for mobile reception, b) intended only in certain geographical areas, 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 that provides 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 Sections 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 [27] 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 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 7.4. 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 [54]), 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 [54].

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 and stable time for synchronization of service components.

Network servers for both broadcast and broadband Components of a Service shall synchronize to a UTC source. A SATNAV system (such as GNSS or other source with similar accuracy and stability) is sufficient.

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 [65].
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 and can be used by receivers when synchronized playback is desired. This mode is not described here as there is no ATSC 3.0 requirement for synchronized playback across multiple devices. Note that this mode causes an additional delay in channel change times.

8.1.1.4 ROUTE/DASH System Model

Figure 8.1 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 [34]. 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 the ROUTE/DASH system model. 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.

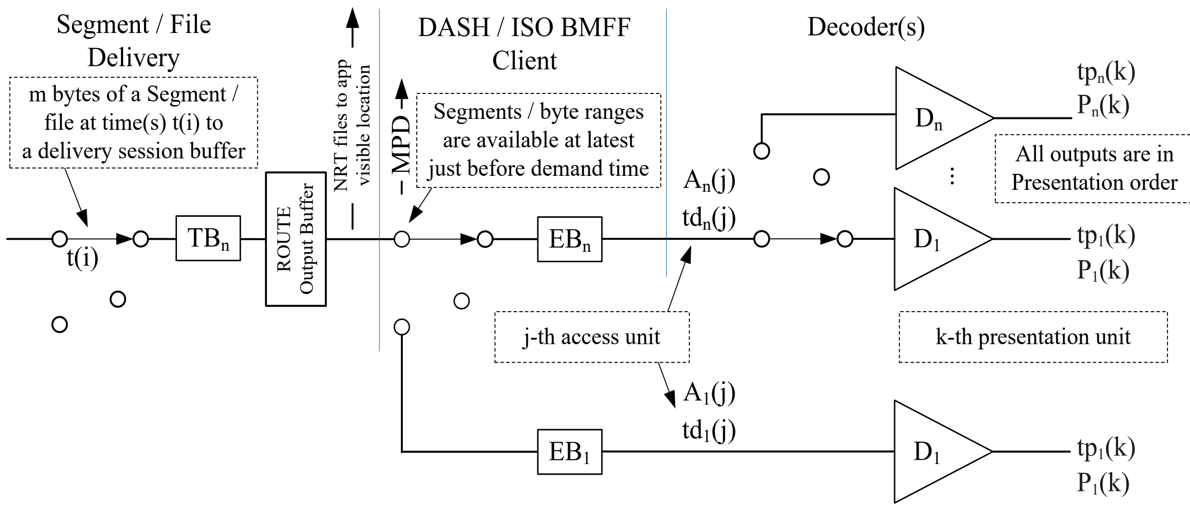

Figure 8.1 ROUTE/DASH system model.

There are a number of distinctions for the ROUTE/DASH system relative to the MPEG-2 Systems [34] 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.

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 [34] 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 make 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 j^{th} 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 j^{th} decoded media frame

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

$P_n(k)$ is the k^{th} 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 k^{th} 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.

> **NOTICE**: Implementer's attention is called to the state of MDE specification detail for: 1) the format of the "MDE data block"; and 2) the signaling of the "MDE data block". Additional design work may be needed for interoperability. Future addition of further data fragmentation methods (also known as "chunking") to reduce latency and possibly acquisition time may impact the MDE details. This applies to Sections 8.1.1.1, 8.1.1.3, 8.1.1.5.2, 8.1.1.5.3, and A.3.7.

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 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.

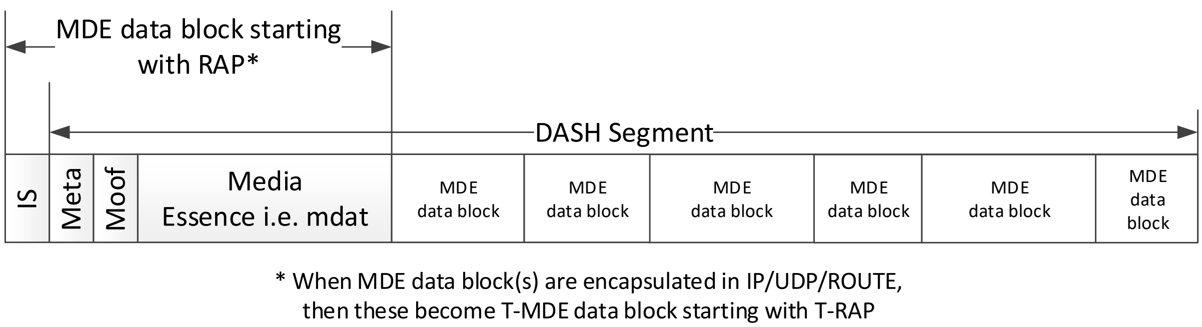

Figure 8.2 Concept of an MDE data block starting with a RAP.

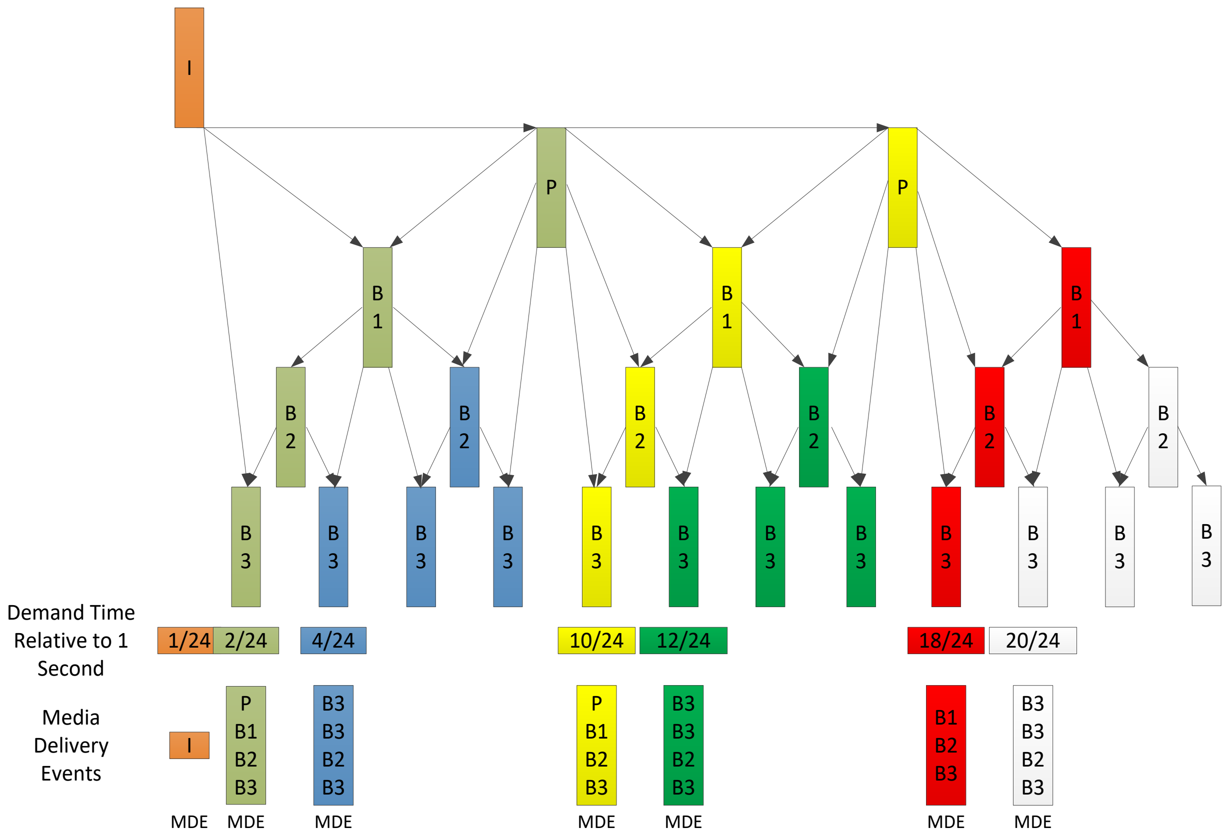

Figure 8.3 Illustration of an MDE data block at the video level.

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. The value in the register on the right-hand side of the figure denotes transport buffer fullness in bytes.

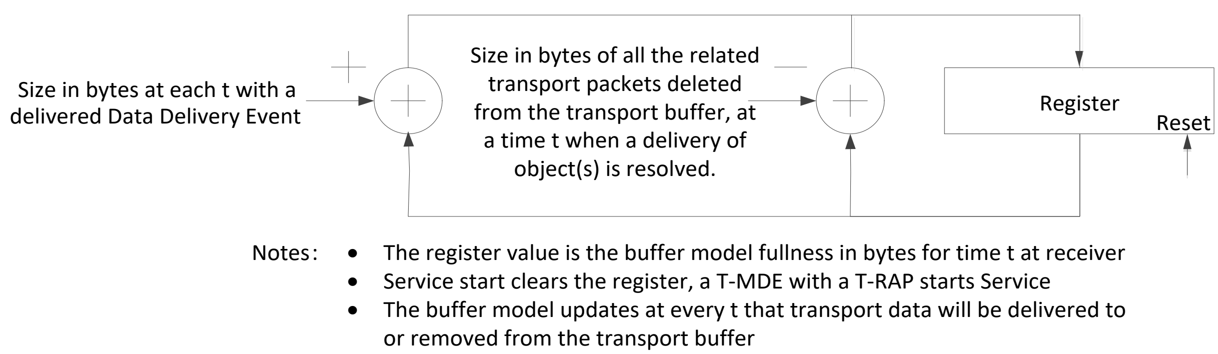

Figure 8.4 ROUTE transport buffer model.

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) is 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 has object physical layer durations in the range of the soft-decision FEC Frame in the physical layer. AL-FEC is applicable to ROUTE applications for both QoS and efficiency improvements. Users may tolerate up to 1% lost time for streaming video[9], 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.

8.1.1.7 Timing and Buffer Model Considerations with IMSC1

Notwithstanding the details of Sections 8.1.1.1 – 8.1.1.6, there are some additional considerations when transmitting IMSC1. Unlike video and audio tracks, IMSC1 text tracks require that the Receiver acquire all LCT packets of the Segment before decoding can begin. The IMSC1 payload

[9] One percent lost time is a more stringent requirement than ESR 5 as described in ITU-R Document 6E/64-E [66].

consists of a single IMSC1 "document" where all structures must be available to the Receiver. This is different than video and audio where decoding (and presenting) can start after a modest amount of data has been acquired, but considerably less than a typical 2-second duration Segment.

Nominal coding of video, audio and IMSC1 without special handling of IMSC1 LCT packets is shown in Figure 8.5.

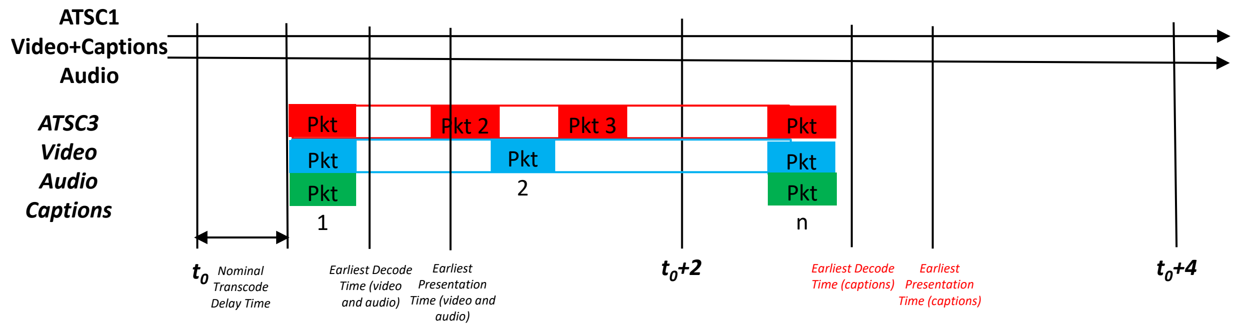

Figure 8.5 Typical Segment LCT packet emission.

The last LCT packet of an IMSC1 Segment cannot be transmitted until after receipt of the last of the (2-sec) input captions. Therefore, IMSC1 decoding and presentation times are either a Segment duration late (resulting in a 2 sec delay in caption text on screen) or the presentation time is "too late" relative to video and audio and nothing is displayed by the Receiver.

It is possible for the Receiver to compensate for this skew by buffering of full segments and delaying the decoding and presentation of video and audio until after the last IMSC1 LCT packet. However, this results in an unsatisfactory viewer experience of poor service acquisition time, delaying it by a Segment duration, e.g. 2 more seconds after acquiring the beginning of a Segment (1 second on average for a 2-second Segment).

To address this in the encoding/packaging prior to emission, in addition to other transmission requirements (notably buffer model constraints):

1) All Segment boundaries for all tracks that are synchronized (nominally video, audio, and captions) shall be aligned in time (start, end or both) with the track with the longest duration Segment;
2) All LCT packets shall be contained within their Segment start and end times; and
3) For each Segment with IMSC1 caption data, the last LCT packet carrying the last data of the IMSC1 document shall be emitted in the first 400ms of the Segment.

One encoder/packager side solution is that the encoder/packager could delay the output of video and audio to be aligned with one full Segment of IMSC1 coded as shown in Figure 8.8.

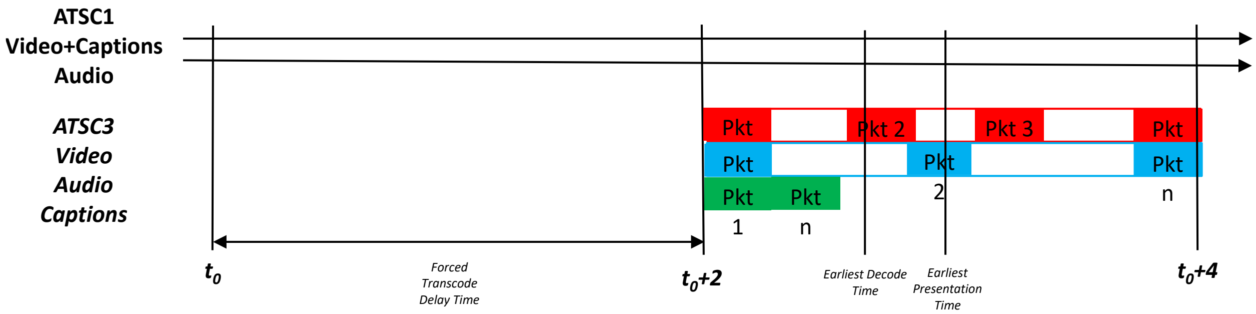

Figure 8.6 Recommended Segment LCT packet emission.

Note that the IMSC1 Segment is transmitted up front in the Segment timeline and can be decoded and presented at the same time as video and audio. From the Receiver (and viewer) perspective the service can be acquired rapidly and IMSC1 remains in sync with video and audio. This might increase the encoding time of Segments depending on current encoding delays today, and thus adversely impact low latency applications.

An alternative encoder solution is reducing the IMSC1 Segment duration to roughly the video and audio decode and presentation delay, e.g. ~400ms. However, this introduces:

1) notable (5x) the Receiver computation to decode and display IMSC1 compared to 2s Segments;
2) does not conform to CMAF minimum Segment duration of 960ms (but could be corrected by using CMAF Fragments); and
3) introduces VBR coding for IMSC1.

IMSC1 decoding is not necessarily performed with hardware assistance and might be performed with a general purpose processor with limited performance. In order to simplify processing and allow adequate time for decoding while minimizing the necessity of receiver-side buffering of audio and video, each IMSC1 document (sample) shall be delivered in its entirety at least 1 second before its associated presentation time.

8.1.2 MMTP/MPU

MMTP is a protocol designed to deliver MPUs as specified in Clause 9 of ISO/IEC 23008-1 [38]. MMTP provides a number of useful features for streaming delivery of MPUs via a unidirectional delivery network such as:

- The media-aware packetization of MPUs
- The multiplexing of various 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 [38]. 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 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 [64].

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 Component is considered as an MMT Asset having a unique identifier as defined in ISO/IEC 23008-1 [38]. 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 [38]. 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 [38].

Figure 8.7 and Figure 8.8 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 8.7, 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 8.8, a MMT Package is delivered over two MMTP sessions.

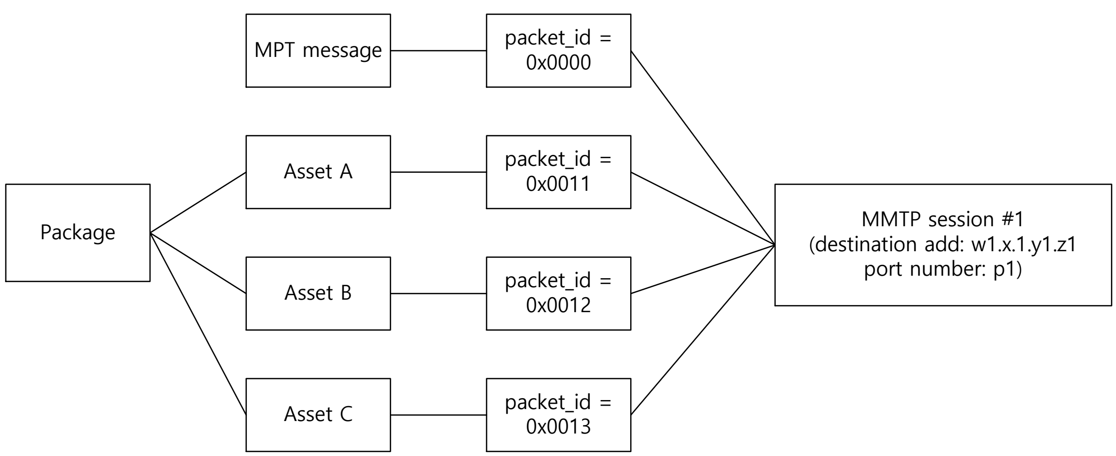

Figure 8.7 An MMT Package delivered over a single MMTP packet flow.

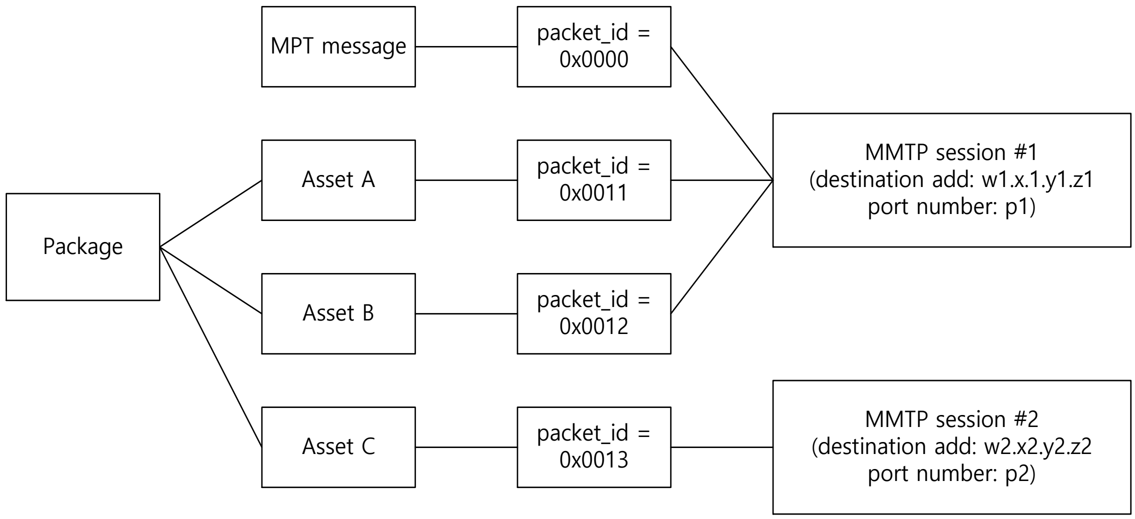

Figure 8.8 An MMT Package delivered over two MMTP packet flows.

8.1.2.1.2 Constraints on MPU

MPUs as specified in Clause 7 of ISO/IEC 23008-1 [38] 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 [38] 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 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 [38] 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 [25].
- 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 [38] 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 8.9.

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:

$$5 \text{ seconds} \times (1.2 \times R_{max}).$$

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 [39] 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.

o 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 [38]. 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:

$$1 \text{ second} \times (1.2 \times R_{max}).$$

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 [39] 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.

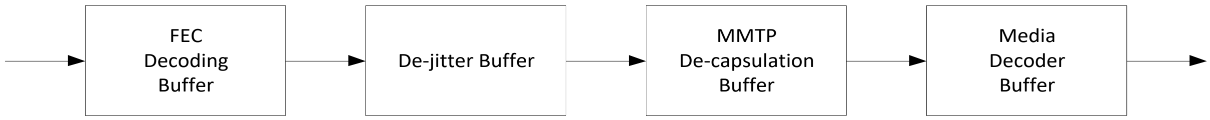

Figure 8.9 Receiver buffer model consuming MMTP sessions.

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 [34] delivery. The media-aware packetization of MPUs into MMTP packets allows interleaving of packets thereby reducing latency by the receiver during initial acquisition of the 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 [38]. 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[10] sequence number and a sample number.

Figure 8.10 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 8.10 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.

[10] "movie fragment" is as defined by ISO/IEC 23009-1:2014 [65].

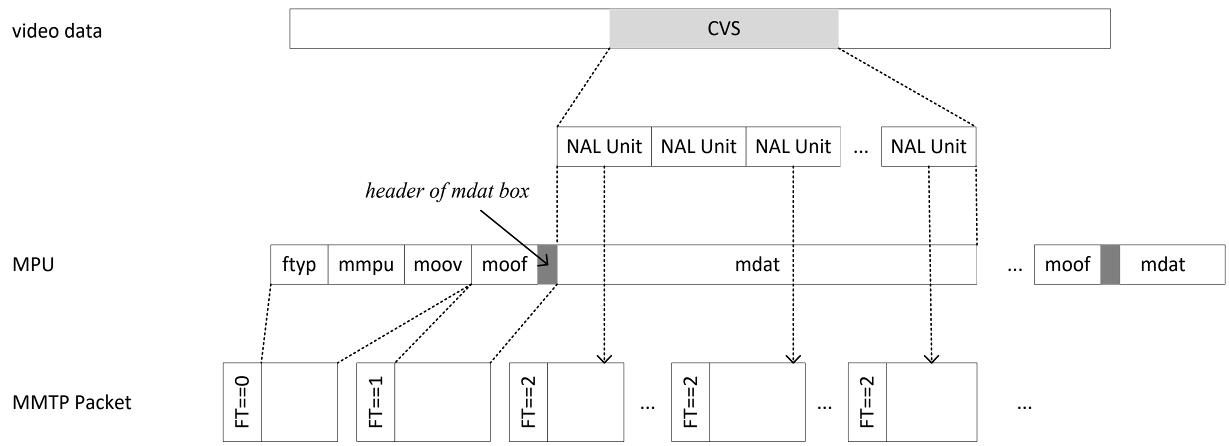

Figure 8.10 Example of media aware packetization of MPU.

8.1.2.3 Synchronization

The MPU_timestamp_descriptor as defined in subclause 10.5.2 of ISO/IEC 23008-1 [38] 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 calculated by using the rule in the ISO BMFF specification [35].

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 7.4. The access information for the S-TSID metadata fragment shall be provided by a **ROUTEComponent** element in the MMT-specific USBD as defined in Table 7.8.

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 [38] 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 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 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 8.2.1.2.3.

8.2.1.1 Synchronization

Figure 8.11 illustrates hybrid mode operation and synchronization within the DASH Client.

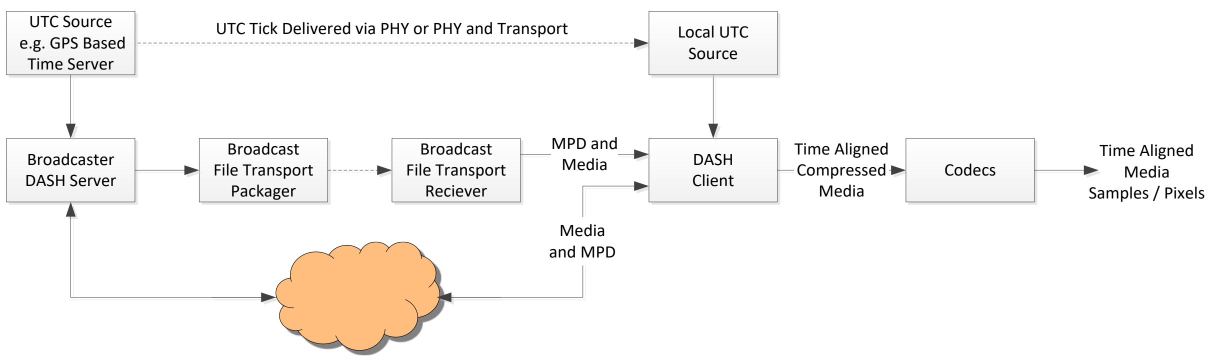

Figure 8.11 ROUTE/DASH hybrid synchronization.

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 [65] 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, but 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 ROUTE Service Configurations

8.2.1.2.1 Broadcast

In this Service Configuration, all Components shall be carried exclusively over broadcast (ROUTE); i.e., no Components are delivered over broadband (HTTPS). This Service Configuration is signaled by `SLT.Service@configuration`="`Broadcast`". If `@configuration` is not present, the value is (by default) "`Broadcast`".

If `SLT.Service@configuration`="`Broadcast`", `MPD.BaseURL` carries no meaning.

8.2.1.2.2 Broadband

In this Service Configuration, all Components shall be carried exclusively over broadband (i.e., HTTPS) – i.e. no Components are delivered via broadcast. This Service Configuration is signaled by `SLT.Service@configuration`="`Broadband`".

If the configuration is Broadband then `SLT.Service@configuration` shall be present and set to "`Broadband`" and `MPD.BaseURL` shall be present.

Access to the service should be protected in a manner that reasonably precludes viewing broadband content outside the authorized geographic area. The Essential Components should be encrypted.

In addition:

- `SLT.Service@broadbandAccessRequired` shall be set to "`true`".

If the USBD is present, then:

- **`BundleDescriptionROUTE.UserServiceDescription.DeliveryMethod.BroadcastAppService`** shall be absent.
- **`BundleDescriptionROUTE.UserServiceDescription.DeliveryMethod.UnicastAppService`** shall be present and **`BasePattern`** shall be set to the value of **`MPD.BaseURL`**.

8.2.1.2.3 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 8.12. 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 7.8. When there is a dependency between an MPU delivered by broadcast and a DASH Segment delivered by broadband, the MMTP-specific USBD contains child element **BroadbandComponentInfo** under **BroadbandComponent** element as defined in Section 7.2.1.1.

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.

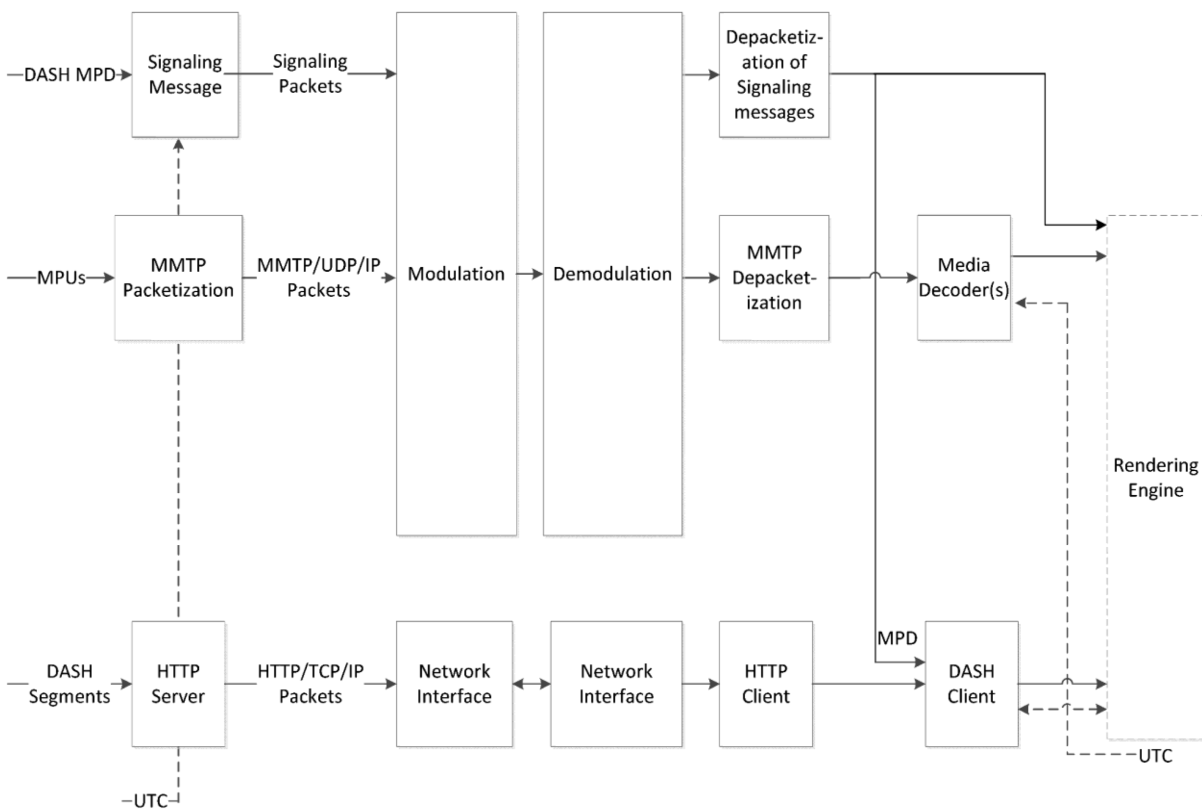

Figure 8.12 Conceptual hybrid Service architecture.

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 [38] 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 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.

8.2.2.6 Hand-off from Broadcast to Broadband Service Access

This operational scenario involves an ATSC 3.0 receiver, which due to user mobility or changing reception conditions, might temporarily cease to receive broadcast coverage and whereby a fallback to service reception via broadband is possible. Furthermore, these broadcast and unicast delivered Components can 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. In order to support this Component-level hand-off, **`BroadbandComponentInfo`** elements can further include the `@simulcastAssetId` attribute which represents the asset ID of an MPU Component that carries the same content as the DASH Representation delivered over broadband and is identified by the `@repId attribute` under the **`BroadbandComponentInfo`** element.

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 features 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 the first scenario. 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 a 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 is 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 is as follows:

1) The receiver determines that broadcast file delivery has ended;
2) The receiver identifies missing data from the broadcast reception as the corresponding byte range(s) in the delivery object;
3) The receiver 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) The 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 a 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 utilizes the APD fragment and the **EFDT** element in the S-TSID fragment to obtain 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 their locations in the delivery object. The sequence of events is as follows:

1) The receiver determines that broadcast file delivery has ended;
2) The receiver 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) The receiver 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) The 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 the **Alternate-Content-Location-1** and **Alternate-Content-Location-2** elements, 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 typically comprises two major elements:

- 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. Use of the repair protocol alone, without the source protocol, could be used for certain constrained or specialized deployment scenarios for which the broadcaster expects all targeted receivers to be AL-FEC capable. Such scenarios might include delivery of Services a) intended strictly for mobile reception, b) intended only in certain geographical areas, etc.

A.1.2 Source Protocol

The ROUTE source protocol is aligned with FLUTE as defined in RFC 6726 [31] as well as the extensions defined in MBMS [14], but also makes use of some principles of FCAST as defined in RFC 6968 [62]; 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 is 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 [60], but in contrast to the FEC Framework in RFC 6363 [60] 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 [60]. 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 [22], 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 [27]. 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, ROUTE supports the protection of multiple objects in one source block which is in alignment with the FEC Framework as defined in RFC 6363 [60].

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 session, LCT channel and delivery object description provided as Service signaling to enable the reception of objects.
- The normative aspects (formats, semantics) of the delivery objects conveyed as a content manifest to be delivered along with the objects to optimize the performance for specific applications; e.g., real-time delivery. The objects and manifest are made available to the application through a Delivery Object Cache.

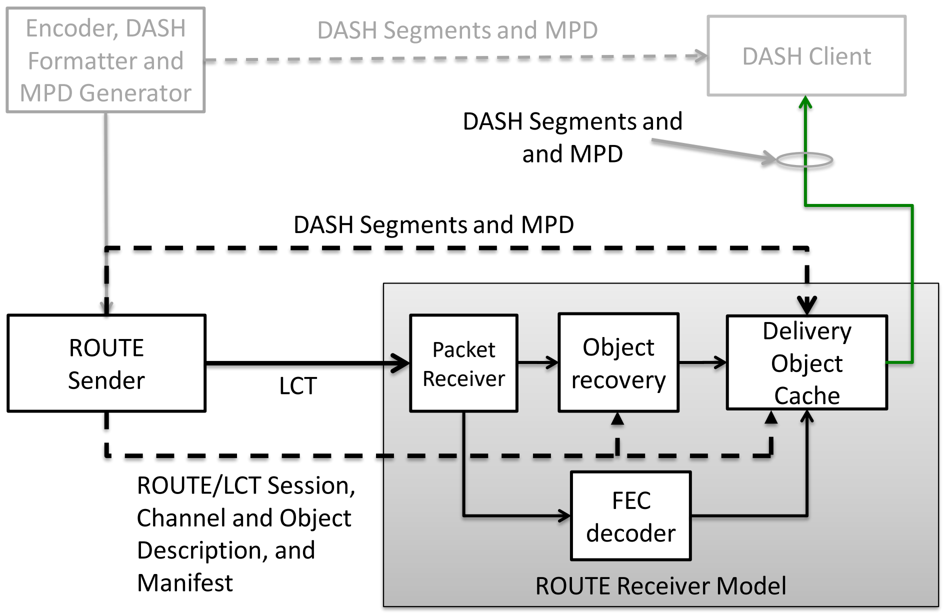

Figure A.1.1 Reference receiver architecture model in ROUTE.

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 ROUTE 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 flows, for example to differentiate the level of protection for the Core Layer and the Enhanced Layer defined in A/322 [4] when a scalable video codec is in use.

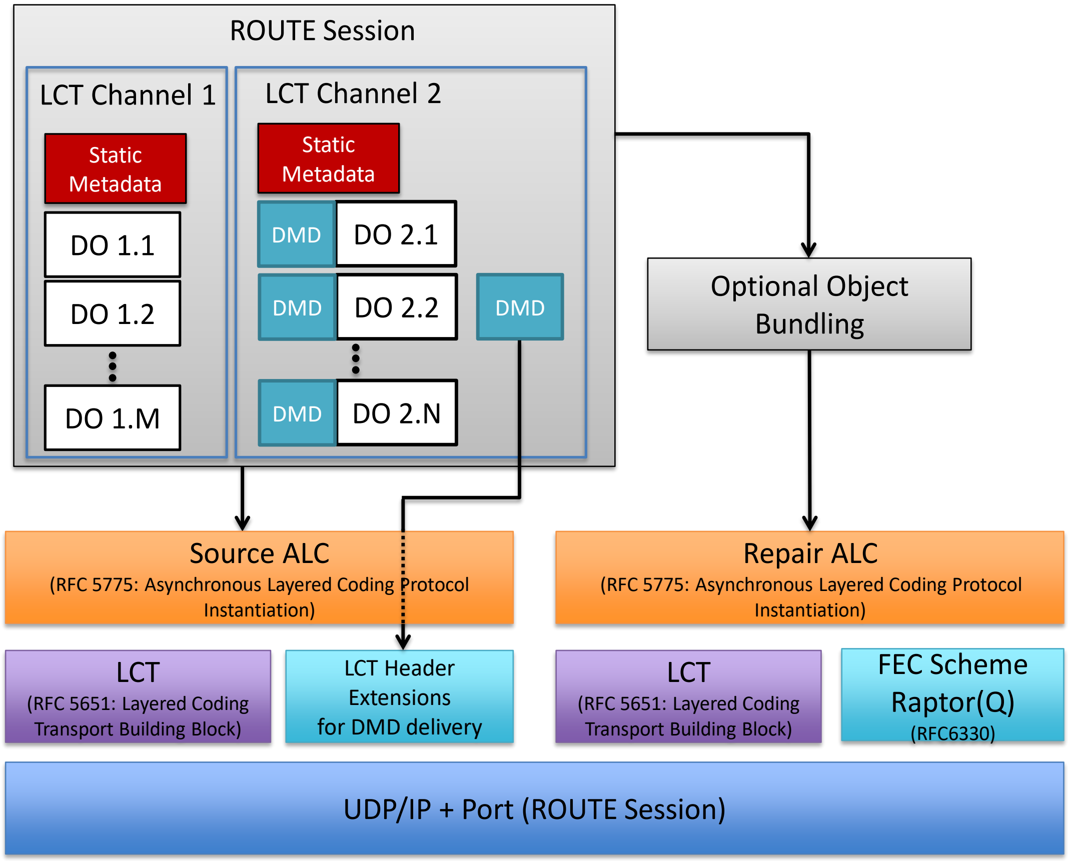

Figure A.1.2 Sender operation of ROUTE protocol.

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 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 Components of one or more DASH Media Presentations. Within each transport session (i.e., LCT channel), 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, HTML5 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 (i.e., LCT channel) 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 of the following:

- 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 element 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 [28] 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.3.9 and A.3.10, 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 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 [31] 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 (i.e., LCT channel).
<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		Deprecated. Use **MPD.Period.AdaptationSet.Rating** element. Content rating information associated with the program to which the media content of this source flow pertains.
@schemeIdUri	0..1	anyURI	Deprecated. Use **MPD.Period.AdaptationSet.Rating** element. Content advisory rating scheme associated with the program to which the media content of this source flow pertains.
@value	1	string	Deprecated. Use **MPD.Period.AdaptationSet.Rating** element. 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 payloads of ROUTE packets carrying the objects of the source flow are fragmented for delivery.
@order	0..1	boolean	Indication of how the payloads 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 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 when absent or set to "false", it shall be an indication of non-real-time content. The default value for this attribute is "false".

@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". When @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 [31] 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 Extended FDT Instance 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. When the **ContentInfo** element is absent, it means that no additional information is available on the 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. 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 is for backwards compatibility and should not be present in XML encodings that conform to this version of A/331. Decoders should ignore it when present.

@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 – Deprecated. **Use MPD.Period.AdaptationSet.Rating** element. This element shall provide content rating information associated with the content in the service and a MediaInfo. The content rating information is usually assigned to content during production and distribution. Content might have a rating, or it might not have a rating (and if it has a rating, that rating might be "None", "NR", or "E", see CTA-766 [11]). This element is required when …MediaInfo@contentType="video" except when the content in the service does not have a rating assigned, then it shall not be present. This element is authoritative over the Service Announcement (SA, ESG) rating. For North America, the term "not rated" shall be as defined in CTA 766 [11].

@schemeIdUri – Deprecated. Use **MPD.Period.AdaptationSet.Rating** element. An attribute which shall identify the content advisory rating scheme associated with the content in the service and MediaInfo. 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 – Deprecated. Use **MPD.Period.AdaptationSet.Rating** element. An attribute which shall identify the content advisory rating value. 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 element 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 [35]. 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 [35]. 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 an 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.3.3.2
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 Extended FDT Instance. 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 Extended FDT Instance.

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 [31], 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 [31], 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 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 only provides a reference model.

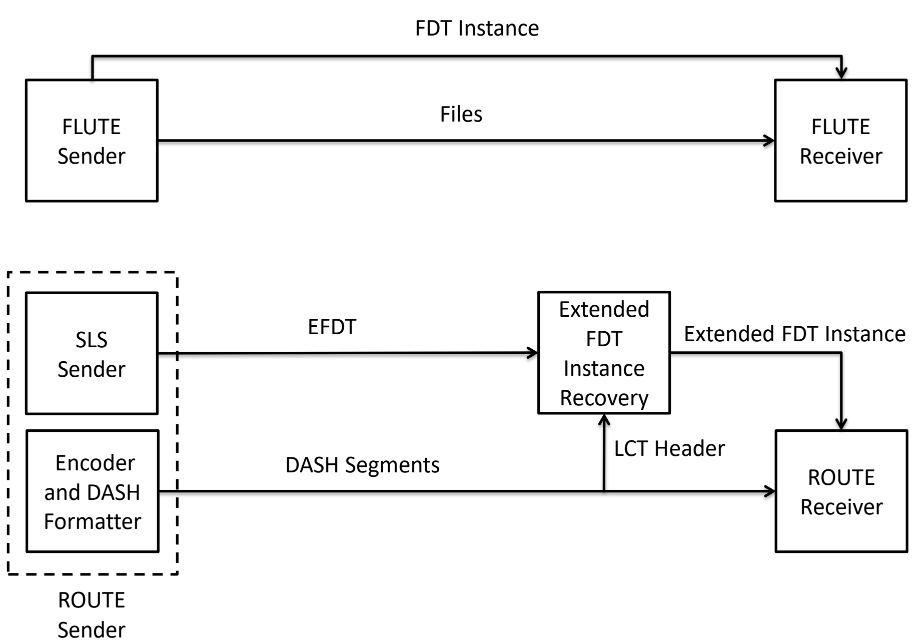

Figure A.3.1 ROUTE distribution in file mode compared to FLUTE distribution.

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 [31], 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 [31] 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-1.0-20230714.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 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 Extended FDT Instance.
@maxTransportSize	0..1	unsignedInt	The maximum transport size of any object described by this Extended FDT Instance.
@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 **File** element.
@filterCodes	0..1	afdt:listOfUnsignedInt	Filter Codes applying to all files in the Source flow.
@maxCacheMemory	0..1	unsignedint	The maximum memory required at any time by the receiver to cache the data when the file(s) are being delivered out-of-order.
@order	0..1	Boolean	Flag indicating out-of-order delivery of files if value is “false” or in-order delivery of files if value is “true”.

@efdtVersion – An 8-bit unsigned integer value that shall represent the version of this Extended FDT Instance. The version shall be increased by 1 modulo 256 each time the Extended FDT Instance is updated. @version is an ATSC-defined extension of the FLUTE FDT as specified in RFC 6726 [31].

@maxExpiresDelta – A 32-bit unsigned integer value, which when present, shall represent a time interval in number of seconds, which when added to the wall clock time at the receiver when the receiver acquires the first ROUTE packet carrying data of the object described by this Extended FDT Instance, shall represent the expiration time of the associated Extended FDT Instance. When @maxExpiresDelta is not present, the expiration time of the Extended FDT Instance shall be given by the sum of a) the value of the ERT field in the EXT_TIME LCT header extension in the first ROUTE packet carrying data of that file, and b) the current receiver time when parsing the packet header of that ROUTE packet. See Section A.3.3.2.7 on additional rules for deriving the Extended FDT Instance expiration time. @maxExpiresDelta is an ATSC-defined extension of the FLUTE FDT as specified in RFC 6726 [31].

@maxTransportSize – A 32-bit unsigned integer value that shall represent the maximum transport size in bytes of any delivery object described by this Extended FDT Instance. This attribute shall be present if a) the **FDT-Instance**@fileTemplate is present; or b) one or more **File** elements, if present in this Extended FDT Instance, 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 **FDT-Instance** element. 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 [54].

Any Application Context Identifiers included 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 [54].

`@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 Extended FDT Instance. The mechanism, as further described in Section A.3.3.2.7, shall involve substituting the TOI value of the delivery object, as contained in the ROUTE packet header, 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 **`FDT-Instance`** element 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 Extended FDT Instance 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 precedence.

`@maxCacheMemory` – When `FDT-Instance.order` flag is false, this optional 32-bit unsigned integer attribute represents the maximum memory required at any time to keep the received file data in cache. This allows the receiver to assess its capability to receive the data in non-persistent storage prior to block transfer of a completed block of contiguous data to persistent storage or for the receiver to repair a block of data in the case when Repair Flow is utilized. When not present the `FDT-Instance.maxTransportSize`, the LCT Header Transfer Length or the Entity Mode Content-Length should be used by the receiver. If `FDT-Instance.order` is true, this attribute has no meaning. When Repair Flows are in use, the value of `@maxCacheMemory` shall be the summation of the source flow symbol payload bytes across the maximum concurrently interleaved source blocks. See RFC 6726 [31] Section 3.4.2 for guidance when FDT-Instance element contains common attribute(s) with File element.

`@order` – This Boolean attribute shall indicate whether the files with File element(s) within FDT-Instance element are delivered out of the order of their generation. When the value is “false”, arbitrary delivery order is indicated. When the value is “true”, in-order delivery is indicated. When this attribute is absent, the default value shall be “true”. See RFC 6726 [31] Section 3.4.2 for guidance when FDT-Instance element contains common attribute(s) with File element.

A.3.3.2.3.1. ATSC Extensions to the **`FDT-Instance.File`** Element

Informative Table A.3.4 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 apply 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. When **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 – A 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 [31], with the following rules.

- At least one **File** element must be present, with the following additional constraints:
  - When exactly one **File** element is present in an Extended FDT Instance that is embedded in the S-TSID, and the Extended FDT Instance describes DASH Segments as the delivery objects carried by a source flow, that **File** element will strictly contain the metadata for the Initialization Segment. In other words, no **File** element instances are present for the purpose of describing Media Segments.
  - When more than one **File** element is present in an Extended FDT Instance that is embedded in the S-TSID, and the Extended FDT Instance describes NRT content files as the delivery objects carried by a source flow, each of those **File** elements will contain the metadata for an individual NRT file.
  - When more than one **File** element is present in an Extended FDT Instance that is transported as TOI=0 in the same LCT channel carrying the associated source flow, the delivery objects transported by the source flow are NRT content files whereby each of those **File** elements will contain the metadata for an individual NRT file.
- The @Expires attribute must be present.

When a @fileTemplate attribute is present, then the sender shall operate as follows:

- The TOI field in the ROUTE packet header shall be set such that Content-Location can be derived according to Section A.3.3.2.7.
- After sending the first packet with a given TOI value, none of the packets pertaining to this TOI shall be sent later than the wall clock time as derived from @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. When @maxExpiresDelta is not present, then the EXT_TIME header with Expected Residual Time (ERT) shall be used to derive the value of **FDT-Instance**@Expires, according to the procedure described below in Section A.3.3.2.7.

When 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

When 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 [28]) or the EXT_TOL header (per Section A.3.8.1), 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.
- When present, the @maxExpiresDelta shall be used to generate the value of the **FDT-Instance**@Expires attribute. The receiver is expected to add this value to its wall clock time when acquiring the first ROUTE packet carrying the data of a given delivery object to obtain the value for @Expires. When @maxExpiresDelta is not present, the EXT_TIME header with Expected Residual Time (ERT) shall be used to derive the expiry time of the Extended FDT Instance. When both @maxExpiresDelta and the ERT of EXT_TIME 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 @Expires. When neither @maxExpiresDelta nor the ERT field of the EXT_TIME header is present, then the expiration time of the Extended FDT Instance is given by its @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 decimal digits to be printed. If the value to be printed is shorter than this number, the result shall be padded with leading zeros. 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 "$". This is performed after replacing the TOI step below.	not applicable
TOI	This identifier is substituted with the TOI.	The format tag may be present. When 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 [32].

This mode reuses the major concepts of FCAST as defined in RFC 6968 [62], 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 that `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) [24], where objects are packaged into one document for transport and the package includes objects necessary for validation of the package. Signed Package delivery objects, when delivering ATSC 3.0 Application or Service Layer Signaling, shall be as described in ATSC A/360 [10], by "Application Code Signing", and "Signatures for Service Layer Signaling Carried over ROUTE/DASH", respectively. When binary files are included in the package, Content-Transfer-Encoding of "binary" should be used for those files. Note that 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 [28] with the following details:

- The Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [27] 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 [29] 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 [27] 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 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 [28], 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 [27]. 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 [27].

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`, as an octet number corresponding to the first octet of the fragment of the delivery object carried in this packet. 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 [29] 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 [27]. 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 [27], 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 ATSC Reserved.

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.

Note that CP values from 5 to 9 are all associated with transmitting a sequence of related delivery objects for a DASH-formatted media stream carried within a source flow. The use of CP values in this range can support MPD-less start-up as described in Section A.3.2.2, i.e., their presence can enable the DASH client to start playback prior to having acquired a complete MPD for the corresponding streaming Service.

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 [27], 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 [27]. 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 Extension Headers

The ROUTE receiver uses either the EXT_FTI per RFC 5775 [28] or the EXT_TOL header specified below to extract the transfer length of the object.

A.3.8.1. EXT_TOL Header

EXT_TOL is an LCT extension header that denotes Transport Object Length (TOL) of the delivery object, i.e., the modified length of the delivery object after any content encoding (via gzip [16]), as either a 24-bit or 48-bit unsigned integer value. The 24-bit and 48-bit formats of the EXT_TOL header are shown in Figure A.3.6 and Figure A.3.7, respectively. The 24-bit EXT_TOL is associated with the HET value of 194 and the 48-bit EXT_TOL is associated with the HET value of 67.

The ROUTE receiver uses either the EXT_FTI header per RFC 5775 [28] or the EXT_TOL header to extract the transfer length of the object. When EXT_FTI is not present, then either the 24-bit or 48-bit version of EXT_TOL should be present.

```
 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
```

Figure A.3.6 24-bit format of EXT_TOL Header.

```
 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.3.7 48-bit format of EXT_TOL Header.

A.3.9 Basic ROUTE Sender Operation

The following description of the ROUTE sender operation on the mapping of the delivery object to the ROUTE packet payloads logically represents an extension of RFC 5445 [26], which in turn inherits the context, language, declarations and restrictions of the FEC building block in RFC 5052 [22]. 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 [26], 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 ≥ 0 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 end of byte X + Y - 1.
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 2.

A.3.10 Basic ROUTE Receiver Operation

A.3.10.1. Overview

Figure A.3.8 illustrates the basic receiver operation. The receiver receives packets and filters those packets accordingly. From the ROUTE session and each contained LCT channel, the receiver 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.

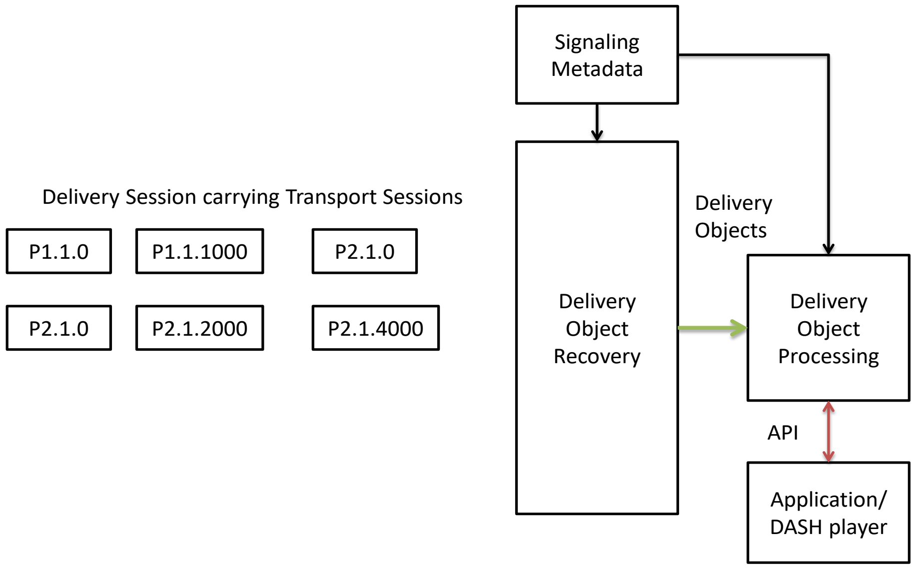

Figure A.3.8 Receiver operation.

A.3.10.2. 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, when present, to determine the length T of the object. When the **`File`**`@Transfer-Length` attribute is not present in the Extended FDT Instance, the receiver uses the `@maxTransportSize` attribute of the associated Extended FDT 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 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.3.10.3. 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. Regarding one defined mechanism for this, please refer to ETSI TS 126 247 [56], Clause 5.3.2 and Clause 7.3.9, as well as to ETSI TS 16 346 [14], Clause 7.9.

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 [32].

A.3.10.4. 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 [60], as well as the FEC building block, RFC 5052 [22], 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 elements 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 an N-symbol-sized chunk of data, where $N \geq 1$.
- 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 elements 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.

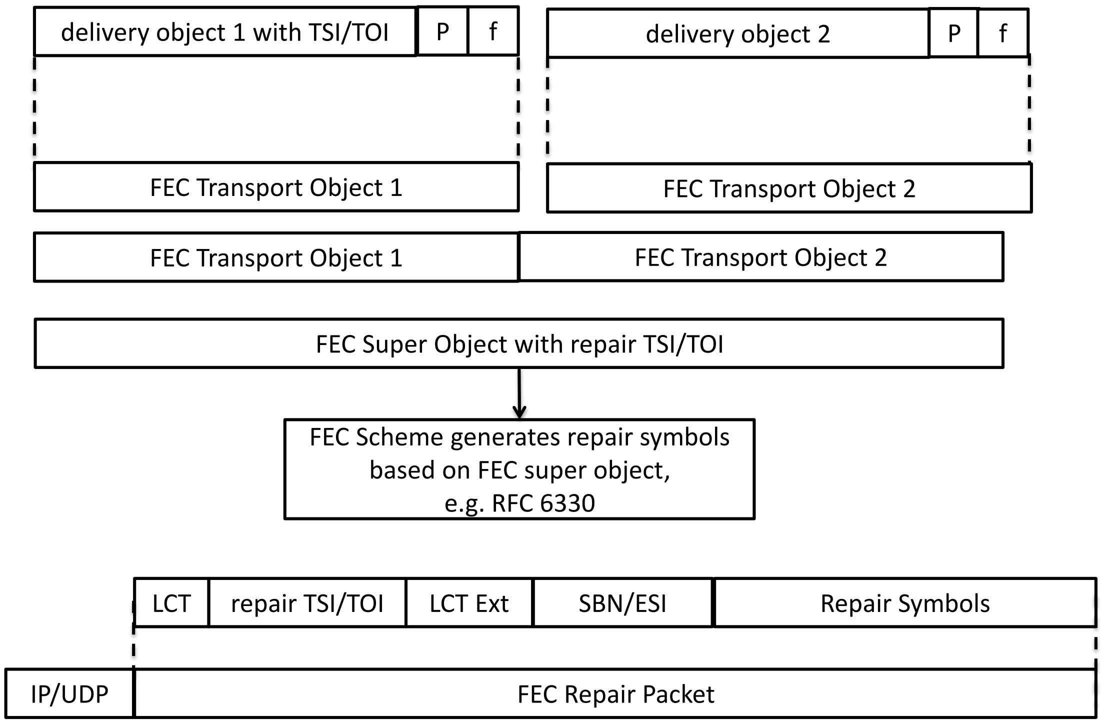

Figure A.4.1 FEC packet generation.

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** denote 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.

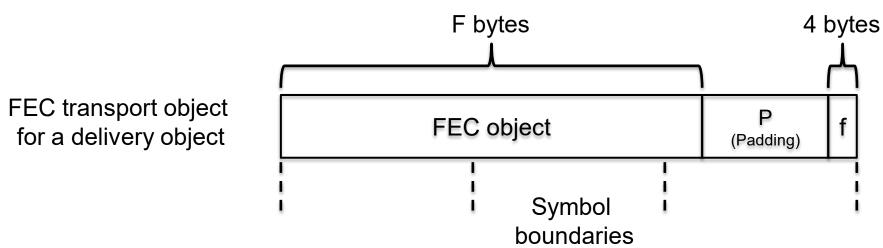

Figure A.4.2 FEC transport object construction (example with S = 3).

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, S

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 in RFC 5775 [28] and the Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [27] with the following details:

- The Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [27] 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 [29], 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 repair symbols for a corresponding FEC transport object/super-object as identified by its 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 [22].
  - Additional FEC information (see Table A.4.1).
- The total number of FEC objects included in the FEC super-object, N.
- 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, S.

The above information is delivered:

- Statically in the `RepairFlow` declaration as defined in Section A.4.3, and
- Dynamically in an LCT extension header

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 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.
@percentRepair	0..1	unsignedShort	The maximum ratio of repair symbols to source symbols as a percentage with values between zero and 200 inclusive.
@checksumList	0..1	listOfString	List of CRC32 hexadecimal formatted checksums for each source block in the order of the source blocks.
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, when 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 up to 1000%. The AL-FEC related parameters shall comprise the attributes `@maximumDelay`, `@overhead`, and `@minBufferSize`, and the `@fecOTI` attribute under the element **`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. When present, 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 [29], and which corresponds 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.

`@percentRepair` – An optional integer that equates to the maximum ratio of repair symbols to source symbols as a percentage with range of 0 to 200 inclusive. Absence of this attribute means the AL-FEC repair data size is not provided.

`@checksumList` – An optional space separated list of 32bit unsigned integers in hexadecimal format (e.g., values are 'hexadecimal' not '0xhexadecimal') that equates to the calculated CRC32 checksums of each source block. This list may be used by the receiver to determine the accuracy of received and/or repaired data in a source block. In its absence the integrity of the complete received and/or repaired file may be available through the optional FDT-Instance MD5 Checksum, however this CRC32 checksum list, when present, allows the receiver to more finely assess the need to replace data at a source block level of resolution.

`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 fully-qualified attributes, shall represent session description information for the source flow, if applicable, protected by the repair flow. The set of fully-qualified attributes 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). When `@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, when 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, when 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, when present, for use in deriving the parent **SourceTOI** value from the above equation. The default value shall be "0".

Note that when 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

When the repair flow declaration contains a **ProtectedObject** element, then the @sourceTOI element 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. When no **SourceTOI** element 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.4.3.

```
<?xml version="1.0"?>
<RepairFlow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/"
    xsi:schemaLocation="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/ S-TSID-1.0-
20210411.xsd">
    <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 x="1" y="0"/>
        </ProtectedObject>
    </FECParameters>
</RepairFlow>
```

Figure A.4.3 `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.4.4.) 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.

```xml
<?xml version="1.0"?>
<RepairFlow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/"
    xsi:schemaLocation="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/ S-TSID-1.0-
20210411.xsd">
    <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">
                <SourceTOI x="1" y="0"/>
        </ProtectedObject>
        <ProtectedObject
                sessionDescription="sIpaddr=199.201.100.16,dIpaddr=226.17.97.55,dport=1234,

    bw=128000,startTime=32159980800,endTime=32162572800,srcFecPayloadId=39"
                tsi="3">
                <SourceTOI x="1" y="0"/>
        </ProtectedObject>
    </FECParameters>
</RepairFlow>
```

Figure A.4.4 `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.4.5.) 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.

```
<?xml version="1.0"?>
<RepairFlow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/"
    xsi:schemaLocation="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/ S-TSID-1.0-
20210411.xsd">
    <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 x="2" y="0"/>
        </ProtectedObject>
        <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 x="2" y="1"/>
        </ProtectedObject>
        <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 x="2" y="0"/>
        </ProtectedObject>
    </FECParameters>
</RepairFlow>
```

Figure A.4.5 `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.4.6.)

```xml
<?xml version="1.0"?>
<RepairFlow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/"
    xsi:schemaLocation="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/ S-TSID-1.0-
20210411.xsd">
    <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 x="1" y="1"/>
        </ProtectedObject>
    </FECParameters>
</RepairFlow>
```

Figure A.4.6 `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.4.7.) The following repair flow declaration with TSI 14 provides the static FDD.

```xml
<?xml version="1.0"?>
<RepairFlow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/"
    xsi:schemaLocation="tag:atsc.org,2016:XMLSchemas/ATSC3/Delivery/S-TSID/1.0/ S-TSID-1.0-
20210411.xsd">
    <FECParameters maximumDelay="5000" overhead="50" minBuffSize="20000000"
        fecOTI="f0f1f2f3f4f5f6f7f8f9fafb">
        <ProtectedObject sessionDescription="http://example.com/session7.xml" tsi="7">
                <SourceTOI x="2" y="0"/>
        </ProtectedObject>
        <ProtectedObject sessionDescription="http://example.com/session7.xml" tsi="7">
                <SourceTOI x="2" y="1"/>
        </ProtectedObject>
    </FECParameters>
</RepairFlow>
```

Figure A.4.7 `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 a 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 the receiver 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 the receiver must receive data on all source flows that are protected by that repair flow to collect the relevant packets.

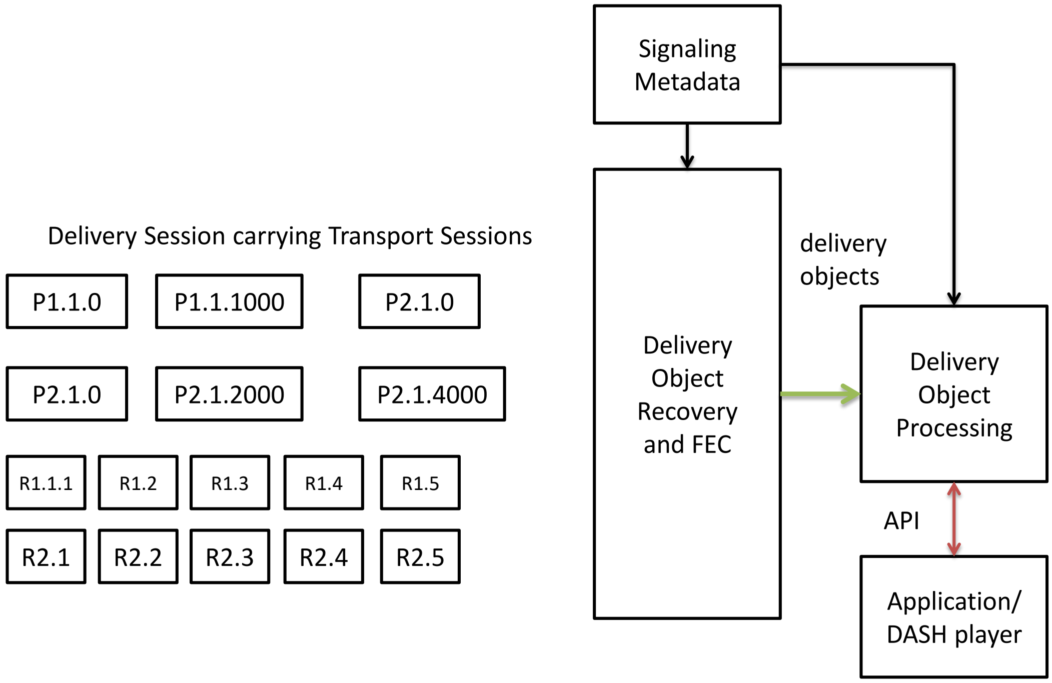

Figure A.4.8 ROUTE receiver with FEC.

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, the receiver is expected to collect all packets of all protected Transport Sessions.

Upon receipt of each packet, whether it is 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.3.10.2. 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

When 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 [28].

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 Components of Service_X. The second S-TSID provides access information for the LCT channel(s) belonging to ROUTE session #N, which carries the 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).

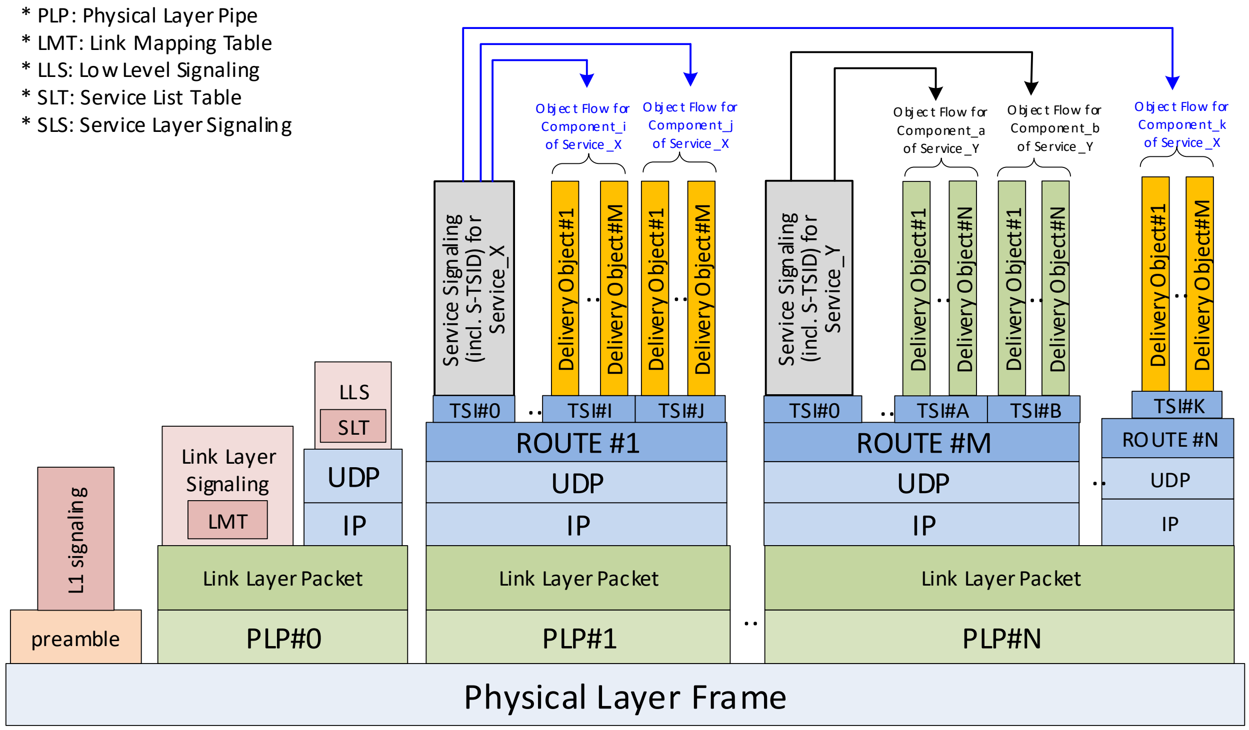

Figure B.1.1 ATSC 3.0 ROUTE hierarchical signaling architecture.

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 Components of Service_X. The second MP table provides access information for the MMTP packet flows belonging to MMTP session #N, which carries the Components of Service_Y.

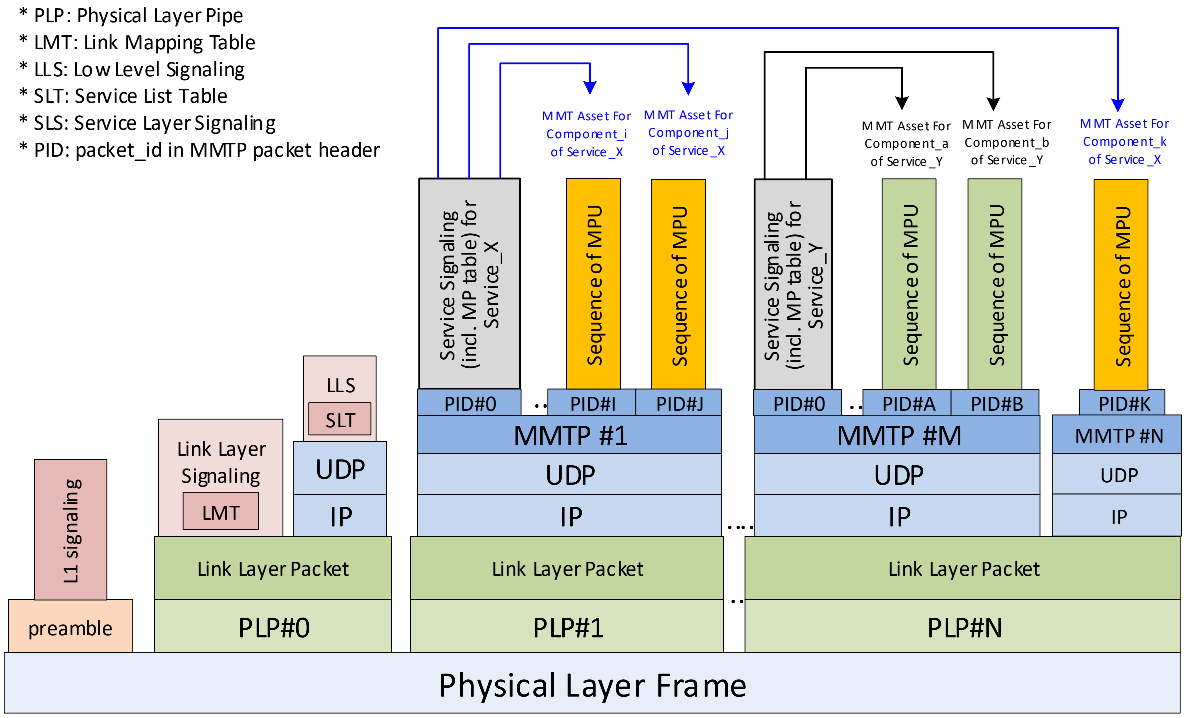

Figure B.1.2 ATSC 3.0 MMTP hierarchical signaling architecture.

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.

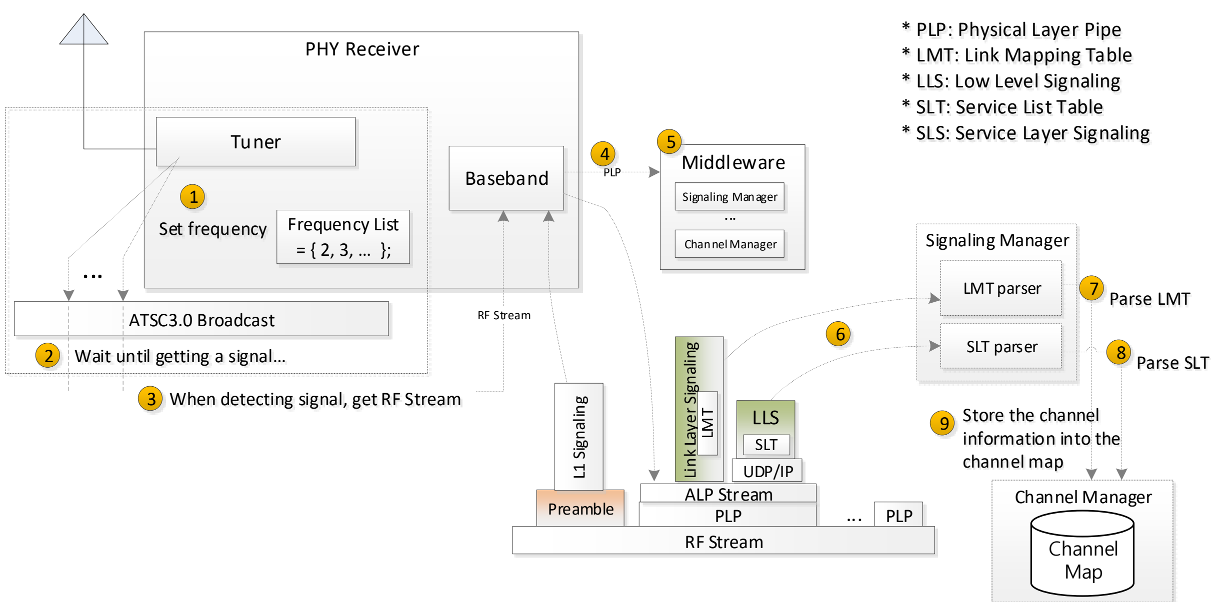

Figure B.2.1 Fast Service scan signaling flow.

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 feature) 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.

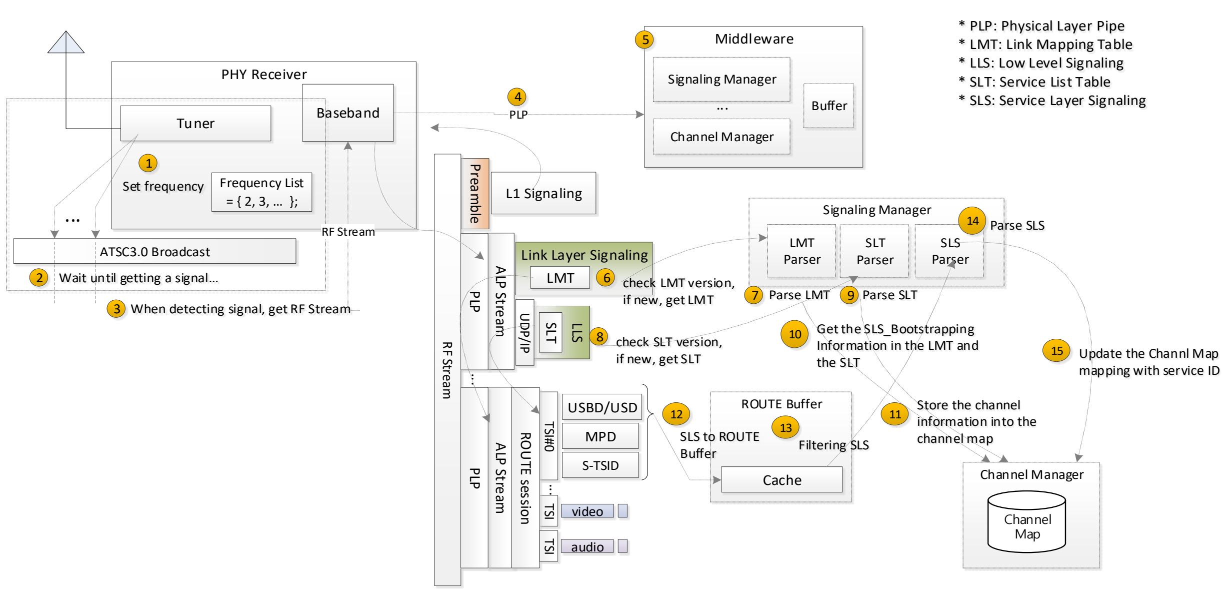

Figure B.3.1 ROUTE full-Service scan signaling flow.

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 the signaling to the appropriate functional module that manages signaling data (among the separate functional modules which manage signaling, audio/video/cc or app-based feature) and controls the delivery of that data to the appropriate parser and internal cache.
6) Receivers should check whether or not the LMT version is new. If so, the middleware module can gather the LMT and pass it to the LMT parser.
7) LMT parser will parse the data, and extract the information.
8) Receivers should check whether or not the LLS_table_version of the LLS_table() containing the SLT is new. It is best to parse the SLT 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) SLS Bootstrapping information will be stored into the channel map.
12) The SLS Bootstrapping information is passed by the receiver to an internal buffer.
13) Receiver uses the signaling filtering scheme to extract the USBD fragment from the SLS and stores it.
14) USBD fragment is parsed by the signaling parser. (It is best to parse the USBD 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.

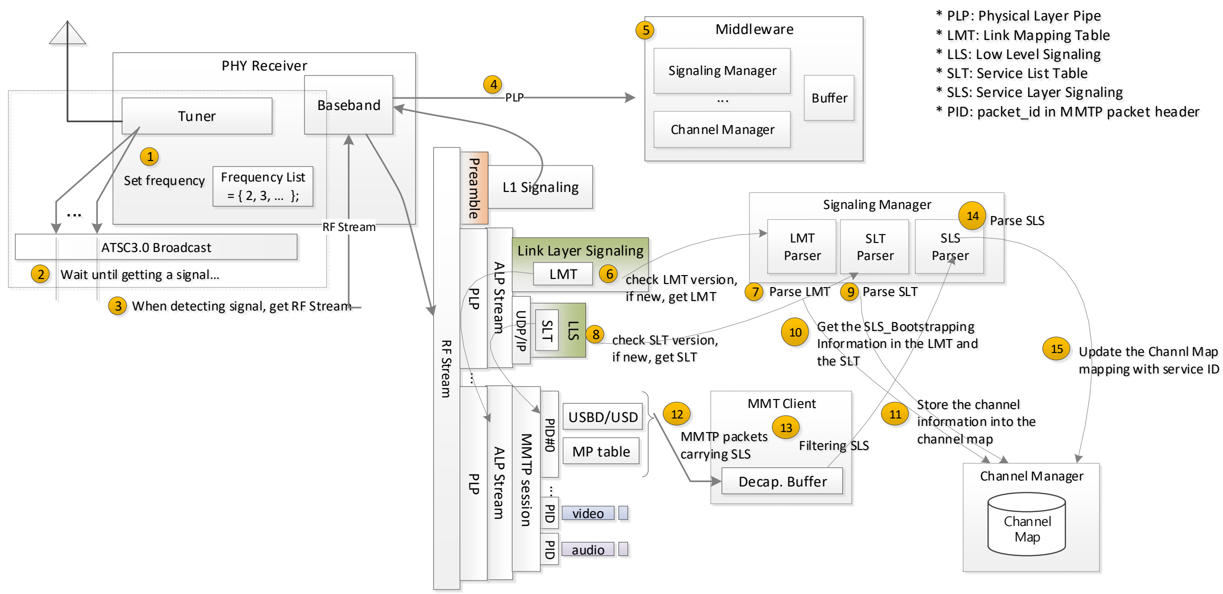

Figure B.3.2 MMTP full-Service scan signaling flow.

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 the signaling to the appropriate functional module that manages signaling data (among the separate functional modules which manage signaling, audio/video/cc or app-based feature) and controls the delivery of that data to the appropriate parser and internal cache.
6) Receivers should check whether or not the LMT version is new. If so, the middleware module can gather the LMT and pass it to the LMT parser.
7) LMT parser will parse the data, and extract the information.
8) Receivers should check whether or not the LLS_table_version of the LLS_table() containing the SLT is new. It is best to parse the SLT 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) SLT Bootstrapping 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 those packets to the MMTP Client.
13) MMTP Client extracts the SLS including the USBD fragment and stores it.

14) USBD fragment is parsed by the signaling parser. (It is best to parse the USBD 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 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.

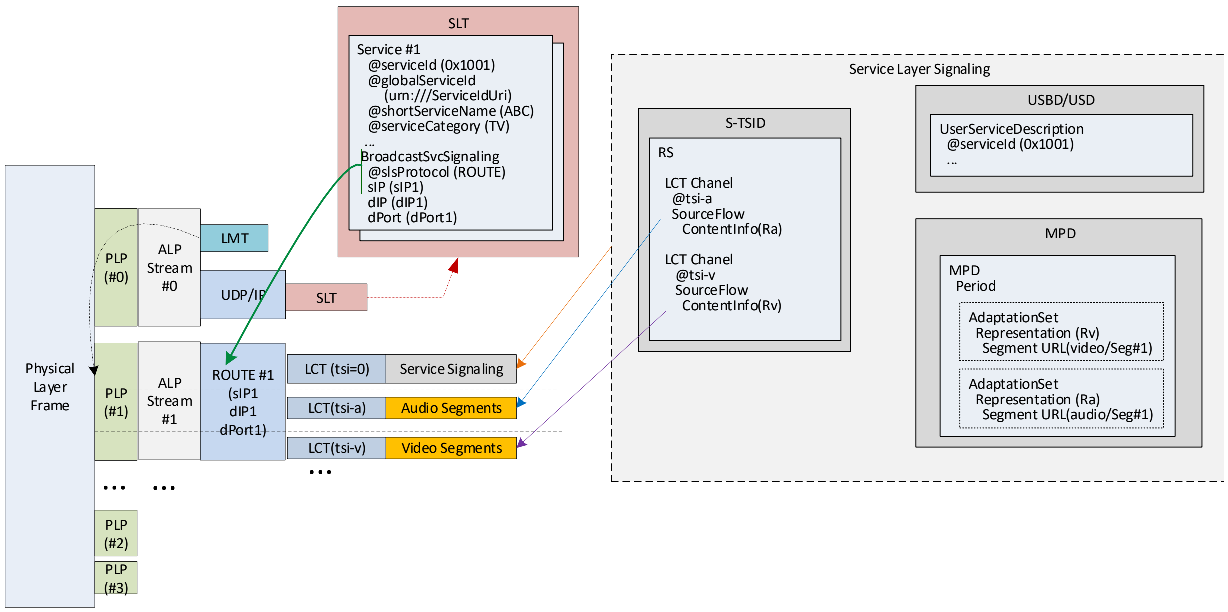

Figure B.4.1 Service acquisition in pure broadcast (one ROUTE session).

1) USBD, 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 (USBD, 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 USBD.

When video and audio MPUs are delivered via pure broadcast with one MMTP session, the Service signaling will be structured like below.

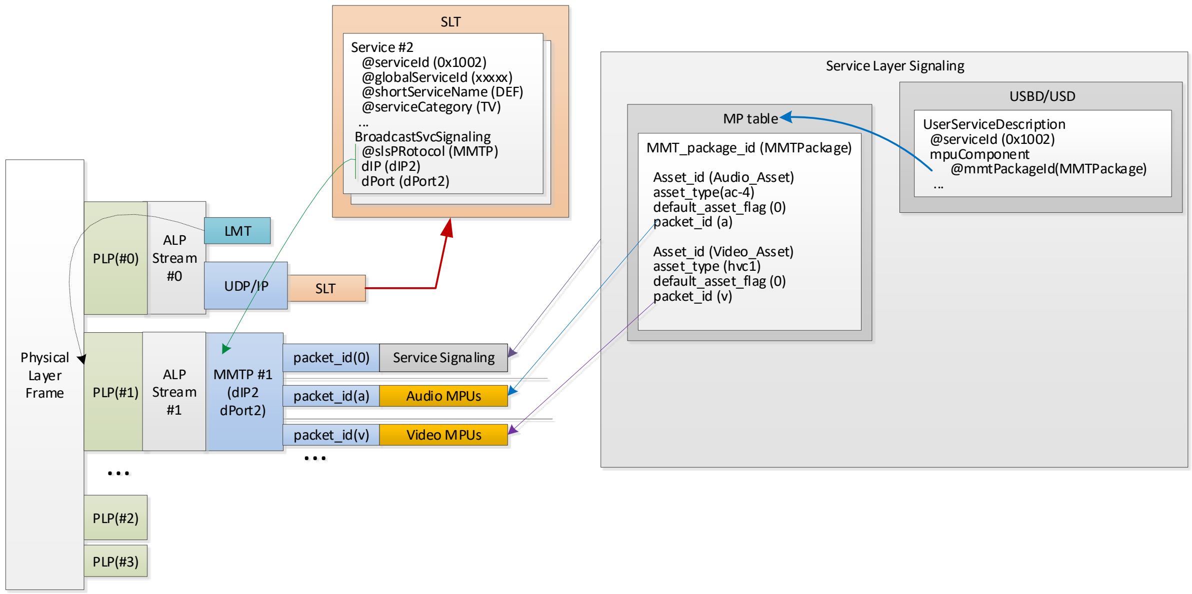

Figure B.4.2 Service acquisition in the pure broadcast (one MMTP session).

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 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.

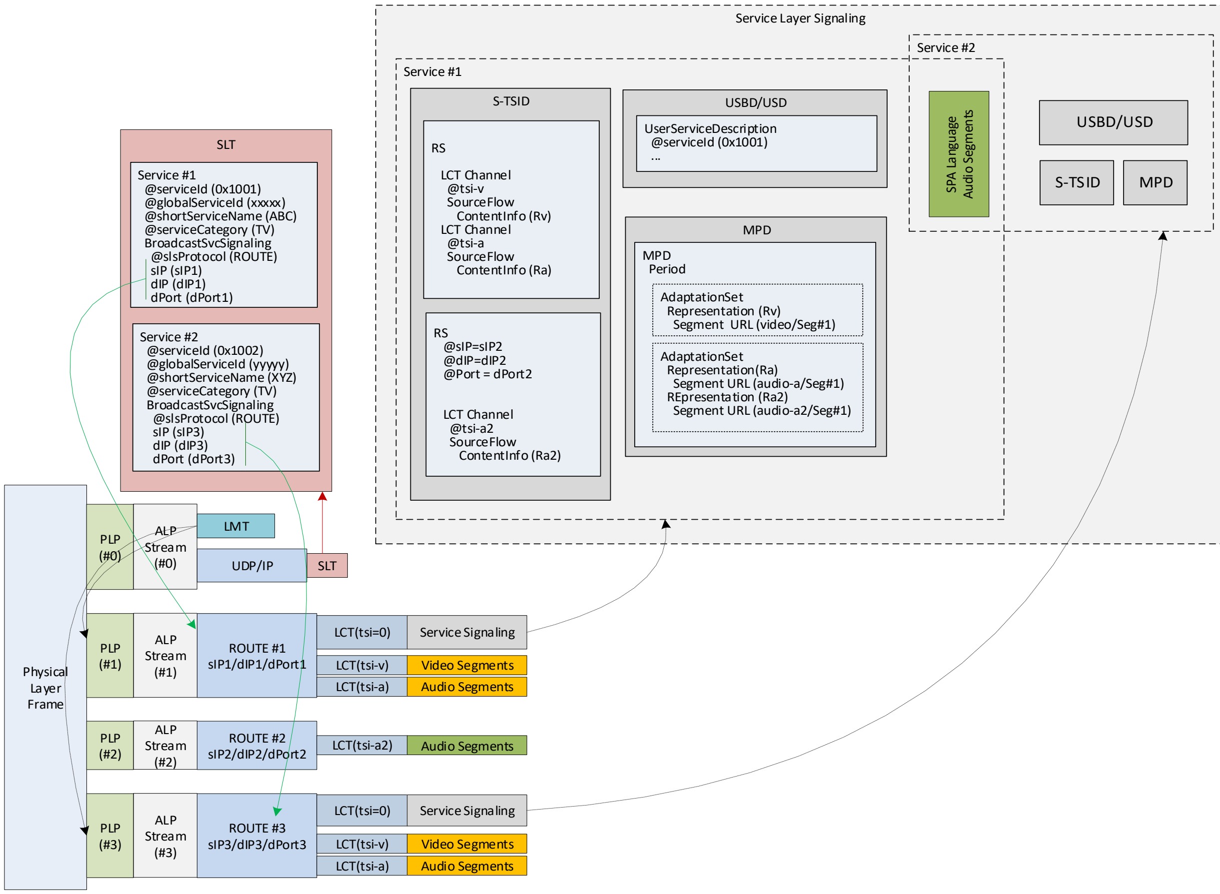

Figure B.5.1 Service acquisition in pure broadcast (multiple ROUTE sessions).

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.

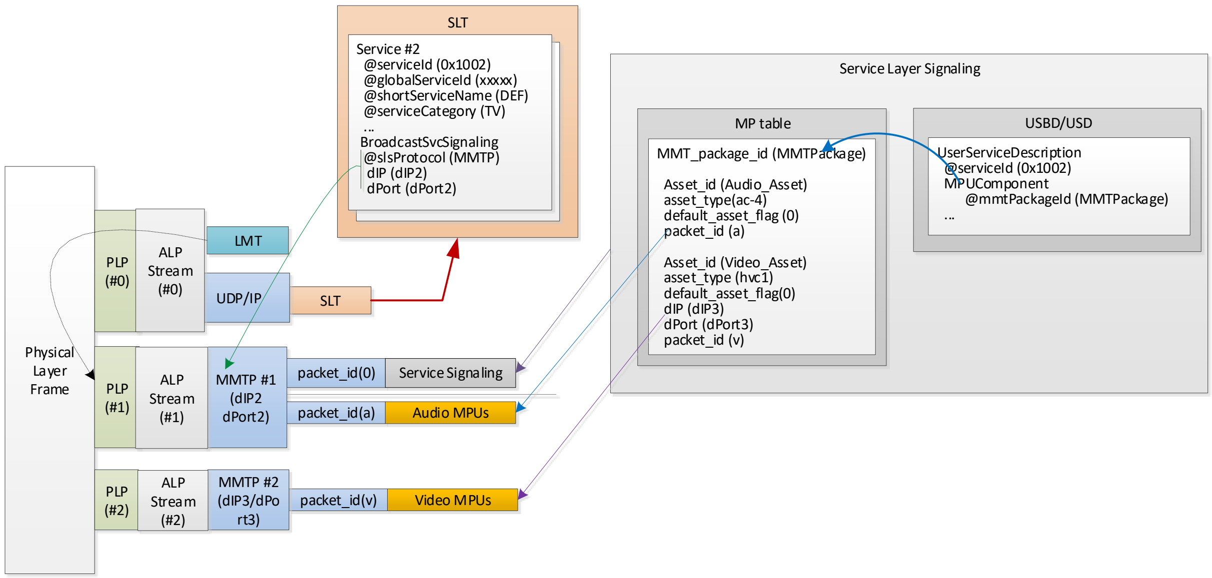

Figure B.5.2 Service acquisition in pure broadcast (multiple MMTP sessions).

Furthermore, one Service can be composed of one MMTP session carrying 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.

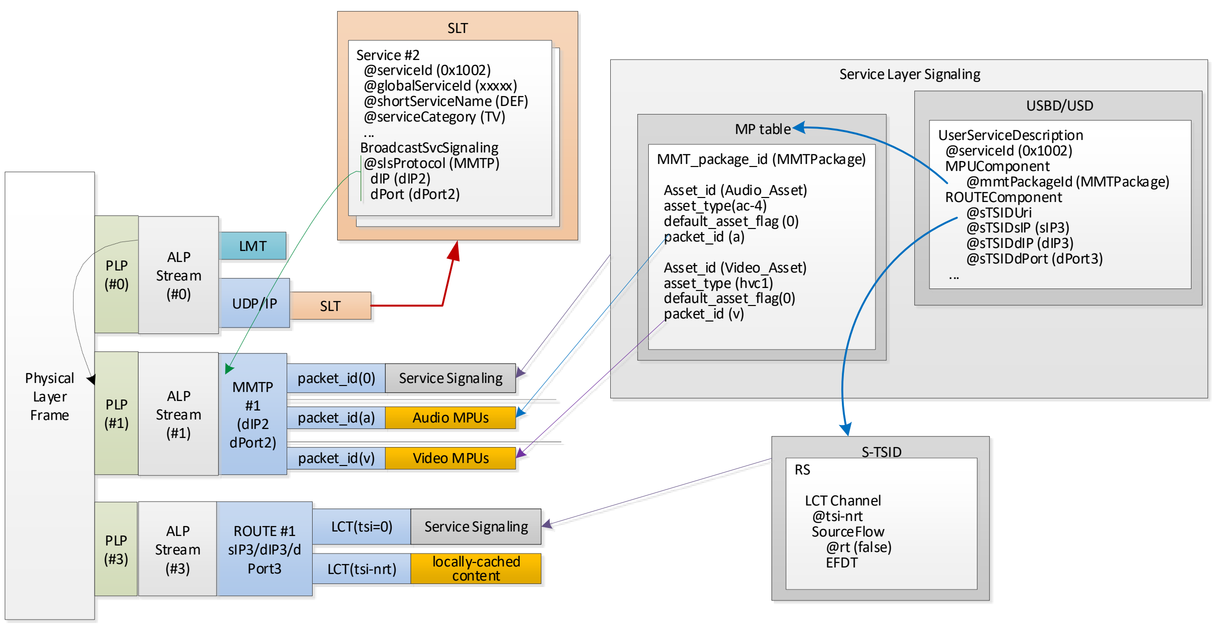

Figure B.5.3 Service acquisition in pure broadcast (one MMTP session and one ROUTE session).

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.

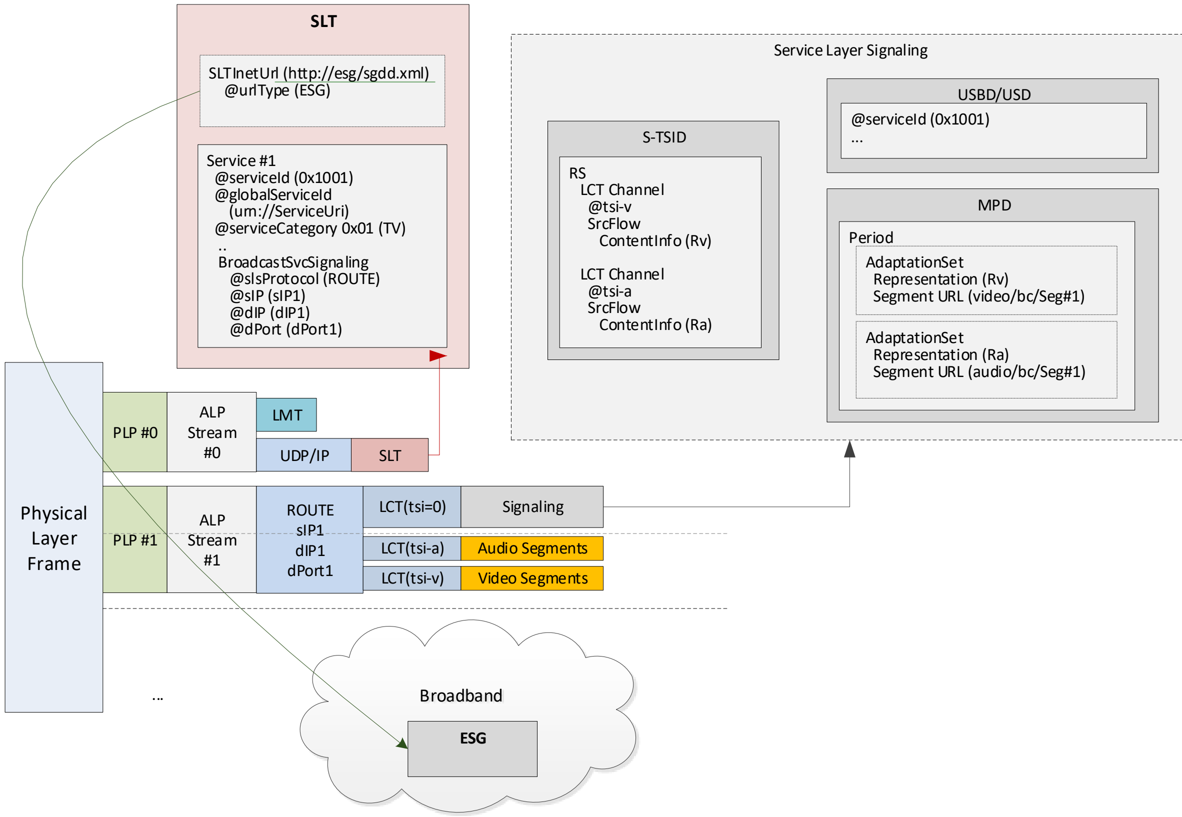

Figure B.6.1 ROUTE ESG bootstrapping via broadband.

B.7 HYBRID DELIVERY (MULTIPLE AUDIO LANGUAGES)

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.

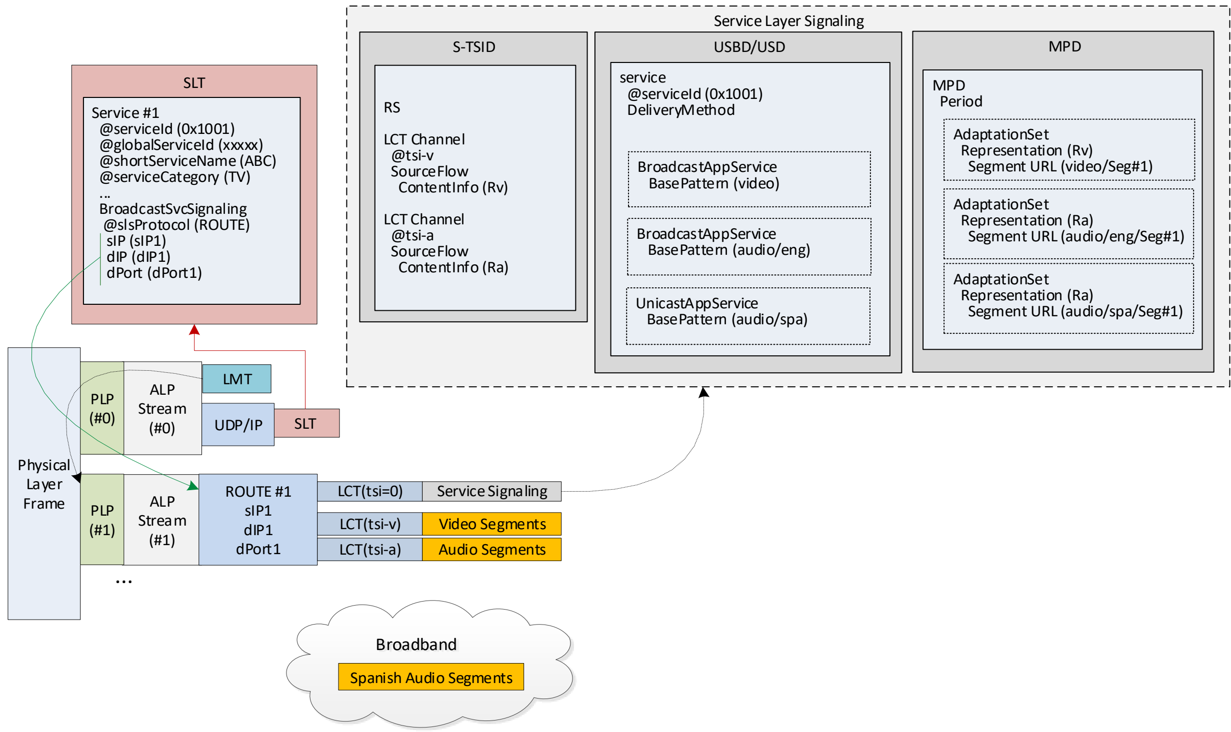

Figure B.7.1 ROUTE hybrid (multiple audio languages).

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.

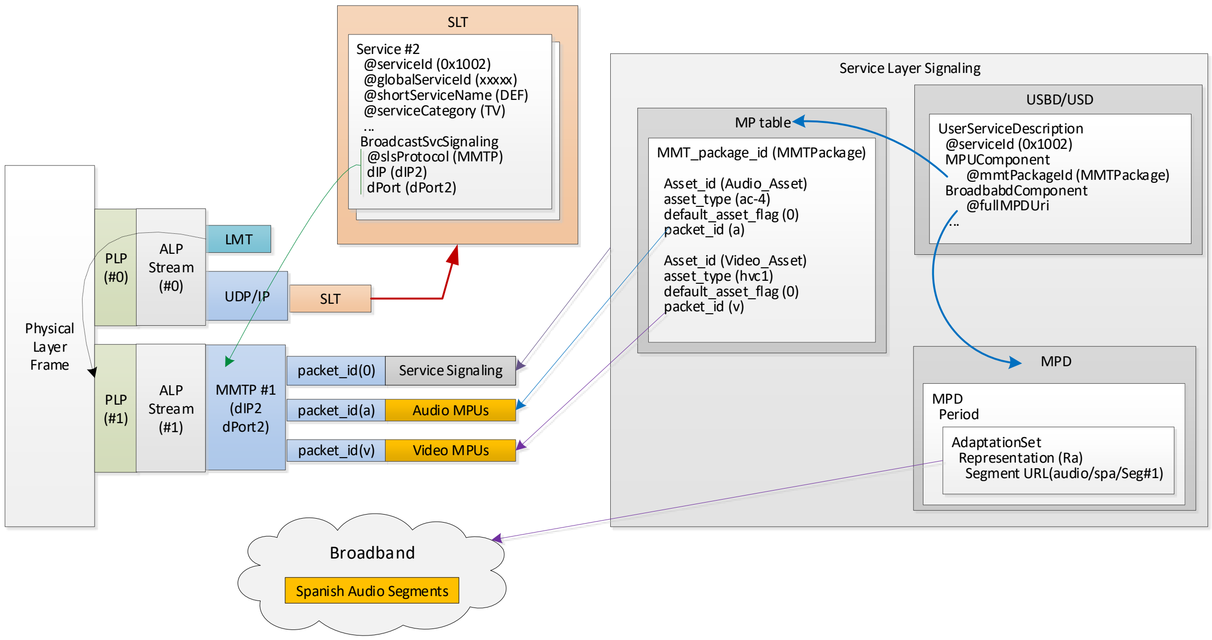

Figure B.7.2 MMTP hybrid (multiple audio languages).

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.

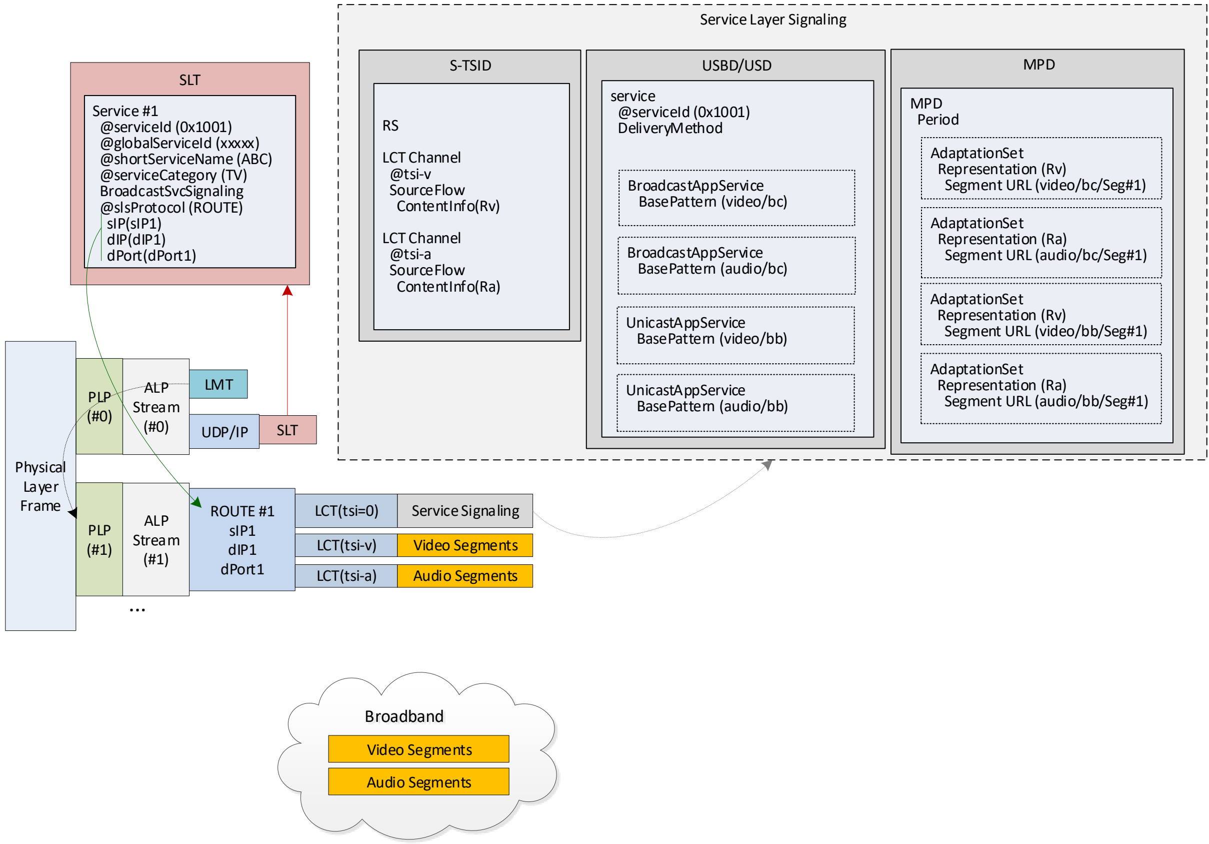

Figure B.8.1 Handoff (broadcast to broadband, and back).

B.9 SCALABLE CODING (CAPABILITY IN THE SLT)

Receivers can do scalable coding using one Service delivery by ROUTE.

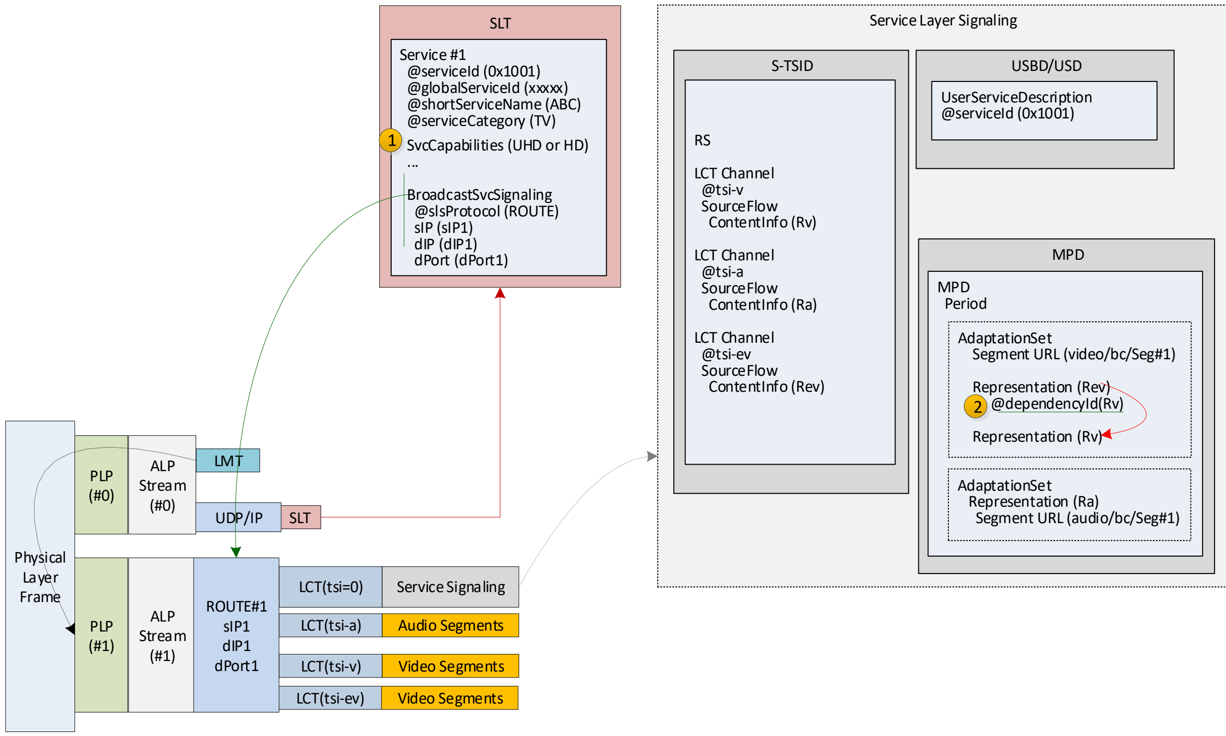

Figure B.9.1 Scalable coding for a linear TV Service delivered by ROUTE (capability in SLT).

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 ‘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.

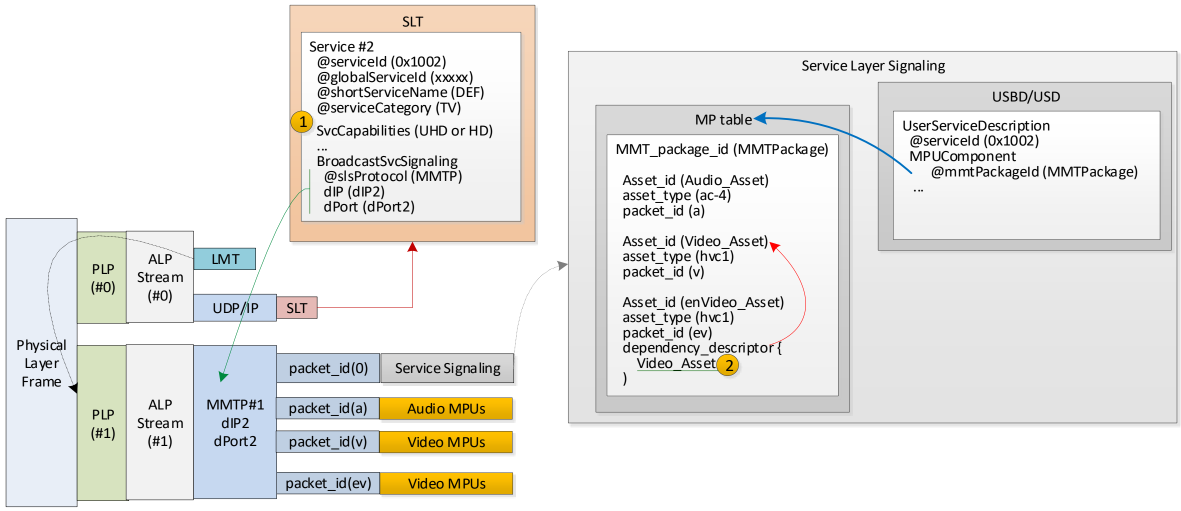

Figure B.9.2 Scalable coding for a linear TV Service delivered by MMTP (capability in SLT).

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 '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.

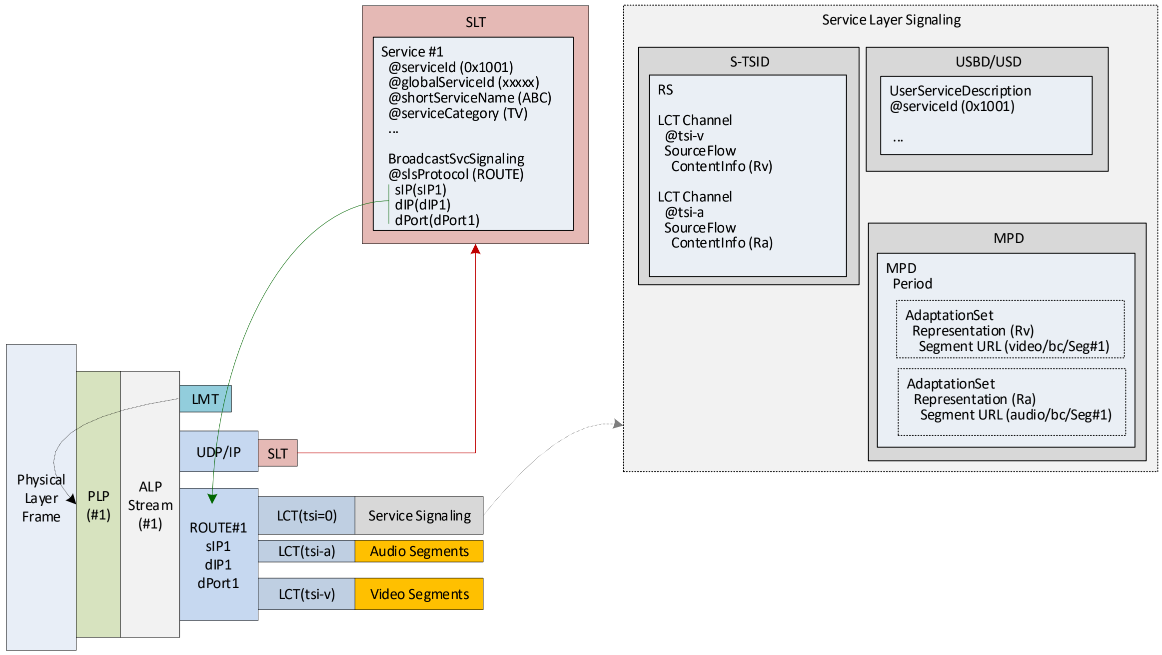

Figure B.10.1 ROUTE SLT delivery path.

B.11 MULTIPLE SLTs IN A BROADCAST STREAM

Multiple SLTs can be present in one broadcast stream.

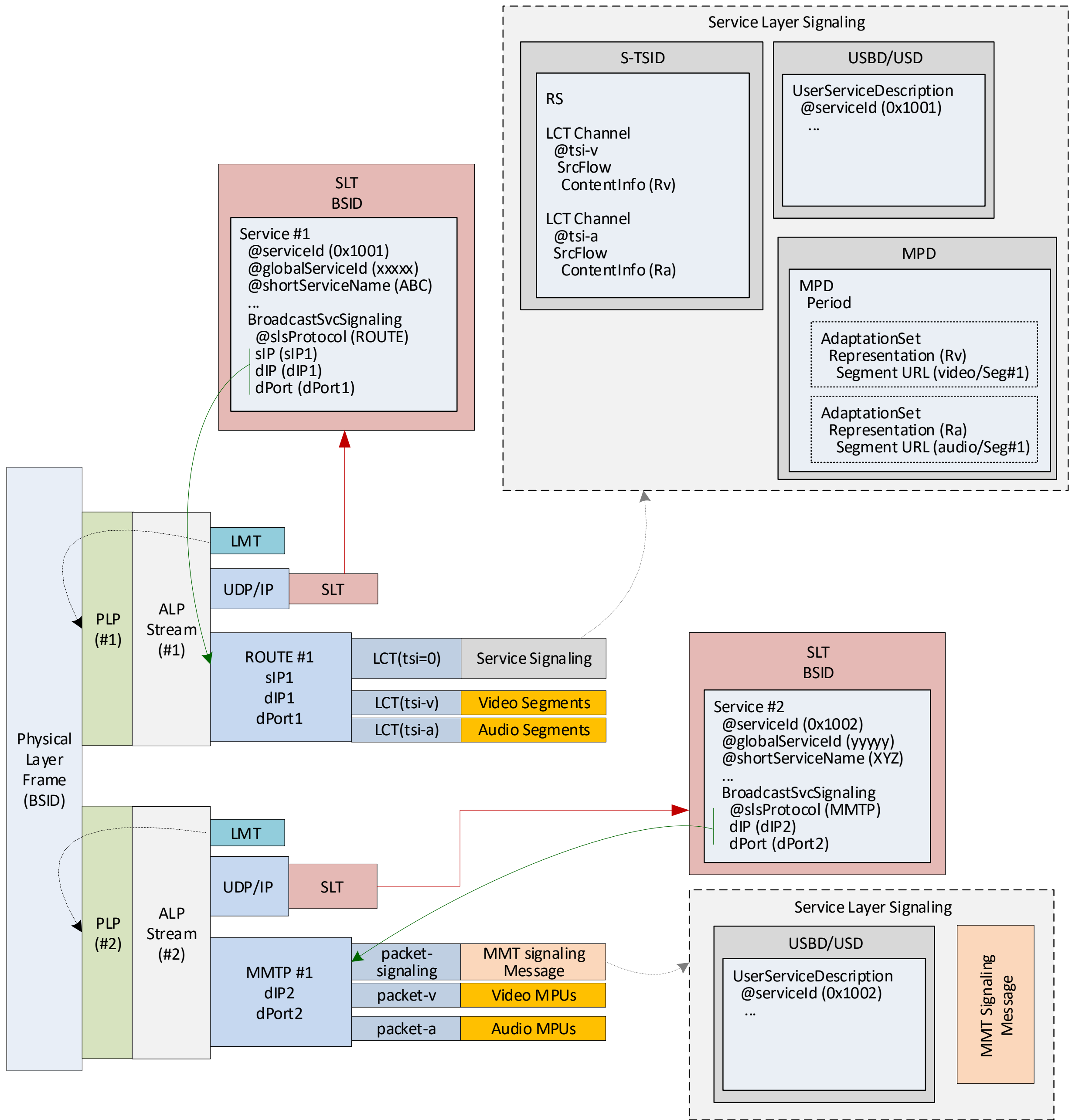

Figure B.11.1 Multiple SLTs.

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.

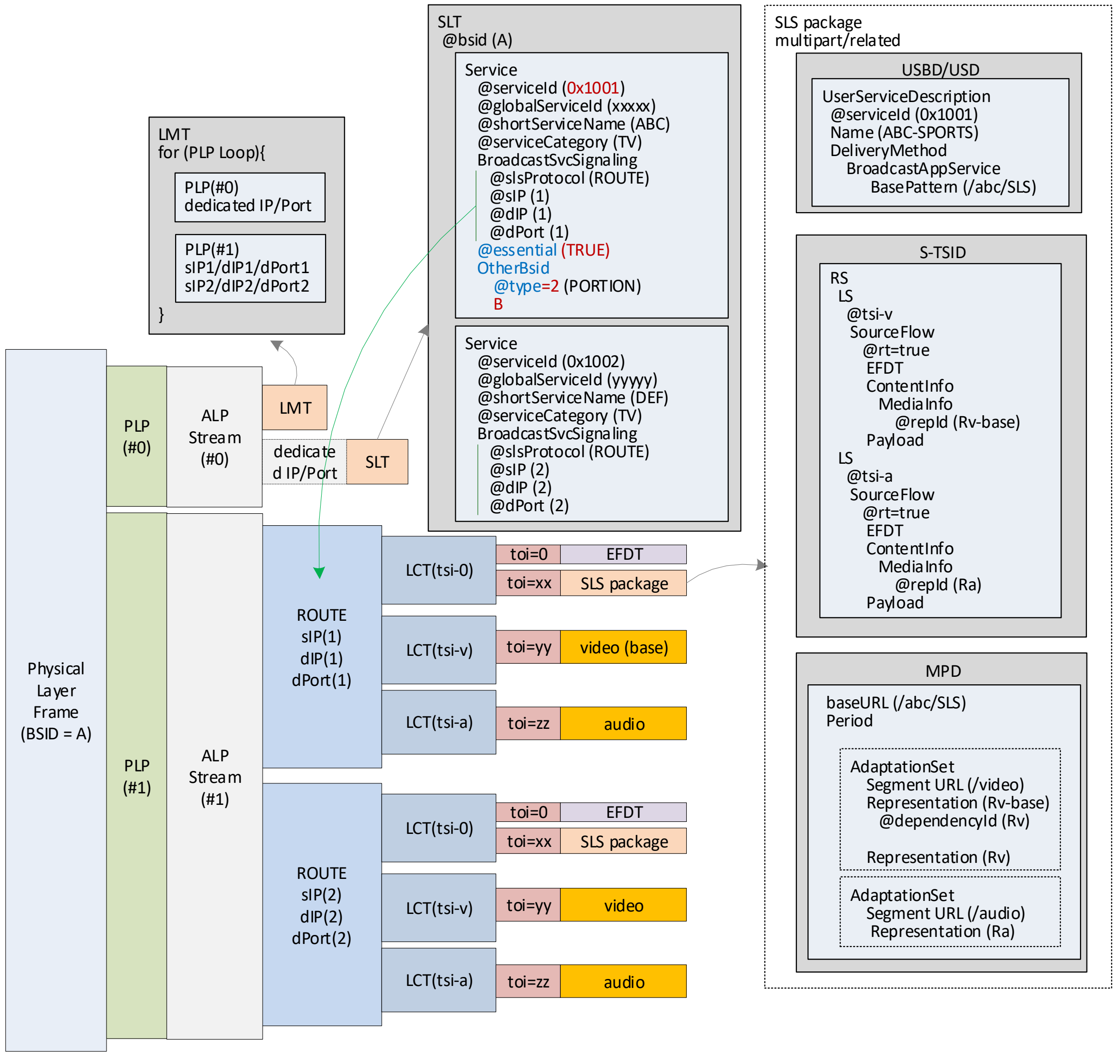

Figure B.12.1 Portions of a Service delivered in multiple RF channels without channel bonding (first RF channel).

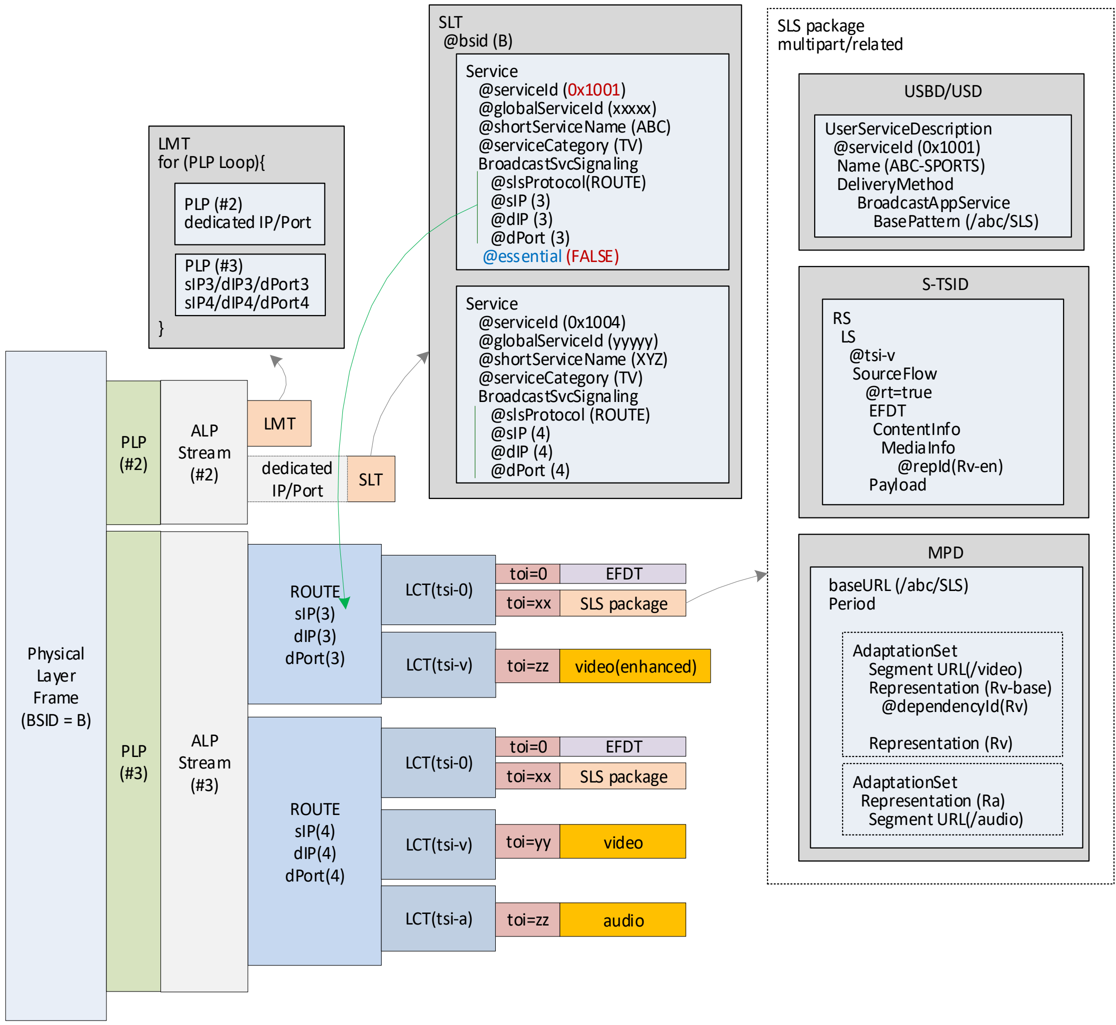

Figure B.12.2 Portions of a Service delivered in multiple RF channels without channel bonding (second RF channel).

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.

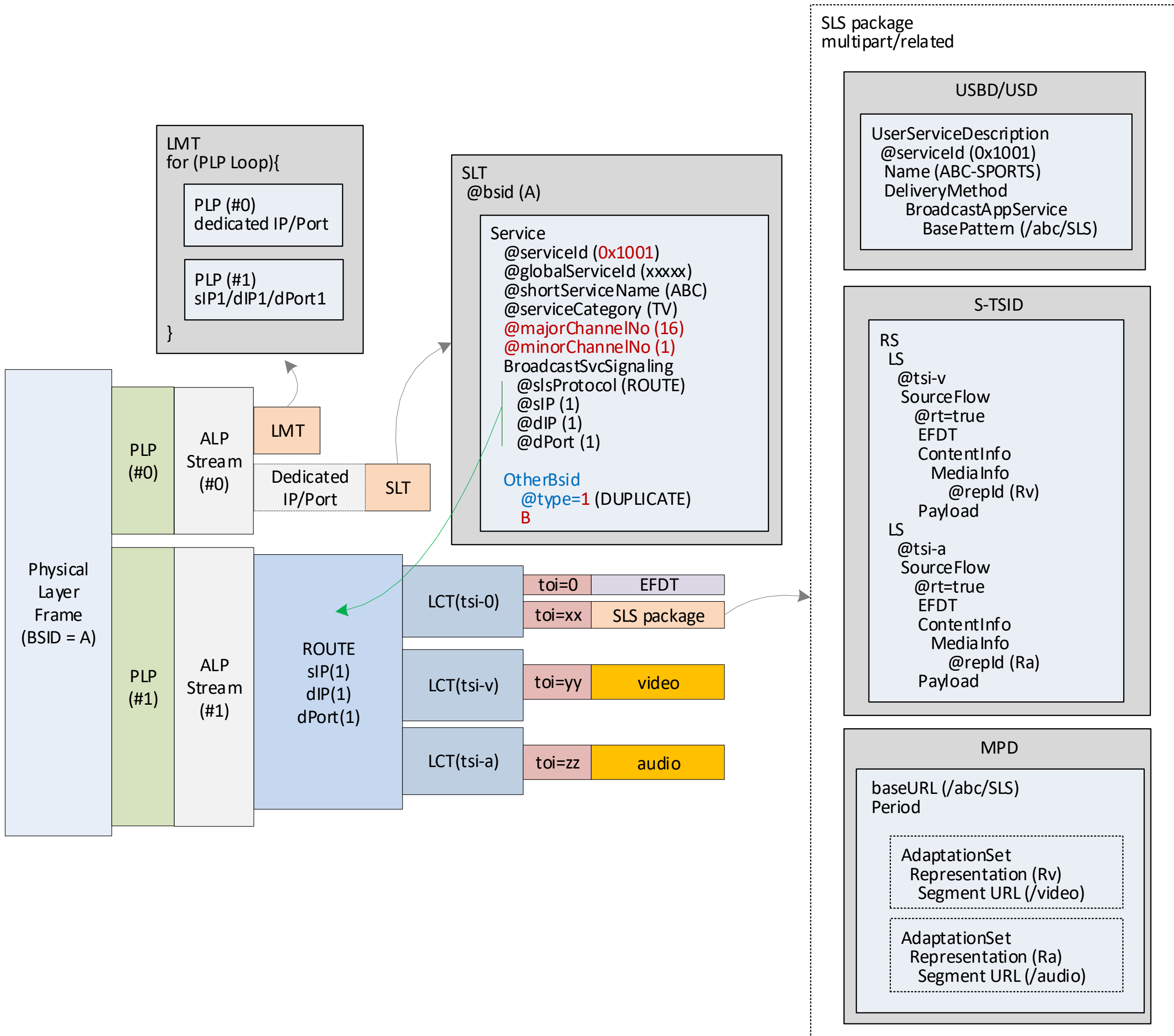

Figure B.13.1 Duplicates of a Service delivered in multiple RF channels without channel bonding (first RF channel).

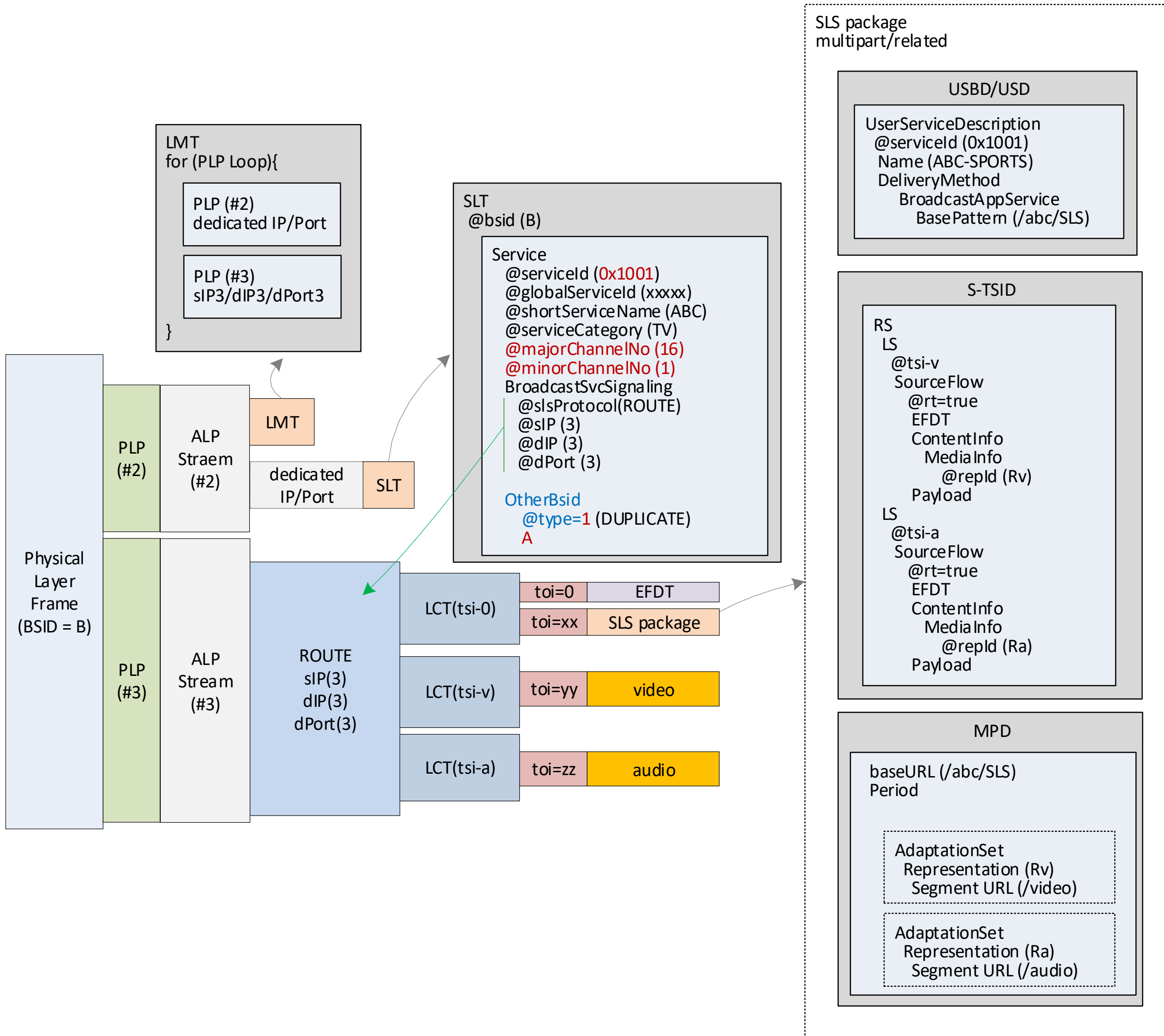

Figure B.13.2 Duplicates of a Service delivered in multiple RF channels without channel bonding (second RF channel).

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", where A and B represent a pair of unique 16-bit unsigned integer values, and it is referenced by the **`SLT.OtherBsid`** element associated with essential portion of the Service.

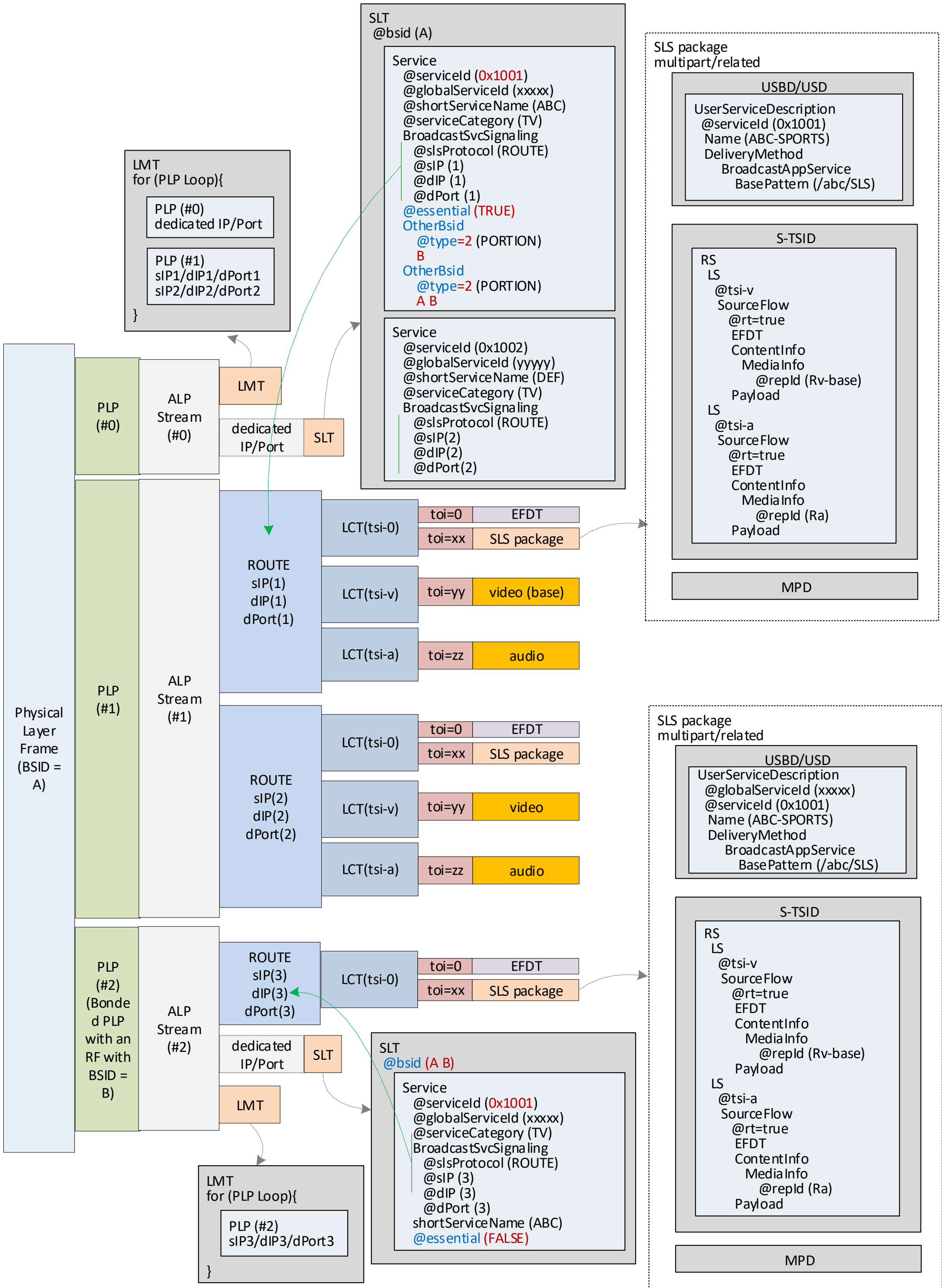

Figure B.14.1 Portions of a Service delivered in multiple RF channels with channel bonding – Case #1: essential portion is delivered in non-bonded PLP(s) (first RF channel).

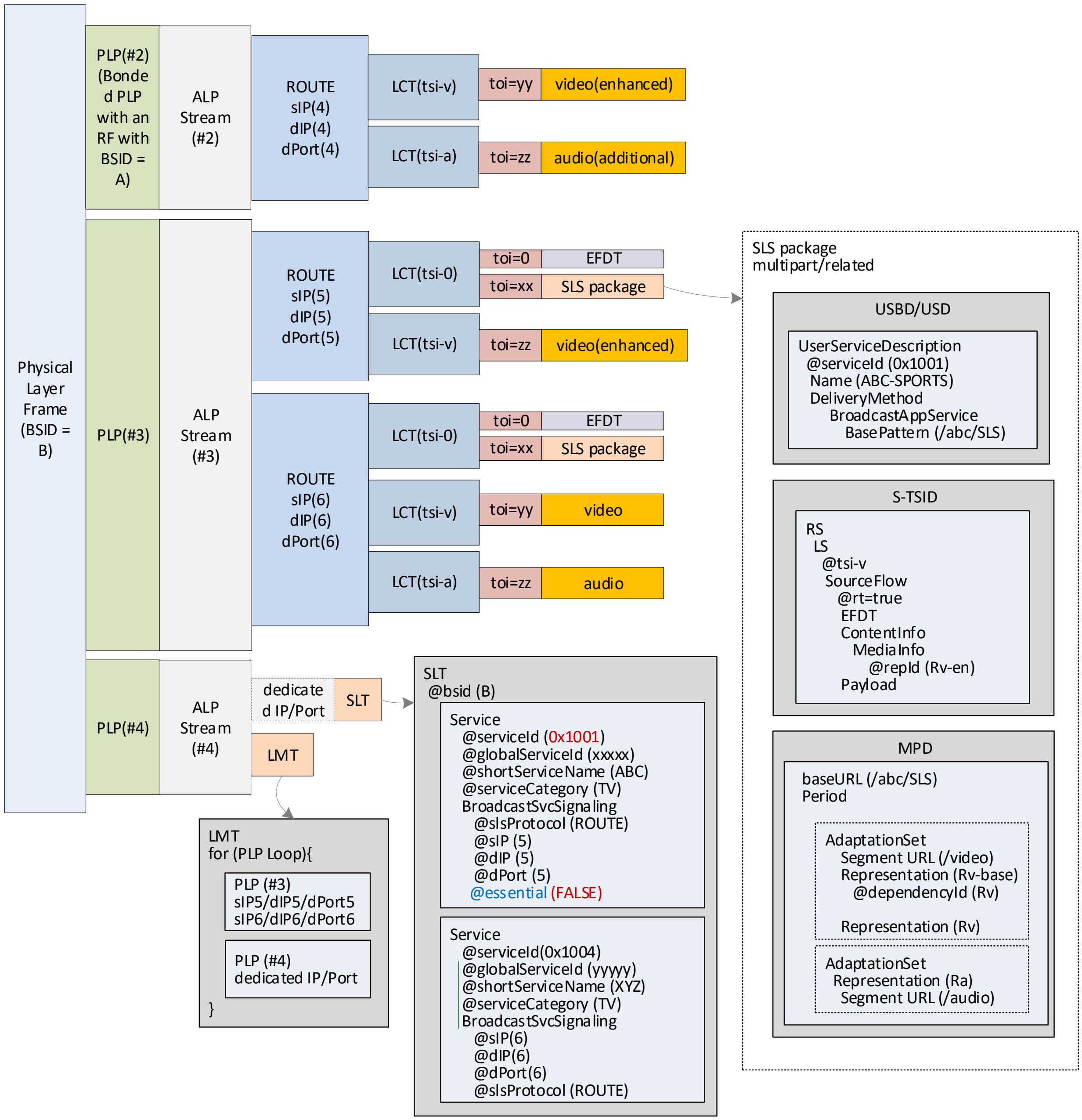

Figure B.14.2 Portions of a Service delivered in multiple RF channels with channel bonding – Case #1: essential portion is delivered in non-bonded PLP(s) (second RF channel).

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 the S-TSID 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".

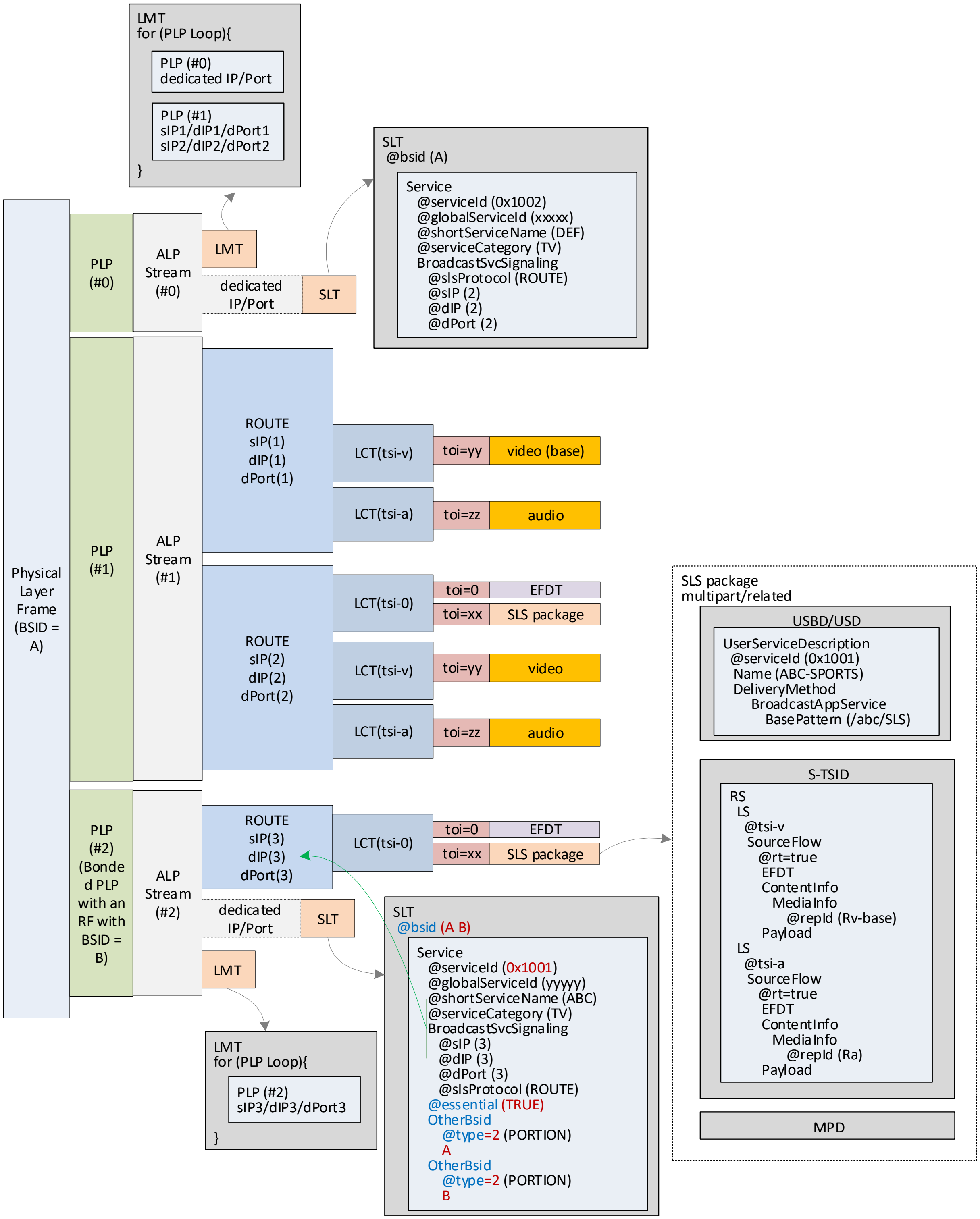

Figure B.15.1 Portions of a Service delivered in multiple RF channels with channel bonding – Case #2: essential portion is delivered in bonded PLP(s) (first RF channel).

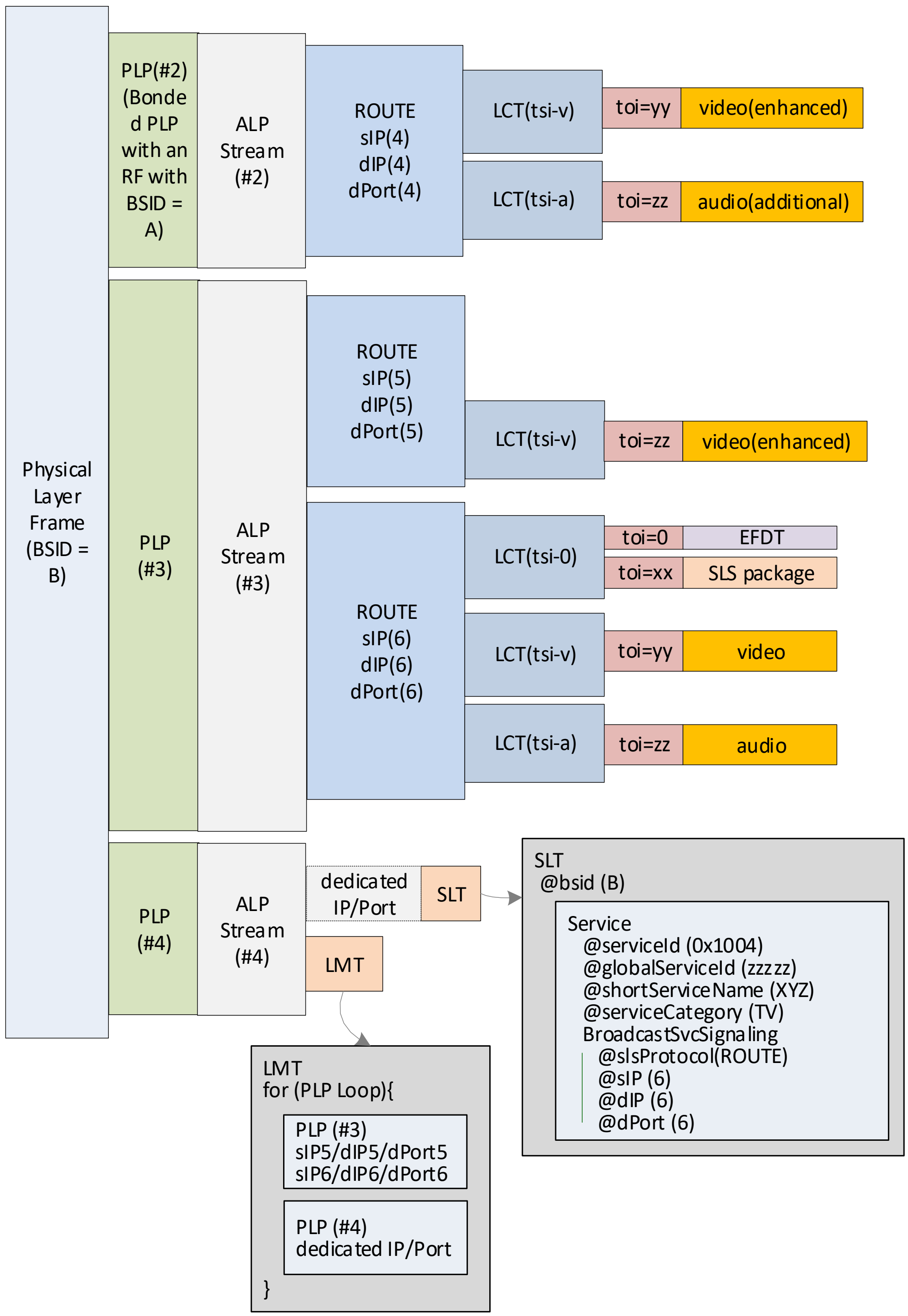

Figure B.15.2 Portions of a Service delivered in multiple RF channels with channel bonding – Case #2: essential portion is delivered in bonded PLP(s) (second RF channel).

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.

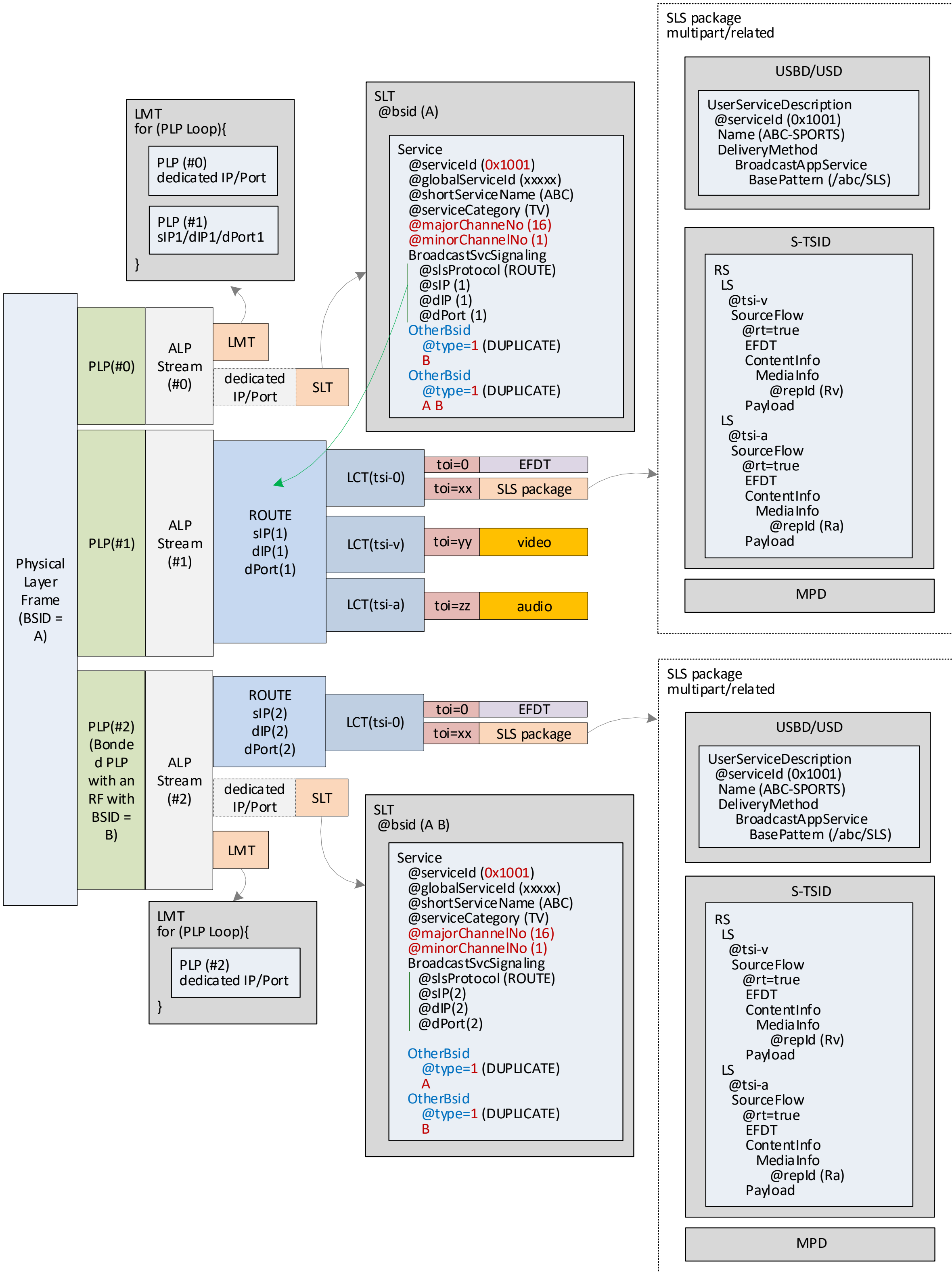

Figure B.16.1 Duplicates of a Service delivered in multiple RF channels with channel bonding (first RF channel).

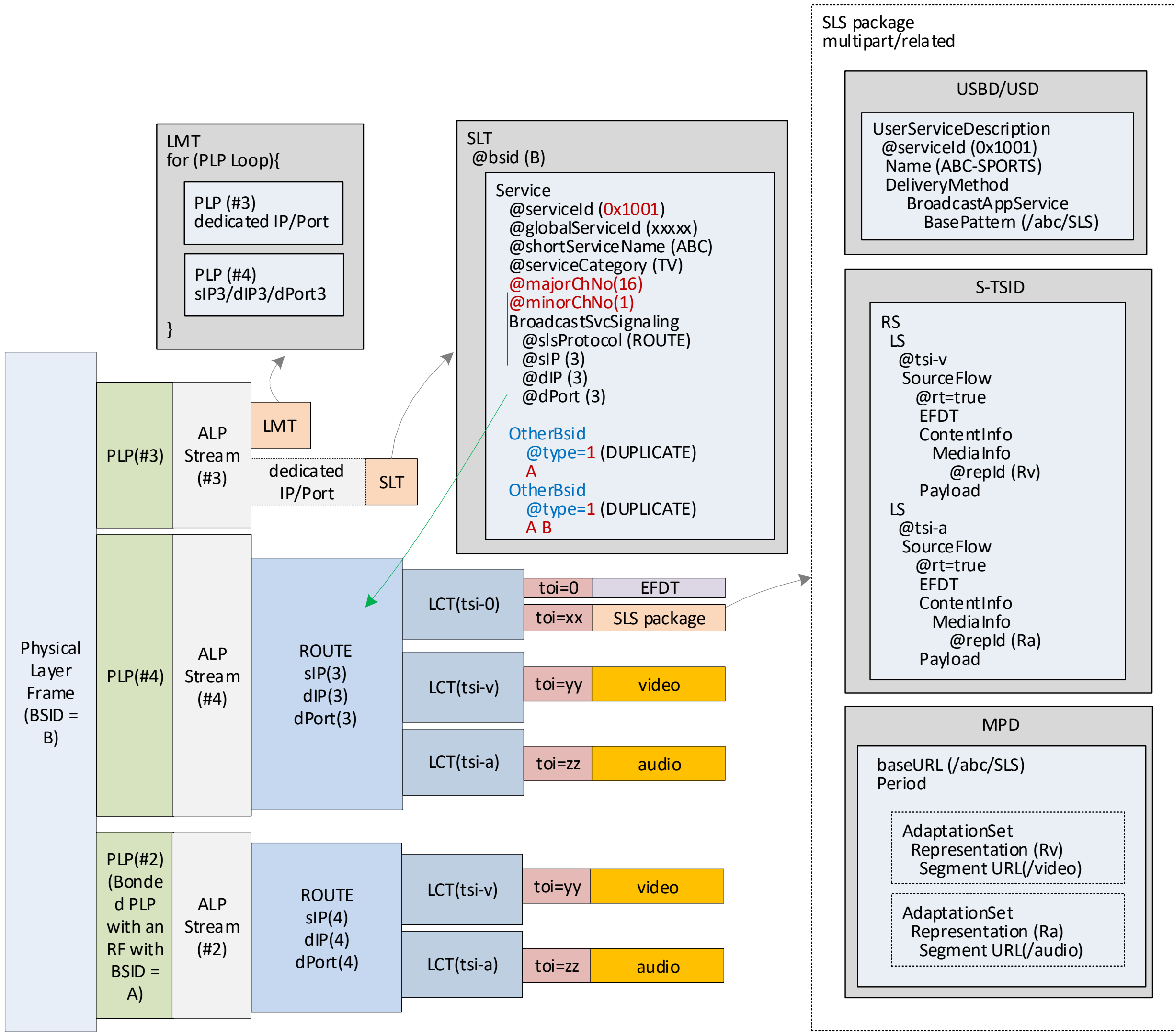

Figure B.16.2 Duplicates of a Service delivered in multiple RF channels with channel bonding (second RF channel).

Annex C: Filtering for Signaling Fragments

The TOI value for SLS packages shall be computed using the following formula:

$$\mathrm{TOI} = \mathbf{G}*2^{31} + \mathbf{U}*2^{16} + \mathbf{S}*2^{17} + \mathbf{M}*2^{18} + \mathbf{A}*2^{19} + \mathbf{H}*2^{22} + \mathbf{D}*2^{23} + \mathbf{R}*2^{24} + \mathbf{Version}$$

where:

G – is equal to 1 when the package containing the SLS fragment(s) is compressed with gzip [16] or 0 when the package is uncompressed;

U – is equal to 1 when the USBD/USD is contained in this package or 0 when USBD/USD is not contained in this package.

S – is equal to 1 when the S-TSID is contained in this package or 0 when S-TSID is not contained in this package.

M – is equal to 1 when the MPD is contained in this package or 0 when MPD is not contained in this package.

A – is equal to 1 when the APD is contained in this package or 0 when APD is not contained in this package.

H – is equal to 1 when the HELD is contained in this package or 0 when HELD is not contained in this package.

D – is equal to 1 when the DWD is contained in this package or 0 when DWD is not contained in this package.

R – is equal to 1 when the RSAT is contained in this package or 0 when RSAT 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  |R|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: (deprecated)

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.

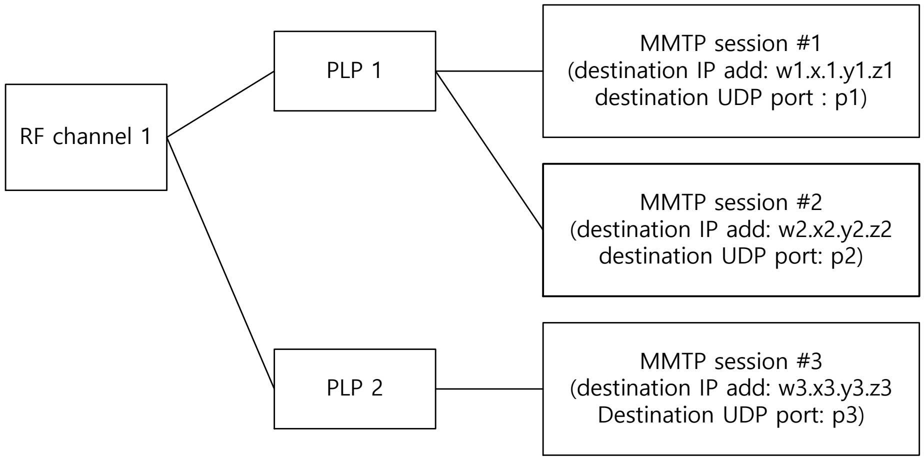

Figure E.2.1 A PHY channel consisting of two PLPs.

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 a 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.

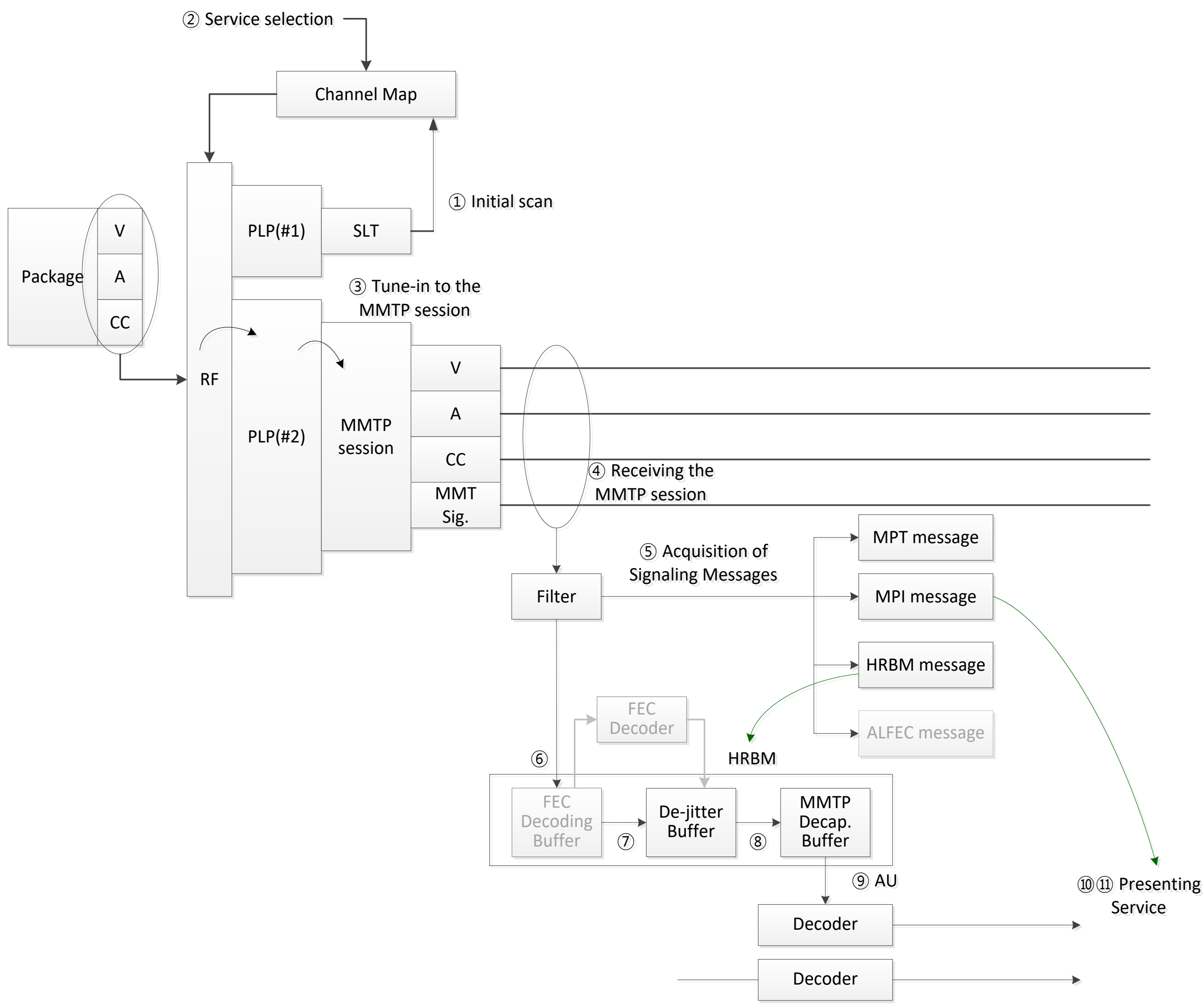

Figure E.3.1 MPU broadcast streaming channel change.

The steps 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 MMT_package_id.
2) The User selects a Service and its MMT_package_id is 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 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 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 Component are selected (filtered) and stored in their corresponding FEC Decoding Buffers. MMTP packets with repair symbols corresponding to each 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 immediately copied to the De-jitter Buffer.
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 Units (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-1.0-20210128*, 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 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".
@dimensionIndex	1	unsignedByte	The rating dimension index for the rating dimension.
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.

`@dimensionIndex` – An unsigned byte value that shall indicate the rating dimension index for the rating 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			string	
	@lang	0..1	lang	The language of the string. Default is 'en' (English).

TextType – A text string.

@lang – The language of the string specified according to BCP 47 [33]. 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

Advanced Emergency Information (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, that receiver 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, the receiver can “wake up” and process emergency signaling information, including providing picture and/or sound to a viewer. Figure G.1.1 shows specific receiver behaviors regarding how a receiver can be switched between the two different states.

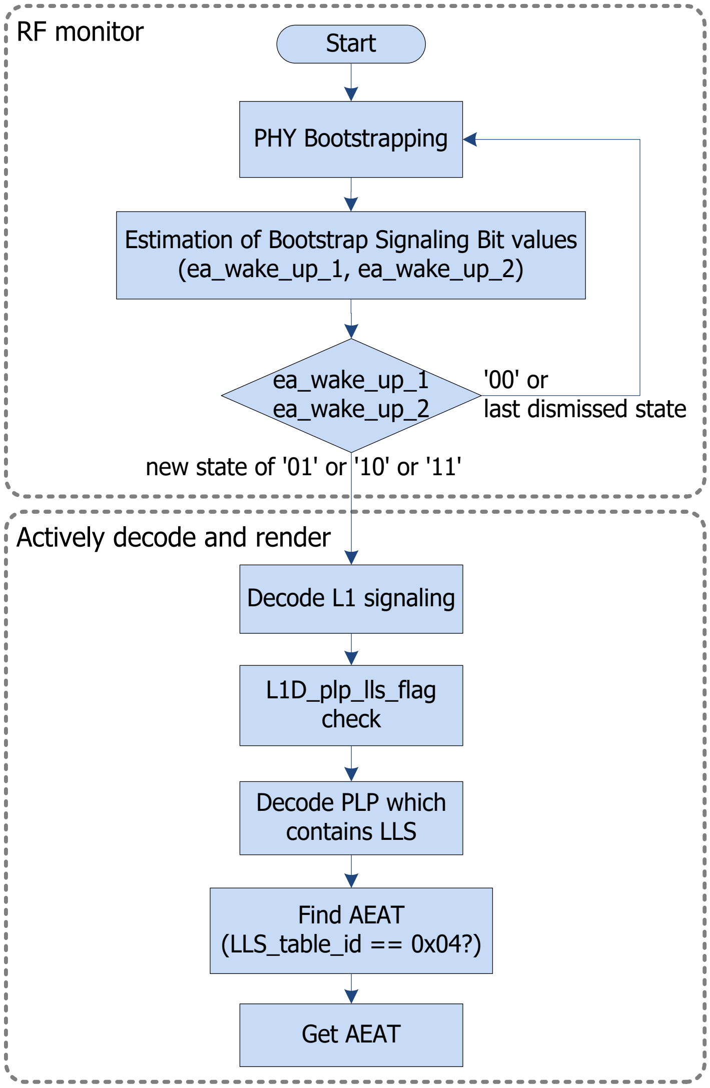

Figure G.1.1 A receiver behavior in two different states.

- Emergency Alert System (EAS) messages – Normally broadcast as of 2019 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 [68].
- 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.
- Broadcaster Application (BA) – A Broadcaster Application conforming to the ATSC 3.0 Interactive Content Standard, A/344 [54], which can provide information pertaining to the alert. Note that Broadcaster Applications that will present emergency information should subscribe to AEAT according to the API documented in A/344 [54].

- 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 BA.
- 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 illustrates the overall Emergency Alert System architecture in the United States.

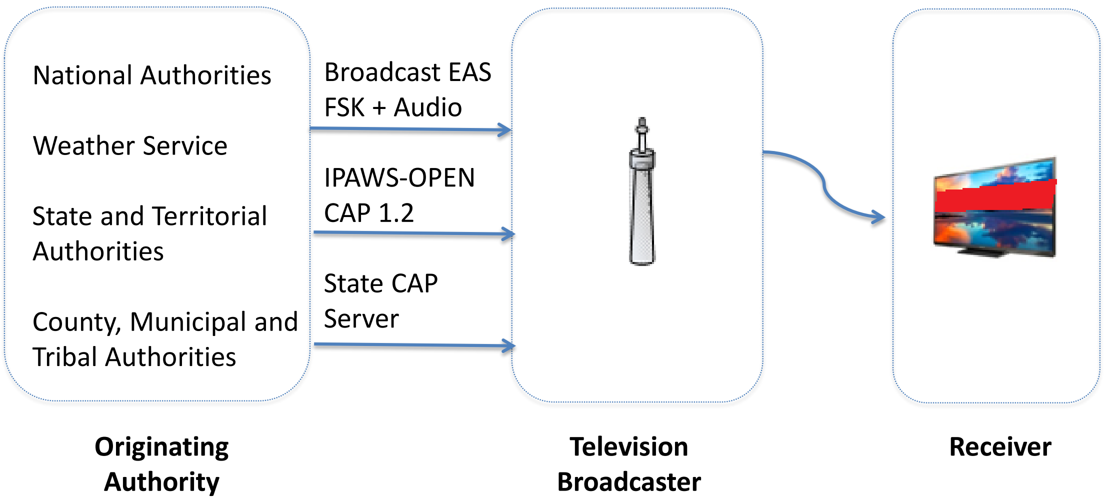

Figure G.1.2 High level architecture of the U.S. Emergency Alert system.

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 illustrate the differences in public alert and warning architectures in the United States and Canada, respectively.

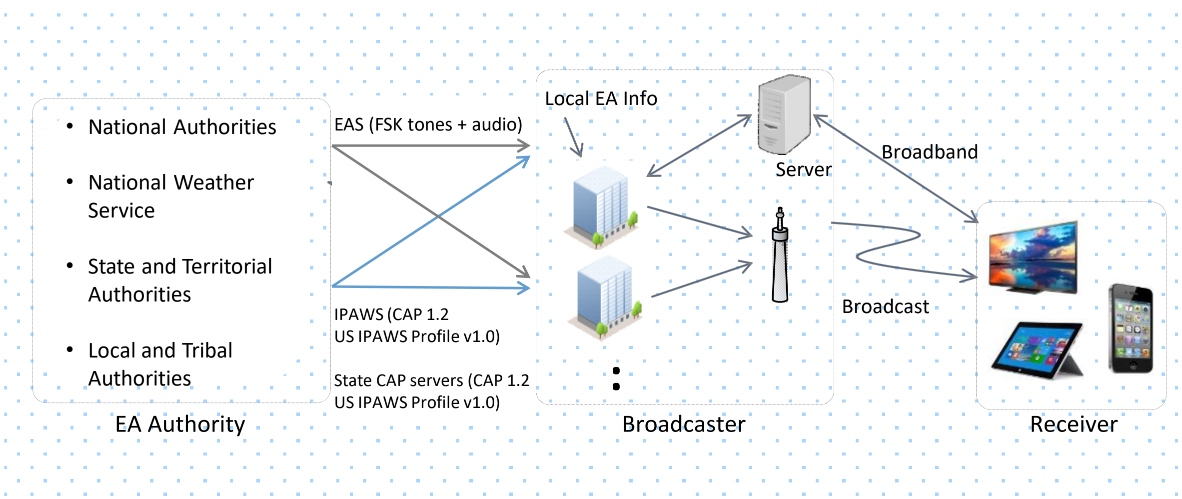

Figure G.1.3 Information Flows in the U.S. Emergency Alert System.

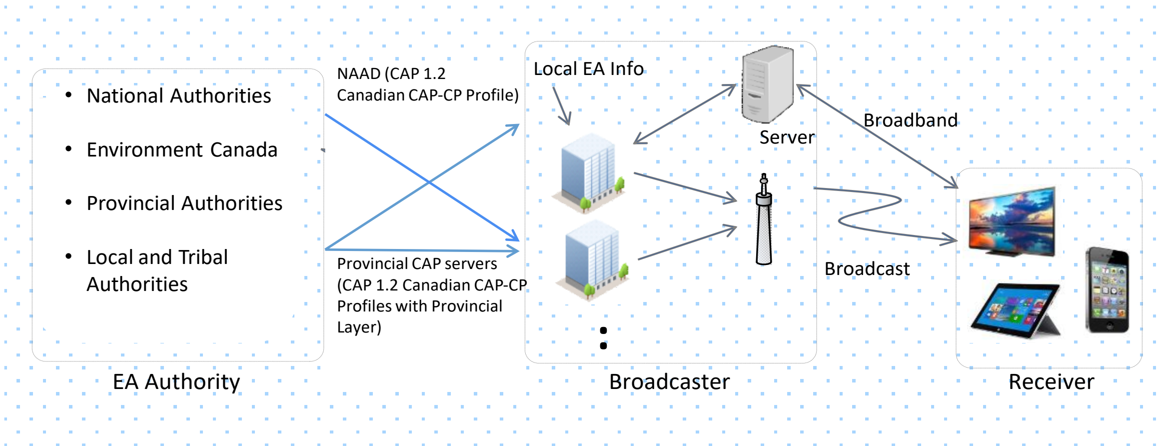

Figure G.1.4 Information Flows in the Canadian National Alert Aggregation and Distribution System.

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 providing additional 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 provides a more detailed view of example flows of emergency alert information, where a station has received an emergency alert message from an external (governmental) source.

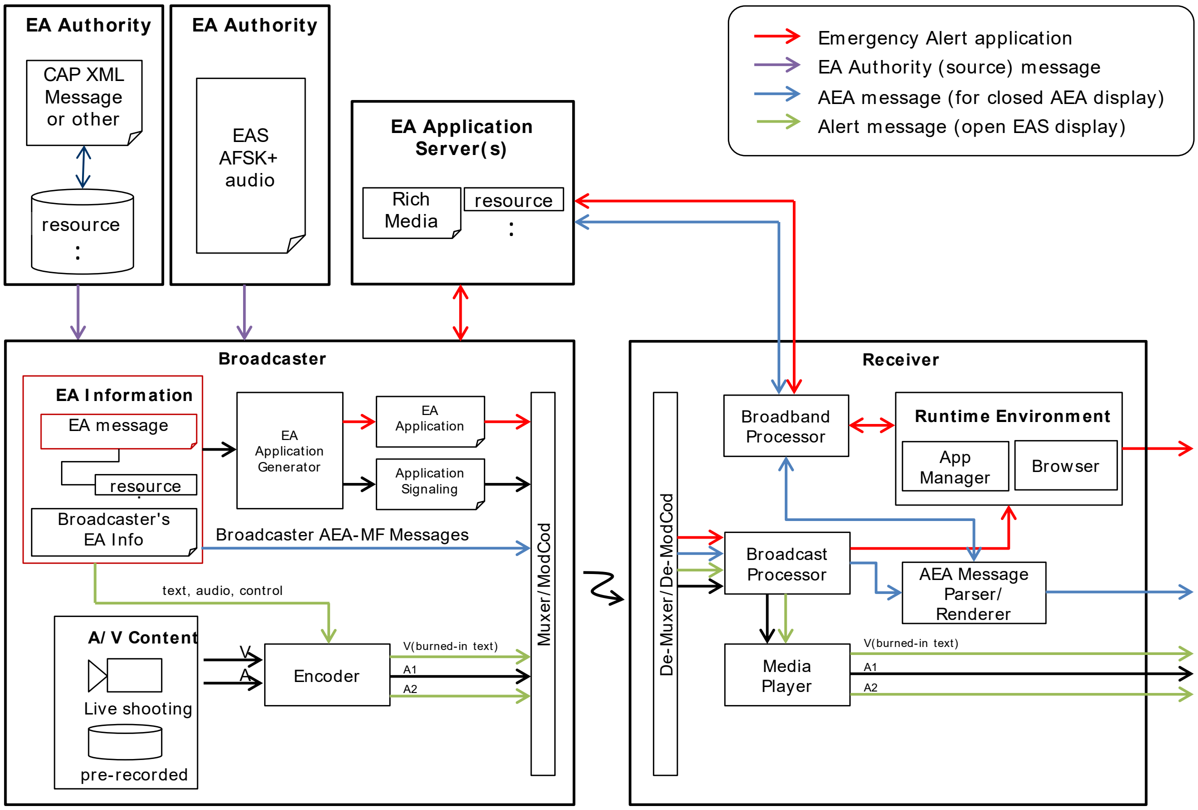

Figure G.1.5 Emergency Alert data flows.

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[11]. 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;
- transmission of the alert as an AEA-only, without the conventional on-screen EAS display.

[11] Notification methods other than use of the CAP message are possible.

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 BA (or content for an existing BA) 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 BA 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 provides a more detailed view of example flows of emergency alert information, where a station originates emergency content from its own sources. This content could include emergency messaging, emergency news, multimedia, or other urgent content/information.

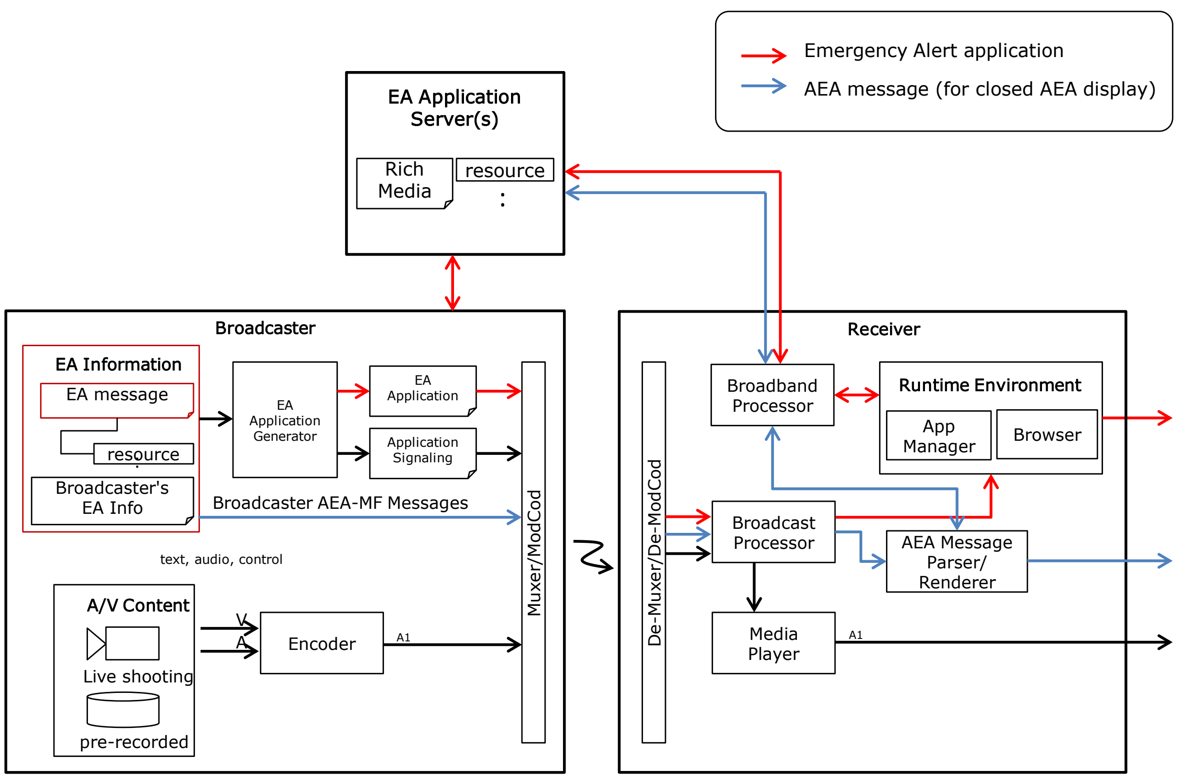

Figure G.1.6 Emergency Alert data flows.

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 **Wake-up Field** herein. The value of **Wake-up Field** 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 **Wake-up Field** and any transitions between values, it is possible to differentiate a new wake-up notification from a previous wake-up notification. The **Wake-up Field** 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 **Wake-up Field** 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 **Wake-up Field**, 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 **Wake-up Field** shall be as specified in Table G.2.1.

Table G.2.1 Meaning of **Wake-up Field**

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 **Wake-up Field** shall be non-zero. The value of **Wake-up Field** shall change when an AEA message with `AEA@wakeup="true"` is added. The **Wake-up Field** 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 **Wake-up Field** shall be '00'.

When the value of **Wake-up Field** 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 **Wake-up Field** 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 **Wake-up Field** value is required to be non-zero and is required to be zero otherwise (see above).

The value of **Wake-up Field** 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 **Wake-up Field** value. Cancellation of the latest alert will not affect the **Wake-up Field** value so long as another alert that triggered a change in the **Wake-up Field** value remains active. Only when all events that triggered changes in the value of **Wake-up Field** have expired, or the broadcaster chooses to cancel the wake-up signaling, will the **Wake-up Field** value revert to zero.

If the user turns off a receiver while the **Wake-up Field** value is non-zero, it is expected that the receiver will remain off unless a new, non-zero **Wake-up Field** 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. When 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 Section 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 [27] 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.

A BA can be designed to handle presentation of AEA-MF message information and the rich media resources signaled in an AEA-MF message. The BA is signaled according to Section 7.1.2 and invokes the “subscribe AEAT” API according to A/344, Interactive Content [54]. 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 [61].

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 [63], 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 [63] 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 [63] 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 [63] 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 [63] 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.

H.5 HELD

Type name:
application
Subtype name:
atsc-held+xml
Required parameters:
None.
Optional parameters:
None.
Encoding considerations:
binary.
Same as for application/xml, except constrained to UTF-8. See IETF 7303, Section 9.1.
Security considerations:
This media type inherits the issues common to all XML media types – see RFC 7303 [63] 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 7.1.8. 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):

.held

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.6 DWD

Type name:

application

Subtype name:

atsc-dwd+xml

Required parameters:

None.

Optional parameters:

None.

Encoding considerations:

Binary.

Same as for application/xml, except constrained to UTF-8. See IETF 7303, Section 9.1.

Security considerations:

This media type inherits the issues common to all XML media types – see RFC 7303 [63] 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 7.1.9. 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):

.dwd

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 —