Adaptive media streaming over IP multicast

DVB Document A176 Rev.6
(Seventh edition)

August 2024

Contents

Intellectual Property Rights

Essential patents

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

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

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

Foreword

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

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

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

The DVB Project is an industry-led consortium of broadcasters, manufacturers, network operators, software developers, regulators and others from around the world committed to designing open, interoperable technical specifications for the global delivery of digital media and broadcast services. DVB specifications cover all aspects of digital television from transmission through interfacing, conditional access and interactivity for digital video, audio and data. The consortium came together in 1993.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

Video delivery has become a dominant class of traffic on public networks. The wider market has embraced unicast streaming with the ability to adapt to network conditions as a means of delivering media on any type of access network. One of the reasons for its widespread adoption is the reuse of existing network technologies used to deliver other Internet services, in particular HTTP and Content Delivery Networks. Dynamic bit rate adaptation allows the streaming session to degrade gracefully as network conditions worsen, and to recover as they improve.

For consumption of the same linear media stream at the same time by a large audience, the number of simultaneous connections to the edge serving infrastructure carrying the same media payloads results in a high degree of redundancy which can be mitigated by the use of multicast packet replication at Layer 3. Unicast streaming is better suited to unsynchronised media consumption, or consumption of linear streams by smaller audiences.

By combining existing media encoding and packaging formats with the efficiency of point-to-multipoint distribution to the edge of IP-based access networks, it is possible to design a system for linear media distribution that is both efficient and scalable to very large audiences, while remaining technically compatible with the largest possible set of already-deployed end user equipment.

Point-to-multipoint topologies also offer opportunities for efficient pre-positioning of assets to devices at the edge of the network. This supports additional non-linear use cases and can help to alleviate peak demand on the access network at synchronization points in the linear schedule.

1 Scope

The present document specifies a reference functional architecture for an end-to-end system that delivers linear content over Internet Protocol (IP) networks in a scalable and standards-compliant manner. Scalability is achieved by means of IP multicast operating in parallel with and alongside conventional unicast delivery. The procedures and protocols used between the system functions are specified, along with data interchange formats, where appropriate.

2 References

2.1 Normative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

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

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 9112: "HTTP/1.1", June 2022.

[2] IETF RFC 9110: "HTTP Semantics", June 2022.

[3] Void.

[4] Void.

[5] IETF RFC 9111: "HTTP Caching", June 2022.

[6] Void.

[7] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3", August 2018.

[8] IETF RFC 1952: "GZIP file format specification version 4.3", May 1996.

[9] IEFF IETF RFC 4648: "The Base16, Base32, and Base64 Data Encodings", October 2006.

[10] ETSI TS 103 285: "Digital Video Broadcasting (DVB); MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP Based Networks", v1.3.1 February 2020.

[11] ISO 8601-1:2019: "Date and time — Representations for information interchange — Part 1: Basic rules", First edition, February 2019.

[12] ISO/IEC 15938-5:2003: "Information technology — Multimedia content description interface — Part 5: Multimedia description schemes", First edition, May 2003.

[13] IETF RFC 5952: "A Recommendation for IPv6 Address Text Representation", August 2010.

[14] IETF RFC 2046: "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", November 1996.

[15] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax", January 2005.

[16] IETF RFC 6585: "Additional HTTP Status Codes", April 2012.

[17] ETSI TS 126 346: "Universal Mobile Telecommunications System (UMTS); LTE; 5G; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs (3GPP TS 26.346 Release 17)".

[18] ATSC A/331:2019: "ATSC Standard: Signaling, Delivery, Synchronization, and Error Protection", 20 June 2019.

[19] IETF RFC 5651: "Layered Coding Transport (LCT) Building Block", October 2009.

[20] IETF RFC 9530: "Digest Fields", February 2024.

[21] IETF RFC 5775: "Asynchronous Layered Coding (ALC) Protocol Instantiation", April 2010.

[22] IETF RFC 6381: "The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types", August 2011.

[23] IETF RFC 3926: "FLUTE - File Delivery over Unidirectional Transport", October 2004.

[24] IETF RFC 5445: "Basic Forward Error Correction (FEC) Schemes", March 2009.

[25] IETF RFC 5053: "Raptor Forward Error Correction Scheme for Object Delivery", October 2007.

[26] IETF RFC 6330: "RaptorQ Forward Error Correction Scheme for Object Delivery", August 2011.

[27] IETF RFC 950: "Internet Standard Subnetting Procedure", August 1985.

[28] IETF RFC 4291: "IP Version 6 Addressing Architecture", February 2006.

[29] IETF RFC 5740: "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", January 2021.

[30] IETF RFC 5510: "Reed-Solomon Forward Error Correction (FEC) Schemes", December 2018.

[31] IETF Internet Draft: "Encoding rules and MIME type for Protocol Buffers", draft-rfernando-protocol-buffers-00, April 2013.

[32] IETF Internet Draft: "MSYNC", draft-bichot-msync-15, December 2023.

[33] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications", July 2003.

[34] IETF RFC 9113: "HTTP/2", June 2022.

[35] IETF RFC 9114: "HTTP/3", June 2022.

[36] IETF RFC 7136: "The JavaScript Object Notation (JSON) Data Interchange Format", March 2014.

[37] Linux Foundation OpenAPI Initiative: "OpenAPI Specification v3.0.1", https://spec.openapis.org/oas/v3.0.1.

[38] IETF RFC 9421: "HTTP Message Signatures", February 2024.

[39] IETF RFC 5280: "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", May 2008.

[40] IETF RFC 5234: "Augmented BNF for Syntax Specifications: ABNF", January 2008.

[41] DVB A177r6: "Service Discovery and Programme Metadata for DVB-I", January 2024.

[42] IANA: "Hash Algorithms for HTTP Digest Fields", February 2024.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

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

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] Void.

[i.2] ISO/IEC 23009-1: "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats".

[i.3] IETF RFC 4918: "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", June 2007.

[i.4] DASH Industry Forum: "DASH-IF Live Media Ingest Protocol" technical specification. https://dashif-documents.azurewebsites.net/Ingest/master/DASH-IF-Ingest.html.

[i.5] IETF RFC 8216: "HTTP Live Streaming", August 2017.

[i.6] ETSI TS 103 770: "Digital Video Broadcasting (DVB); Service Discovery and Programme Metadata for DVB-I".

[i.7] IETF RFC 6762: "Multicast DNS", February 2013.

[i.8] Fielding, Roy Thomas: "Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D. dissertation). University of California, Irvine, 2000.

[i.9] Broadband Forum TR-069: "CPE WAN Management Protocol".

[i.10] Broadband Forum TR-369: "User Services Platform (USP)".

[i.11] ISO/IEC 23000-19: "Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media".

[i.12] DASH Industry Forum: "Guidelines for Implementation: DASH-IF Interoperability Points", Version 4.3, November 2018.

[i.13] DVB BlueBook A181: "Adaptive media streaming over IP multicast; Implementation guidelines and worked examples".

[i.14] IETF RFC 1918: "Address Allocation for Private Internets", February 1996.

[i.15] IETF RFC 4193: "Unique Local IPv6 Unicast Addresses", October 2005.

[i.16] IETF RFC 3927: "Dynamic Configuration of IPv4 Link-Local Addresses", May 2005.

[i.17] IETF RFC 1122: "Requirements for Internet Hosts -- Communication Layers", October 1989.

[i.18] ISO/IEC 14496-12: "Information technology — Coding of audio-visual objects — Part 12: ISO base media file format".

[i.19] Marc Stevens, Arjen K. Lenstra, Benne de Weger: "Chosen-prefix collisions for MD5 and applications". International Journal of Applied Cryptography, Vol. 2, No. 4, 2012.

[i.20] IETF RFC 8792: "Handling Long Lines in Content of Internet-Drafts and RFCs", June 2020.

3 Definition of terms, symbols and abbreviations

3.1 Terms

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

ancillary media object: a media object conveyed via a multicast transport session or via a unicast transport session that does not comprise linear content

NOTE: Examples of ancillary media objects include presentation manifests, initialisation segments and other resources supporting the delivery of the linear service such as DVB-I service discovery and programme metadata [i.6].

ancillary ingest object: an ingest object that is an ancillary media object

ancillary multicast transport object: a multicast transport object that is an ancillary media object

carousel: a repeating schedule of multicast transport objects conveyed via a multicast transport session

NOTE: The repetition frequency need not be the same for every multicast transport object in a carousel.

content: any media object involved in a streaming session, including the presentation manifest and packaged media

NOTE: Each such object is uniquely identifiable by a Uniform Resource Identifier.

ingest object: media object offered at reference point $\mathbf{P_{in}'}$ or $\mathbf{O_{in}}$

NOTE: The *Multicast server* function maps each ingest object to one or more multicast transport objects.

linear service: set of service components that together make up an editorial linear user experience such as a television channel or radio station

media object: single externally addressable unit of packaged encoded media essence or related metadata to be conveyed via a multicast transport session or via a unicast transport session, e.g. a presentation manifest or DASH Segment identified by a URL

NOTE: Control plane documents, such as the multicast session configuration, are not media objects.

multicast media transport protocol: specification of the packet syntax to be used at reference point **M** between a *Multicast server* and a population of *Multicast gateway* and/or *Unicast repair* functions in a given deployed system

multicast session: time-bound transmission comprising all the multicast transport sessions required to deliver a linear service according to operational needs

NOTE: Not all service components need be delivered by multicast transport sessions at all times: some service components may be conveyed by unicast transport sessions alone. This is an operational decision.

multicast session configuration: set of multicast transport session parameters that completely describe all of the multicast transport sessions that comprise a single multicast session

multicast transport object: binary resource related to a media presentation that is conveyed in a multicast transport session and is uniquely identifiable in the multicast media transport protocol, e.g. a partial or complete media object

multicast transport object identifier: a value defined in a multicast media transport protocol that uniquely identifies a multicast transport object

NOTE: In the case of FLUTE [23] and ROUTE [18], the multicast transport object identifier concept is realised by the LCT Transport Object Identifier [19].

multicast transport session: time-bound stream of packets transmitted from a *Multicast server* to a *Multicast gateway* via reference point **M**, formatted according to a particular multicast media transport protocol, that conveys a multiplex of one or more service components and/or Forward Error Correction information

NOTE: A multicast transport session is uniquely identifiable by the combination of a source IP address, a destination multicast group IP address and a destination UDP port number, collectively referred to as the *<S, G, p>* triplet.

multicast transport session parameters: set of metadata defining the technical parameters required to deliver or receive a single multicast transport session

playback delivery object: media object provided in response to an HTTP Request at reference point **L**

presentation manifest: metadata providing information used in the playback of a linear service

NOTE: Examples include the Master Playlists (.m3u8) for an HLS streaming session [i.5], the ISML file for a Microsoft SmoothStreaming session, the Media Presentation Description (MPD) for a DVB DASH session [10] or the SDP file and MPEG-DASH MPD for a 3GPP MBMS session.

Presentation session: consumption of a linear service by a *Content playback* function making requests for a presentation manifest and playback delivery objects at reference point **L**

presentation timeline: schedule of media objects that need to be acquired by a *Content playback* function in order to sustain seamless playback of a particular linear service

NOTE 1: This may be governed by timing metadata in the media objects themselves, and an explicit timing model embedded in the presentation manifest, or it may be implied by media objects being listed in the presentation manifest as they become available for download.

NOTE 2: This term is consistent with the concept of Media presentation timeline in MPEG-DASH [i.2].

service component: sequence of media objects that is part of a linear service, e.g. a video stream or an audio stream

service description metadata: media object(s) used to describe the technical parameters of a single linear service

NOTE 1: The service description metadata for a particular linear service includes references (such as URLs) to one or more presentation manifests.

NOTE 2: The format of the service description metadata and the means of its acquisition both lie outside the scope of the present document.

system operator: business entity responsible for operating the functions specified in the present document

terminal device: consumer device that renders presentation sessions of linear services

3.2 Symbols

Void.

3.3 Abbreviations

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

ALC	Asynchronous Layered Coding
AL-FEC	Application Level FEC
API	Application Programmer's Interface
ATSC	Advanced Television Systems Committee
BBC	British Broadcasting Corporation
BMFF	Base Media File Format
CCI	Congestion Control Information (pertaining to LCT)
CDN	Content Delivery Network
CMAF	Common Media Application Format
CP	Content Provider
DASH	Dynamic Adaptive Streaming over HTTP
DNS	Domain Name System
DRM	Digital Rights Management
EPG	Electronic Programme Guide
ESI	Encoding Symbol Identifier (pertaining to AL-FEC)
FDT	File Delivery Table (pertaining to FLUTE)
FEC	Forward Error Correction
FEC	Forward Erasure Correction
FLUTE	File Delivery over Unidirectional Transport
FQDN	Fully-Qualified Domain Name (pertaining to DNS)
HLS	HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IP	Internet Protocol
ISO	International Organization for Standardization
LCT	Layered Coding Transport
MBMS	Multimedia Broadcast Multicast Services (pertaining to 3GPP)
MPD	Media Presentation Description (pertaining to MPEG-DASH)
MPEG	Moving Pictures Experts Group
MTU	Maximum Transmission Unit
NAT	Network Address Translation
PES	Packetized Elementary Stream (pertaining to MPEG-2 Transport Stream)
PID	Packet Identifier (pertaining to MPEG-2 Transport Stream)
PSI	Protocol-Specific Indication (pertaining to LCT)
ROUTE	Real-time Object delivery over Unidirectional Transport
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SAST	Segment Availability Start Time
SDP	Session Description Protocol
STB	Set-Top Box
S-TSID	Service-based Transport Session Instance Description (pertaining to ROUTE)
TLS	Transport Layer Security
TOI	Transport Object Identifier (pertaining to ALC/LCT)
TSI	Transport Session Identifier (pertaining to ALC/LCT)
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator (pertaining to HTTP)
XML	eXtensible Markup Language

4 General

The present document specifies a reference functional architecture for an end-to-end system that delivers linear content over Internet Protocol (IP) networks in a scalable and standards-compliant manner. Scalability is achieved by means of IP multicast operating in parallel with and alongside conventional unicast delivery. The individual functions required for such a system are depicted in figure 5.2-1, and the interactions between them are shown as named reference points. The functional architecture is intended as an abstract reference: real implementations may, for example, combine multiple functions in a single deployable unit. The architecture is intended to be independent of any particular Internet Protocol address family.

Two mandatory multicast media transport protocols are specified in annex F and annex H:

- Every conformant realization of the *Multicast server* shall implement at least one of the mandatory multicast media transport protocols. A conformant realization may additionally implement one or more optional multicast media transport protocols specified in subsequent normative annexes to the present document.
- Every conformant realization of the *Multicast gateway* shall implement at least one of the mandatory multicast media transport protocols. A conformant realization may additionally implement one or more optional multicast media transport protocols specified in subsequent normative annexes to the present document.
- Every conformant realization of the *Unicast repair server* implementing reference point **M** shall implement at least one of the mandatory multicast media transport protocols. A conformant realization may additionally implement one or more optional multicast media transport protocols specified in subsequent normative annexes to the present document.

In the following clauses, references to HTTP refer to the Hypertext Transfer Protocol message exchange semantics as defined in IETF RFC 9110 [2]. Implementations shall, at minimum, conform to HTTP/1.1 protocol syntax as specified in IETF RFC 9112 [1] and may optionally implement subsequent protocol syntaxes, for example HTTP/2 [34] or HTTP/3 [35].

Similarly, references to HTTPS refer to HTTP/1.1 or HTTP/2 over TLS 1.3 [7] or HTTP/3 (for which the use of HTTPS is required). Implementations may optionally implement equivalent or better transport layer security protocols.

References to HTTP(S) mean "HTTP or HTTPS".

Implementation guidelines and worked examples are provided in the companion document [i.13].

5 Reference architecture

5.1 Reference points

5.1.0 Introduction

The relationships between the logical functions in the reference architecture are identified by named reference points. In a practical deployment, each of these is realized by a concrete interface and conveys information between the relevant functions using a particular protocol. The reference points and the logical functions are illustrated by the reference architecture diagram in clause 5.2 and are summarized in figure 5.1.0-1 below.

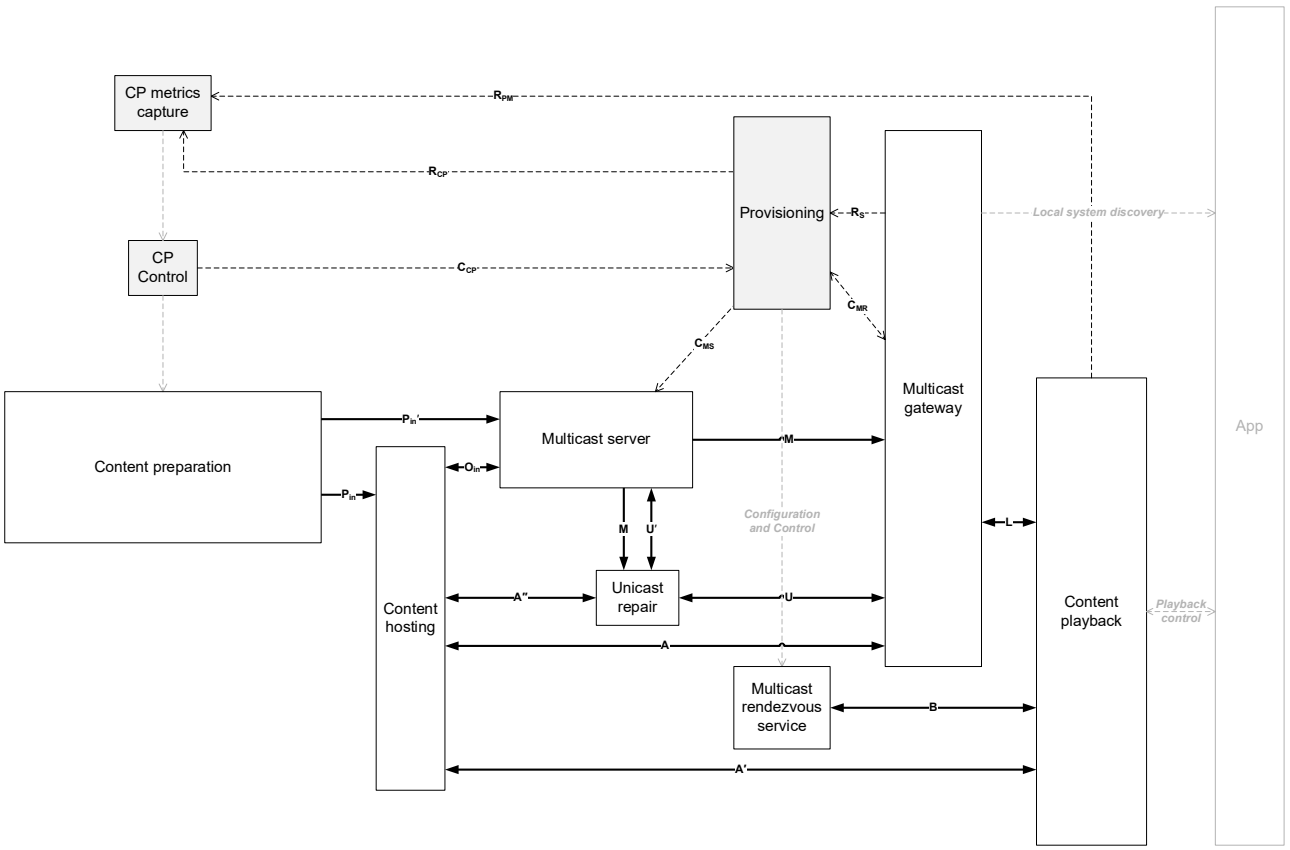


Figure 5.1.0-1: Simplified reference architecture

5.1.1 Data plane reference points

The reference points defined in this clause are used primarily to transport content:

L Unicast HTTP(S) [2] interaction between a *Content Playback* function and a *Multicast gateway*. This interface includes the fetching of all specified types of content.

NOTE: When a *Multicast gateway* and a *Content playback* function are co-located on a single end device, such as a set-top box (see clause 6.3), reference point **L** may be realized as a local API.

B Bootstrap unicast HTTP(S) interaction directly between a *Content playback* function and a *Multicast rendezvous service* function. Used to request the presentation manifest at the start of a linear playback session, as outlined in clause 5.3.9 and specified in clause 7.5.

A Used by a *Multicast gateway* for HTTP(S) acquisition from a *Content hosting* function of content not provided over reference point **M**. Also used by a *Multicast gateway* for retrieving content directly from a *Content hosting* function via unicast when **U** is unable to perform content repair, as specified in clause 9.

A′ Used by a *Content playback* function to retrieve unicast HTTP(S) content out of scope for provision over reference point **L** from a *Content hosting* function.

A″ Used in some deployments by a *Unicast repair service* function to retrieve unicast HTTP(S) content from a *Content hosting* function in order to effect content repair.

M Multicast IP content transmission by a *Multicast server* function and reception by a *Multicast gateway* function and, in some deployments, reception by a *Unicast repair service* function.

U Unicast interaction between a *Unicast repair client* in a *Multicast gateway* and a *Unicast repair service*. This interface may be used to carry the payloads used for content repair functions in addition to the requests for such payloads.

U′ The unicast interaction between a *Unicast repair service* and a *Multicast server* as an alternative to fetching repair content over **A**. This interface may be used to carry the payloads used for content repair functions in addition to the requests for such payloads.

P_{in} Publication of content to a *Content hosting* function by a *Content packaging* subfunction. This could be implemented as a push interface, or content could be pulled on demand from a *Content packaging* function.

O_{in} Ingest of content by a *Multicast server* function from a *Content hosting* function. This is typically implemented as a pull interface.

P_{in}' Ingest of content by a *Multicast server* direct from a *Content packaging* function. This is typically implemented as a push interface.

5.1.2 Control plane reference points

The reference points defined in this clause are used for control signalling and operational reporting.

C_{MS} Control interface for configuration of a *Multicast server* function.

C_{MR} Control interface for configuration of a *Multicast gateway* function.

C_{CP} Control interface for configuration of a *Provisioning* function.

R_S Service reporting by a *Multicast gateway* function to a *Service reporting capture* function.

R_{CP} Service reporting by a *Service reporting capture* subfunction to a *Content Provider metrics reporting capture* function.

R_{PM} Reporting of playback metrics by a *Content playback* function to a *Content Provider metrics reporting capture* function.

5.2 Reference architecture diagram

A diagram of the reference architecture is shown on the next page (figure 5.2-1). Logical functions are depicted as named boxes and these may be nested in cases where a high-level function is composed of several subfunctions. Functions that lie within the scope of the present document are shown with black text. Those beyond the scope of the present document (but relevant to the functional architecture) are depicted with grey text. Functions lying primarily in the data plane are shown with unfilled boxes; control plane functions are shaded.

Data plane interactions are depicted using solid lines. Control plane interactions are depicted using dotted lines. Interactions that lie within the scope of the specification are depicted as black lines with a reference point name. Those beyond the scope of the present document (but relevant to the functional architecture) are shown with grey lines.

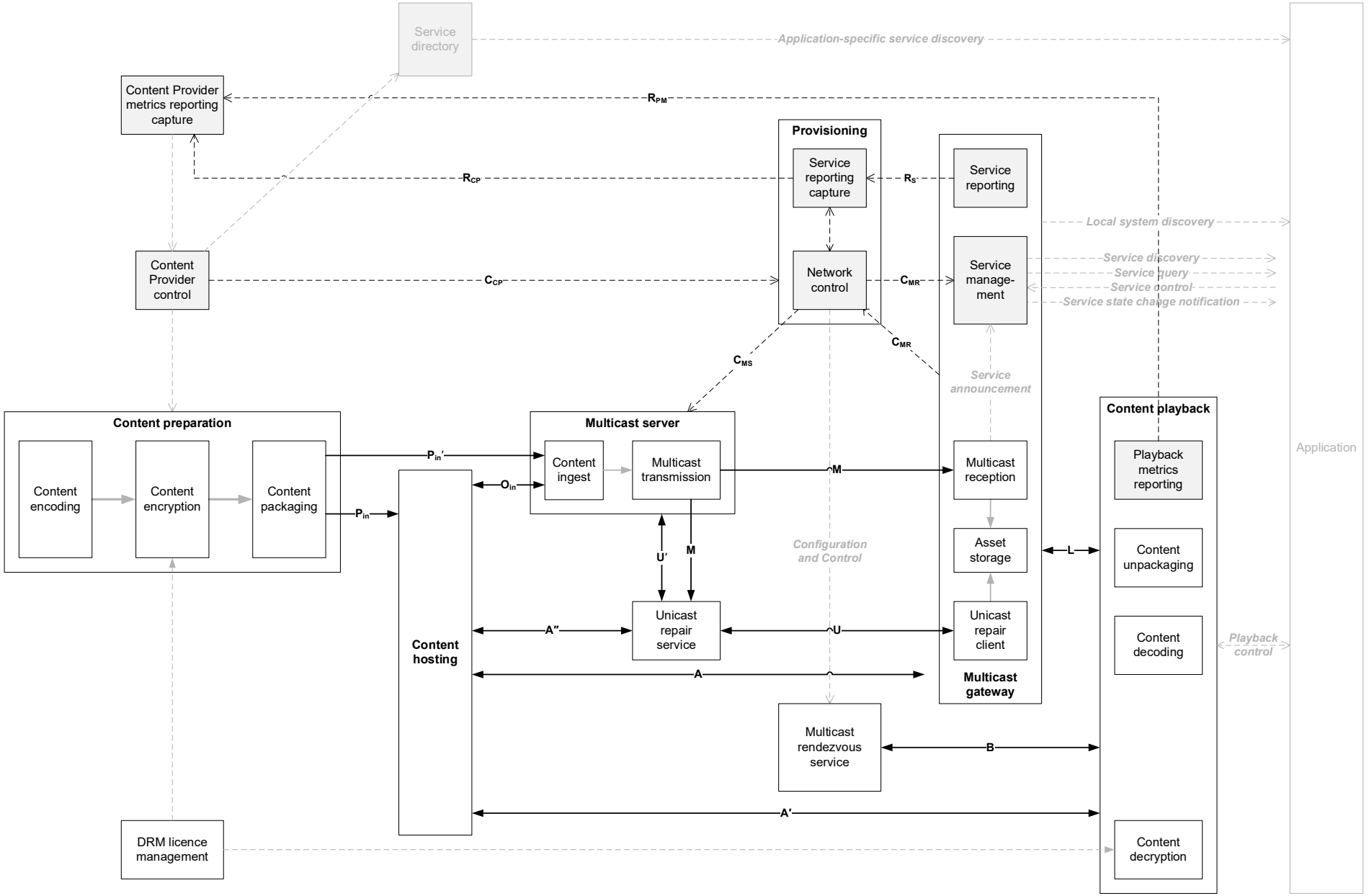


Figure 5.2-1: Reference architecture

5.3 Functions

5.3.1 Content preparation

5.3.1.0 Introduction

The operation of the *Content preparation* function is specified in clause 8.1.

5.3.1.1 Content encoding

The *Content encoding* function transforms source media streams into encoded media with the aim of reducing the bit rate. A single source media stream may be transformed into a number of different encoded representations to match delivery conditions. Virtual segment boundary markers may be placed in the encoded media representation to assist an adaptive *Content playback* function in its operation.

The output of the encoder is a set of unencrypted media streams formatted so as to be suitable for ingest by an encrypter or packager. This could, for example, be an MPEG elementary stream, an MPEG-2 Transport Stream, or some other proprietary intermediate format.

5.3.1.2 Content encryption

The *Content encryption* function takes a cleartext stream and encrypts some or all of it to form a ciphertext stream. The encryption keys are obtained from the *DRM licence management* function.

This function is optional in the case where encryption is not a requirement for a particular stream.

5.3.1.3 Content packaging

The *Content packaging* function ingests the media streams of one (or more) encoded representations and formats each one according to a desired packaging format. In the context of dynamic adaptive streaming, the output of the packager is a sequence of packaged media segments with representation switching points that are aligned across different representations of the same source media stream.

Examples of packaging formats suitable for fragmentation into media segments are ISO Base Media File Format and MPEG-2 Transport Stream.

5.3.2 Content hosting

Prepared content is made available by the *Content hosting* function for unicast delivery to the *Multicast server* (for content ingest via reference point **O_{in}**), to the *Multicast gateway* (for cache misses via reference point **A)**, to instances of the *Content playback* function that are not connecting through a *Multicast gateway* (at reference point **A′**), or to the *Unicast repair service* (at reference point **A″**).

The *Content hosting* function may be realized as simple web servers, as part of an origin cluster, or operating as a distributed Content Delivery Network system. As such, load balancing and request distribution techniques (e.g. DNS round-robin, HTTP 302 redirect) may be used to direct clients to the most appropriate content server.

The operation of the *Content hosting* function is specified in clause 8.2.

5.3.3 Multicast server

5.3.3.0 Introduction

The *Multicast server* function ingests content from the configured content sources. In the simple case, media streams are retrieved, typically using the same protocols that a media player might employ, via reference point **O_{in}**.

In the *Multicast server*, the payloads of the ingested media stream are encapsulated into the delivery units of the multicast media transport protocol and transmitted through the *Network* to subscribed *Multicast gateway* clients using IP multicast via reference point **M**.

This entity can be configured by the *Network control* function via reference point $\mathbf{C_{MS}}$.

NOTE: This entity uses a Source-Specific Multicast methodology for sending and serving multicast traffic.

The operation of the *Multicast server* function is specified in clause 8.3.

5.3.3.1 Content ingest

Both push and pull content ingest methods are possible for the *Multicast server*:

- **HTTP(S) Pull Ingest via reference point $\mathbf{O_{in}}$:** The subfunction mimics a conventional adaptive streaming media player and downloads packaged media segments from the *Content hosting* function as described by a presentation manifest. In this case, reference point $\mathbf{O_{in}}$ may be functionally identical to **L** (although its operational behaviour may differ). Segments may, for example, be packaged using MPEG-DASH or HLS. Segments from one or more representations of the presentation may be downloaded simultaneously.

NOTE: DVB DASH, MPEG-DASH, HLS and other manifest formats may be supported.

- **HTTP(S) Push Ingest via reference point $\mathbf{P_{in}'}$:** The subfunction offers an HTTP(S) push interface, such as the `PUT` method specified for WebDAV [i.3]. The *Content packaging* subfunction uploads media segments to the *Content ingest* function immediately as they are created. Segments may, for example, be packaged using MPEG-DASH or HLS.

- **RTP Push Ingest via reference point $\mathbf{P_{in}'}$:** The subfunction offers an RTP-based push ingest mechanism to the *Content packaging* subfunction. The packager sends MPEG-2 Transport Stream packets using RTP. Segment boundaries are marked with virtual segment boundary markers.

5.3.3.2 Multicast transmission

This subfunction is responsible for serializing streams received by the *Content ingest* subfunction and for transmitting the serialized streams in the payloads of IP multicast packets via interface **M**.

5.3.4 Unicast repair service

The *Unicast repair service* offers a payload repair function to the *Unicast repair client* in the *Multicast gateway* via reference point **U**.

The following repair modes could be considered for the *Unicast repair service*:

1) The *Unicast repair service* listens to multicast content transmissions over reference point **M** and locally caches a copy of the packet stream which it uses to satisfy repair requests received from the *Unicast repair client*.

2) If the requested packet(s) cannot be satisfied from the *Unicast repair service*'s cache, packet repair requests may be passed to the *Multicast server* via reference point **U'**.

3) Packet repair requests are converted by the *Unicast repair service* to equivalent HTTP(S) requests (e.g. byte ranges) on the *Content hosting* function via reference point **A"**.

4) If near-simultaneous requests for the same repair are received by the *Unicast repair service* from multiple *Multicast gateways*, it may be more efficient for the repair packets to be transmitted using multicast via reference point **M**.

5.3.5 Multicast gateway

5.3.5.0 Introduction

The primary purpose of this function is to provide packaged content segments to the *Content playback* function. The *Multicast gateway* may be realized as a forward proxy or as a local origin (including reverse proxy).

The *Multicast gateway* could be instantiated in customer premises equipment like a home gateway device or IP-connected Set-Top Box (STB). It could also be located in an upstream network node as an alternative to the customer's premises.

Content requests are received, via reference point **L**, from one or more instances of the *Content playback* function and these are serviced either directly from content cached in the *Asset storage* subfunction or indirectly via reference point **A**, with retrieved content then optionally cached in the *Asset storage* subfunction.

Unicast fill operations are performed via reference point **A** until a cache is established in *Asset storage* for a given linear service.

NOTE: Some streams may never be sent using a multicast session, while others may require a short period of time before the cache is established.

The operation of the *Multicast gateway* function is specified in clause 8.4.

5.3.5.1 Service management

The *Service management* subfunction collates multicast session configuration information about multicast content streams available via interface **M** as well as the location(s) of the *Service reporting capture* function. This information may be received from one or more of the following sources:

- Directly from *Network control* via reference point $\mathbf{C_{MR}}$ (see clauses 10.4.4.1 and 10.4.4.2).
- Indirectly via the *Multicast reception* subfunction, in the case where the information is transmitted over reference point **M** (see clauses 10.4.5 and 8.3.5).
- In unicast responses from the *Multicast rendezvous service* function carried over reference point **B** (see clauses 8.5 and 7.5).

5.3.5.2 Multicast reception

The *Multicast reception* subfunction ingests content streams via interface **M** that have been requested by or configured for an end device. Content that has been received intact may also be cached in *Asset storage* for later use. Content damaged in transit may be repaired using any specified mechanism(s) at the *Multicast gateway*'s disposal (e.g. Forward Error Correction, unicast repair by the *Unicast repair client* via **U** or unicast retrieval via **A**) prior to caching. Irreparable content should not be served via reference point **L**.

5.3.5.3 Unicast repair client

Multicast packet loss detection is performed and recovered from using either Forward Error Correction information received via interface **M**, loss recovery using the *Unicast repair service* via interface **U** (e.g. unicast packet retransmission or multicast segment loss signalling) or, as a last resort, unicast fill via reference point **A**.

5.3.5.4 Asset storage

The *Asset storage* subfunction provides temporary storage of assets to be served over reference point **L**. Authority over the storage lies with the *Multicast gateway*.

- Managed pre-positioned media content assets. For example, pre-positioning all or part of a popular asset, or advertising material in advance of its availability date to a large population of users.
- Temporary caching of linear media content segments.

5.3.5.5 Service reporting

Service-related metrics (e.g. telemetry and analytics data) are reported by the *Service reporting* subfunction to the *Service reporting capture* subfunction via interface $\mathbf{R_S}$.

5.3.6 Provisioning

5.3.6.0 Introduction

The purposes of the *Provisioning* function are:

1) To collect service reporting information centrally from the deployed *Multicast gateway* instances.

2) To configure resources in the *Network*.

3) To configure the *Multicast server* to use the configured *Network* resources.

4) To configure the *Multicast gateway* to use the configured *Network* resources.

The *Provisioning* function may be influenced by the *Content Provider control* function via reference point **C_{CP}**.

5.3.6.1 Service reporting capture

Service reporting information captured by the *Multicast gateway* is supplied to the *Service reporting capture* function via reference point **R_S**. The reports may include metrics and other key indicators describing the performance of the service (e.g. cache hit ratio, viewership). The metrics depend on which channels are requested, when channels are established and how many segments are in cache. The service reporting information could be used for instance to improve service performance and to configure multicast channels.

The *Service reporting capture* function may also export service reporting information to the *Content Provider metrics reporting capture* function via reference point **R_{CP}**. This information may include data on multicast content and bit rate.

5.3.6.2 Network control

This function is responsible for controlling, configuring and provisioning *Network* resources. This includes the resources for multicast transmission (over reference point **M**) and well as those for unicast operation (over reference points **U**, **A**, **A′**, **A″** and **B**).

In systems with centralized co-ordination, the *Network control* function distributes configuration information about the set of available multicast streams to the *Network* resources and may additionally distribute this configuration information to the *Multicast server* (via **C_{MS}**) and/or to the *Multicast gateway* (via **C_{MR}**). The configuration information about the set of available multicast streams can be updated according to *Content Provider control* policy rules and/or the number of client requests.

5.3.7 Content Provider control

This function uses the control interface **C_{CP}** provided by the *Network control* function to provision information about services that can be made available over the multicast delivery path **M**. A single *Content Provider control* function may be interacting with multiple *Network control* functions, each one of the latter operated by a different network provider.

5.3.8 Content playback

5.3.8.0 Introduction

This is the entity managing the request, reception, decryption and presentation of content. It only supports unicast delivery via reference point **L**. Playback behaviour is agnostic to the delivery path traversed by the content.

The *Content playback* function may be located separately from the *Multicast gateway* on an end device such as a smartphone (clauses 6.1 and 6.2). It may alternatively be combined with a *Multicast gateway*, for example in a set-top box or connected TV set (clause 6.3).

Additional functions of the *Content playback* function are:

- To retrieve, via reference point **B**, a presentation manifest for the linear service.
- To retrieve, via reference point **B**, any content that is not intended to be retrieved via the *Multicast gateway*.

The operation of the *Content playback* function is specified in clause 8.5.

5.3.8.1 Content unpackaging

The *Content unpackaging* subfunction is responsible for extracting elementary stream data from retrieved transport objects and presenting it to the *Content decryption* and *Content decoding* subfunctions. For example, with ISO Base Media File Format segments this involves extracting the appropriate media data box(es), while with MPEG-2 Transport Streams the desired PID is filtered and the payloads of reassembled PES packets are extracted.

5.3.8.2 Content decryption

If a Digital Rights Management system is in operation, the *Content decryption* subfunction is responsible for obtaining appropriate decryption keys from the *DRM licence management* function and for decrypting any encrypted elementary streams.

5.3.8.3 Content decoding

The *Content decoding* subfunction is responsible for parsing and interpreting the contents of elementary media streams, allowing them to be rendered for playback on, for example, a screen or loudspeakers.

5.3.8.4 Playback metrics reporting

The *Playback metrics reporting* subfunction reports information relating to the behaviour and quality of experience of content playback to the *Content Provider metrics reporting capture* function via reference point **R_{PM}**. The metrics may include (but are not limited to) details of HTTP request/response transactions, initial playback delay, buffer level, representation switching events and measured network throughput. The playback metrics reported by this function are directly related to the end user quality of experience and may be used to optimize the experience either at the Content Provider or in the *Network*.

5.3.9 Multicast rendezvous service

The *Multicast rendezvous service* maintains records of managed *Multicast gateway* instances, including their current status, and records of current multicast sessions, including their status. The *Network control* function is responsible for supplying all of this information to the *Multicast rendezvous service*, but the means by which it is supplied lies outside the scope of the present specification.

The *Multicast rendezvous service* handles the initial request from the *Content playback* function at reference point **B** for a presentation manifest. The *Multicast rendezvous service* determines whether there is an active multicast session for the linear service corresponding to the requested presentation manifest, and whether there is an active *Multicast gateway* suitable for use by the requesting *Content playback* function. If at least the second condition is met, the *Multicast rendezvous service* may redirect the request to that *Multicast gateway* instance. Otherwise, the *Multicast rendezvous service* redirects the request to the *Content hosting* function and the session will proceed using unicast only.

The operational procedures for the *Multicast rendezvous service* are specified in clause 7.5.

5.3.10 DRM licence management

This optional entity is responsible for core content protection services, providing appropriate encryption keys to be used by the *Content encryption* function, and supplying licences to the *Content decryption* subfunction to enable the *Content playback* function to decrypt encrypted content.

5.3.11 Application

The *Application* is responsible for controlling the *Content playback* function. Examples include an embedded control application on an integrated television set or set-top box ("EPG application") or a third-party application contributed by a Content Provider. The interface the *Application* function uses to control the *Content playback* function is outside the scope of the present specification, but would generally involve passing a reference to a presentation manifest (e.g. the URL of an MPEG-DASH MPD) to initiate playback of a particular linear service.

The *Application* may interact with the *Service management* subfunction of the *Multicast gateway* in order to discover the existence of linear services and control their reception by the *Multicast gateway*. At the present time, these interactions are also out of scope.

The *Application* may alternatively discover the existence of linear services through a private interaction with an application-specific *Service directory* function. This interaction is also outside the scope of the present document.

5.3.12 Service directory

The *Application* may use a private *Service directory* in order to locate available linear services. The *Service directory* function may be configured by the *Content provider control* function. Because the *Service directory* function is application-specific it is outside the scope of the present document.

6 Deployment models

6.0 Introduction

The reference architecture described in clause 5 is intended to support the deployment models described in this clause.

6.1 Multicast gateway deployed in network edge device

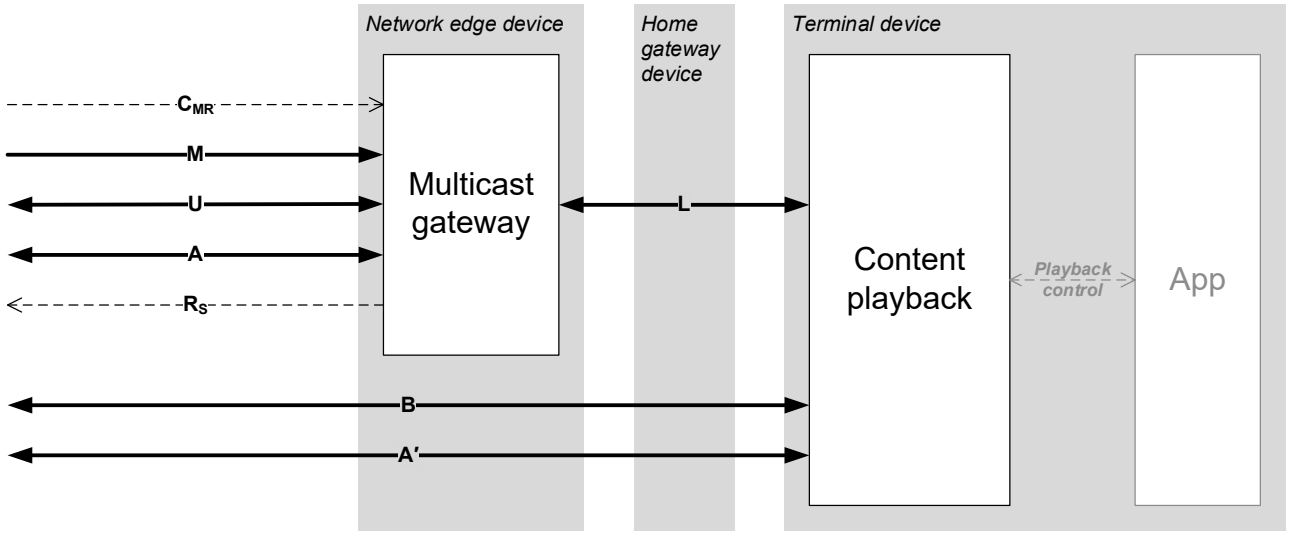


Figure 6.1-1

The terminal device does not support reception of IP multicast from the home network. It includes the *Content playback* function, and loads an *Application* to control linear playback.

The *Multicast gateway* is deployed in a network edge device upstream of the terminal device and provides multicast-to-unicast conversion facilities for several homes. All in-scope traffic on the access network between the network edge device and the home gateway device is therefore unicast.

6.2 Multicast gateway deployed in home gateway device

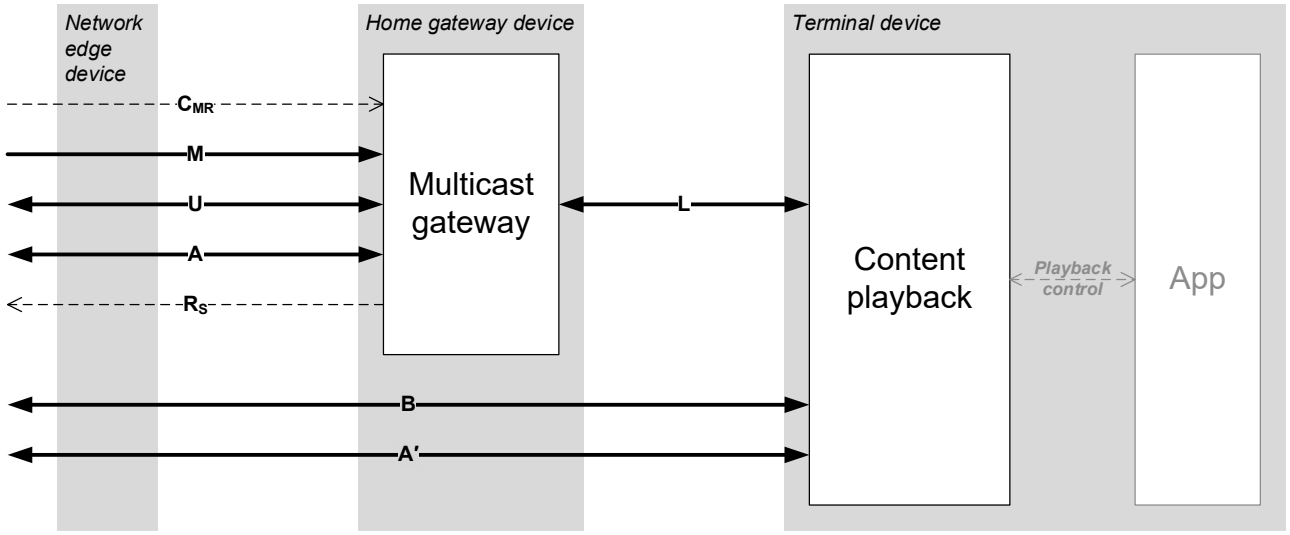


Figure 6.2-1

The *Multicast gateway* is deployed in a home gateway device, such as a router (typically supplied by the Internet Service Provider) and provides multicast-to-unicast conversion facilities for multiple terminal devices in the same home network. These terminal devices each have an instance of the *Content playback* function and load the desired *Application*.

6.3 Multicast gateway deployed in terminal device

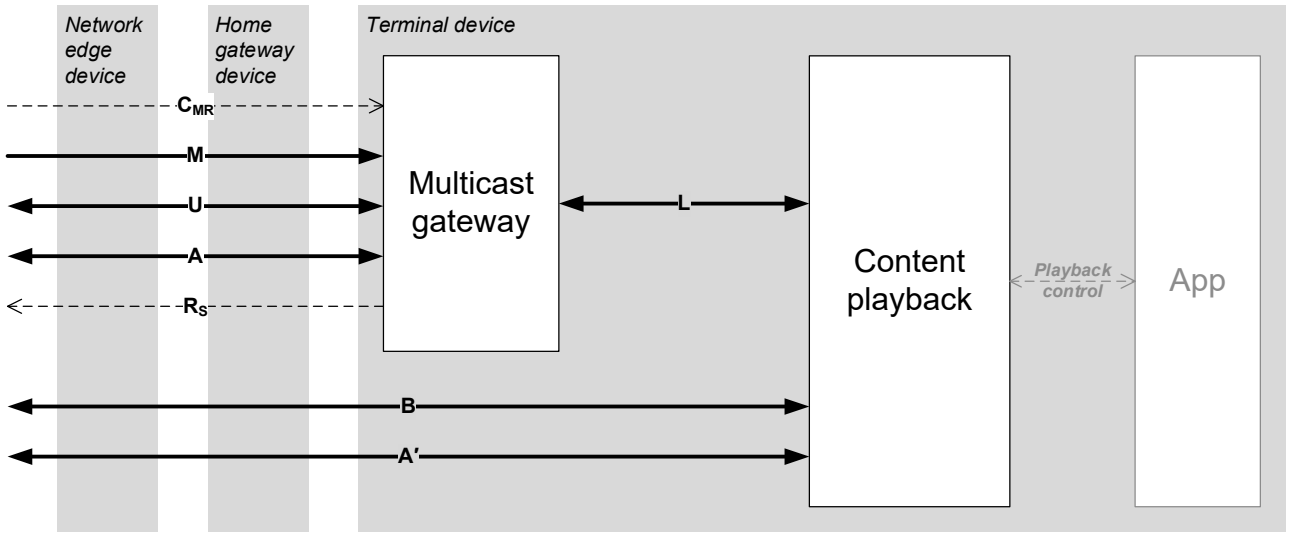


Figure 6.3-1

The terminal device supports reception of IP multicast within the home network. Each such terminal device includes both *Multicast gateway* and *Content playback* functions, and loads its own *Application* to control linear playback. In this deployment model, the *Multicast Gateway* function shall provide content services only to its host terminal device.

The home gateway device performs only multicast group subscription operations. (This may result in unpredictable quality behaviour when the home network does not fully support multicast delivery.)

7 Modes of system operation

7.0 Introduction

Two modes of operation are depicted in the following clauses allowing the support of HTTPS and the support of unidirectional deployment:

- In a **regular deployment** (clause 7.1), the *Multicast rendezvous service* is located in the *Network* and managed by the system operator.
- In a **co-located deployment** (clause 7.2), the *Multicast rendezvous service* is integrated with the *Multicast gateway* in the same entity. It may be used, for example, in the case of a unidirectional deployment scenario.

7.1 Regular deployment

The different functional entities involved in a regular deployment are shown in figure 7.1-1 below and conform to the reference architecture specified in clause 5.

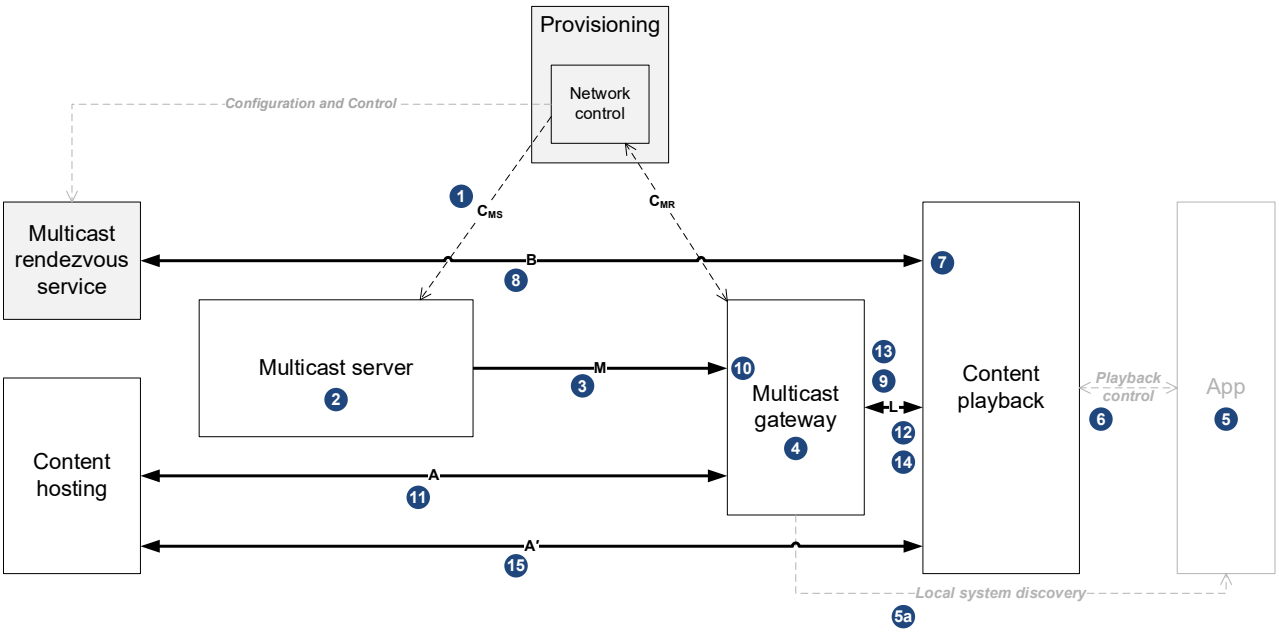


Figure 7.1-1: Regular deployment workflow

The following activities may occur asynchronously at any time:

- The *Provisioning* function may be configured with the details of all available *Multicast gateway* functions by means outside the scope of the present document, or it may be configured dynamically using the procedure specified in clause 7.6.
- The *Provisioning* function keeps the configuration of the *Multicast rendezvous service* up to date by means outside the scope of the present document.
- The *Multicast gateway* may be configured with the current set of provisioned multicast sessions at reference point **C_{MR}**, or at reference point **C_{MS}** and **M**, as specified in clause 10.1.2.
- The *Multicast gateway* may be configured with a set of basic parameters such as the endpoint address of a multicast gateway configuration transport session, as specified in clause 10.4.5.

The workflow is composed of the following steps:

1) The *Network control* subfunction configures the *Multicast server* with the current provisioned set of multicast sessions. The means for providing the multicast server configuration at reference point **C_{MS}** is specified in clause 10.4.2.

2) Once a multicast session has started, the *Multicast server* retrieves (via reference point **O_{in}**) the presentation manifest and media segments from the configured origin server in the *Content hosting* function (if the multicast session indicates the pull content ingest method), or waits for these to be posted via reference point **P_{in}'** (if the push content ingest method is indicated).

3) The *Multicast server* sends the media segments (and, optionally, the presentation manifest) over the *Network* according to the multicast server configuration.

NOTE 1: Sending over the multicast network does not imply that a *Multicast gateway* is receiving packets from the corresponding multicast transport session. The *Multicast gateway* may be required first to enable multicast IP reception as specified in clause 8.4.2.

4) The *Multicast gateway* is active and discoverable by the *Application* (see clause 7.3).

5) The *Application* obtains the URL of the presentation manifest, for example by interacting with a *Service directory* function such as that specified in [i.6]. This URL points to a remote *Multicast rendezvous service* in the network:

   a) Optionally, the *Application* may discover the *Multicast gateway* by means of local system discovery (see clause 7.3). It discovers the IP address and port number of the *Multicast gateway* and the system operator identifier associated with the *Multicast gateway*. The *Application* includes this information as query parameters in the presentation manifest request URL.

6) The *Application* launches the *Content playback* function with the presentation manifest URL.

NOTE 2: From that point the *Content playback* function behaves as a normal ABR media player. Some control/signalling information may be "piggybacked" as URL query parameters over reference point **L** in a manner that is transparent to the *Content playback* function (see clause 7.5).

7) The *Content playback* function resolves the fully-qualified domain name of the *Multicast rendezvous service*.

8) The *Content playback* function requests the presentation manifest from the *Multicast rendezvous service* via reference point **B** (see clause 7.5). The latter checks the *Multicast gateway* status either in its records or as part of the information "piggybacked" in the request URL (see clause 7.5.2.0), performs access control and sends back a redirect URL referring to the *Multicast gateway* if the requested linear service is part of the multicast session configuration. Otherwise, the redirect URL refers to the *Content hosting* function:

   a) The *Multicast rendezvous service* may "piggyback" in the redirect URL an up-to-date multicast session configuration corresponding to the requested presentation manifest (see clause 7.5.2.1).

The following steps assume that the requested service is part of the multicast session configuration.

9) The *Content playback* function follows the redirect and requests the presentation manifest from the *Multicast gateway*.

10) The *Multicast gateway* subscribes to the relevant multicast transport session(s) according to the configuration of the multicast session in question.

11) If the requested presentation manifest is not already cached (for example, received from a multicast gateway configuration transport session at reference point **M**) the *Multicast gateway* retrieves it from the *Content hosting* function via reference point **A**.

12) The *Multicast gateway* returns the presentation manifest back to the *Content playback* function via reference point **L**.

NOTE 3: From this point the *Content playback* function processes the presentation manifest and commences media playback normally.

13) The *Content playback* function requests a media segment (or presentation manifest) at reference point **L**.

14) If the requested media segment (or presentation manifest) is not already cached by the *Multicast gateway* (e.g. not yet received from the corresponding multicast transport session), the latter retrieves the media segment (or presentation manifest) from the *Content hosting* function at reference point **A**. For a segment retrieval only, the *Multicast gateway* may redirect the request to the *Content hosting* function.

15) In the case where a media segment request is redirected, the *Content playback* function retrieves the media segment from the *Content hosting* function via reference point **A′**.

7.2 Co-located deployment

The *Multicast rendezvous* function is co-located with the *Multicast gateway*. This is depicted in figure 7.2-1 below.

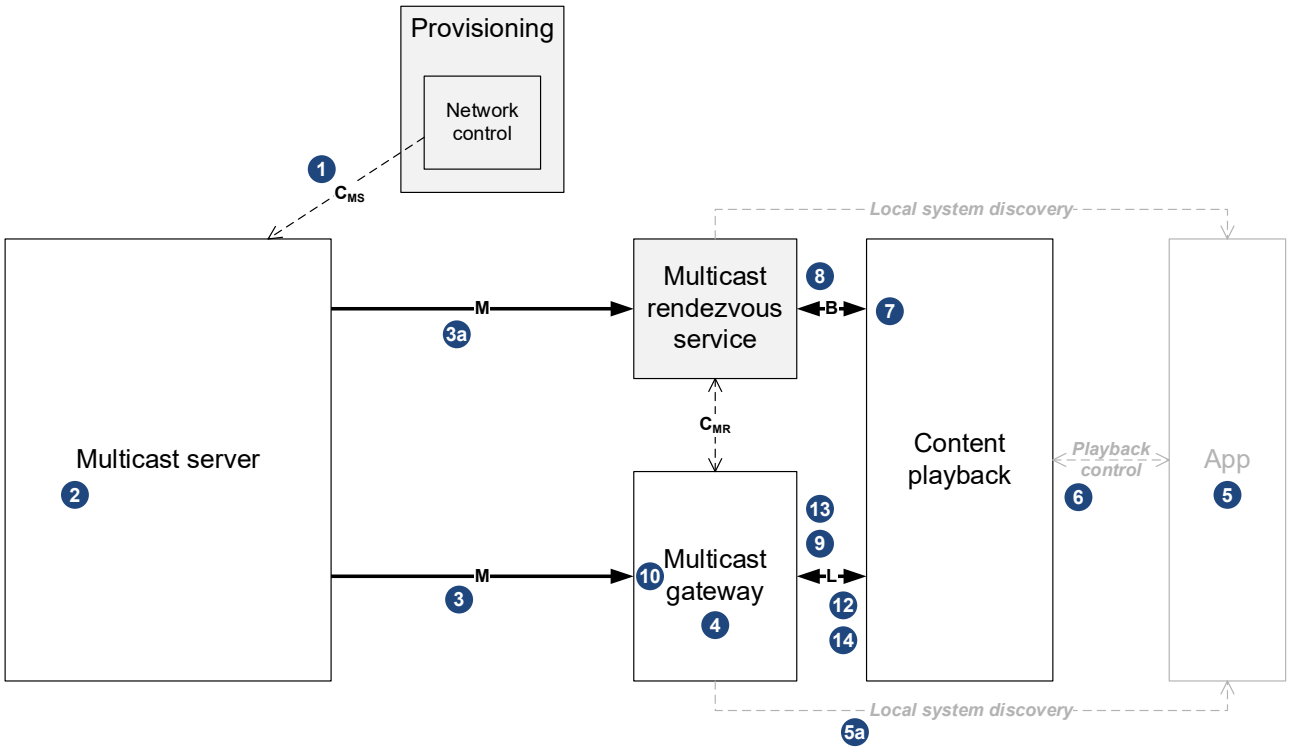


Figure 7.2-1: Co-located deployment workflow

The following activities may occur asynchronously at any time:

- The *Multicast rendezvous service* shall be configured with a set of basic parameters such as the endpoint address of a multicast gateway configuration transport session, as specified in clause 10.4.5.
- The *Multicast gateway* shall be configured with the current set of provisioned multicast sessions at reference point **M**, as specified in clause 10.4.5.

The workflow is composed of the following steps:

1) The *Network control* subfunction configures the *Multicast server* with the current provisioned set of multicast sessions. The means for providing the multicast server configuration at reference point **C_{MS}** is specified in clause 10.4.2.

2) Once a multicast session has started, the *Multicast server* retrieves (via reference point **O_{in}**) the presentation manifest and media segments from the configured origin server in the *Content hosting* function (if the multicast session indicates the pull content ingest method), or waits for these to be posted via reference point **P_{in}′** (if the push content ingest method is indicated).

3) The *Multicast server* sends the presentation manifest(s) as well as the media segments over the *Network* according to the multicast server configuration:

NOTE 1: Sending over the multicast network does not imply that a *Multicast gateway* is receiving packets from the corresponding multicast transport session. The *Multicast gateway* may be required first to enable multicast IP reception as specified in clause 8.4.2.

   a) The *Multicast server* also sends the multicast gateway configuration in a dedicated multicast gateway configuration transport session at reference point **M**. This is described in clause 8.3.5 and clause 10.4.5.

4) The *Multicast gateway* is active and discoverable by the *Application* (see clause 7.3).

5) The *Application* obtains the URL of the presentation manifest, for example by interacting with a *Service directory* function such as that specified in [i.6]. This URL should point to the remote *Multicast rendezvous* function co-localized with the *Multicast gateway*:

   a) Optionally, the *Application* may discover the *Multicast gateway* by means of local system discovery (see clause 7.3). It discovers the IP address and port number of the *Multicast gateway* and the system operator identifier associated with the *Multicast gateway*. The *Application* includes this information as query parameters in the presentation manifest request URL.

NOTE 2: If the presentation manifest request URL has an FQDN in the host part, the *Application* may substitute this FQDN with the IP address and port number of the *Multicast gateway* so discovered.

6) The *Application* launches the *Content playback* function with the presentation manifest URL.

NOTE 3: From that point the *Content playback* function behaves as a normal ABR media player. Some control/signalling information is "piggybacked" as URL query parameters over reference point **L** in a manner that is transparent to the *Content playback* function (see clause 7.5).

7) The *Content playback* function resolves the fully-qualified domain name (if any) of the *Multicast rendezvous service*.

NOTE 4: In the case where a fully qualified Internet domain name is to be resolved, it is assumed that the DNS lookup is processed through a suitable DNS server located either in the Local Area Network, or upstream of it.

8) The *Content playback* function requests the presentation manifest from the *Multicast rendezvous service* via reference point **B** (see clause 7.5). The latter checks the *Multicast gateway* status either in its records or as part of the information "piggybacked" in the request URL (see clause 7.5.2.0), performs access control and sends back a redirect URL referring to the *Multicast gateway* if the requested linear service is part of the multicast session configuration. Otherwise, the redirect URL refers to the *Content hosting* function:

   a) Using information received from the multicast gateway configuration transport session at reference point **M**, the *Multicast rendezvous service* "piggybacks" in its redirect response the current multicast session configuration for the multicast session corresponding to the requested presentation manifest (see clause 7.5.2.1).

The following steps assume that the requested service is part of the multicast session configuration.

9) The *Content playback* function follows the redirect and requests the presentation manifest from the *Multicast gateway*.

10) The *Multicast gateway* subscribes to the relevant multicast transport session(s) according to the configuration of the multicast session in question.

11) If the requested presentation manifest file is not already cached (for example, received from a multicast gateway configuration transport session at reference point **M**) the *Multicast gateway* retrieves it from the *Content hosting* function via reference point **A** (if available).

12) The *Multicast gateway* returns the presentation manifest back to the *Content playback* function via reference point **L**.

NOTE 5: From this point the *Content playback* function processes the presentation manifest and commences media playback normally.

13) The *Content playback* function requests a media segment (or presentation manifest) at reference point **L**.

14) If the requested media segment (or presentation manifest) is not already cached by the *Multicast gateway* (e.g. not yet received from the corresponding multicast transport session), the latter retrieves the media segment (or presentation manifest) from the *Content hosting* function at reference point **A** (if any). For a segment retrieval only, the *Multicast gateway* may instead redirect the request to the *Content hosting* function at reference point **A′** (if any), depending on the *Multicast gateway* implementation.

15) In case of redirection, the *Content playback* function retrieves the media segment (or presentation manifest) from the *Content hosting* function via reference point **A′** (if available).

7.3 Discovering the Multicast gateway using local system discovery (informative)

An *Application* may discover the presence of a *Multicast gateway* (and possibly a co-located *Multicast rendezvous service*) in its local network using, for example, the multicast DNS protocol specified in IETF RFC 6762 [i.7]. This is useful for obtaining information (identification and status) that may be required for instance when dealing with a third-party CDN broker or in the case of a unidirectional deployment.

For example, if a *Multicast gateway* implements IETF RFC 6762 [i.7], it should respond to a multicast DNS query with its IP address, transport port number and the fully-qualified domain name of its service endpoint in the local network.

Following local system discovery, the *Application* should "piggyback" the discovered information as query parameters in the request URL (targeting the *Multicast rendezvous service*) at reference point **B**, as specified in clause 7.5.1 below. In the case of co-located deployment, the *Application* may substitute the request URL's host name (FQDN) with the IP address of a previously discovered *Multicast rendezvous service* (if co-located with the *Multicast gateway*).

7.4 Discovering the Multicast rendezvous service using third-party CDN broker redirect (informative)

Where a third party, such as a content provider, has a business relationship with the system operator, the discovered presentation manifest URL may point initially to a third-party CDN broker service. This service assists the requesting *Content playback* function in locating a suitable *Multicast rendezvous service*.

The *Content playback* function sends its presentation manifest request to the third-party CDN broker service. The latter determines whether multicast operation is possible and desired for the linear service in question. If so, it responds with an HTTP redirect containing the URL of a *Multicast Rendezvous service* accessible to the *Content playback* function. If required by the *Multicast Rendezvous service*, the redirect URL may also need to include an authentication token, as specified in clause 7.5.1 below.

In order to assist the CDN broker in selecting the most suitable *Multicast rendezvous service*, it is recommended that the initial request to the CDN broker includes a query parameter that identifies the system operator. This identifier could for example be obtained through local system discovery (see clause 7.3 above).

7.5 Multicast rendezvous service operational procedures

7.5.0 Introduction

This clause specifies the interaction at reference point **B** between the *Content playback* function and the *Multicast rendezvous service* (see clause 5.3.9). This is typically a request for a presentation manifest at the start of a presentation session. The presentation manifest URL is typically provided by the *Application* and requested by the *Content playback* function.

In some deployments, the presence and details of a *Multicast gateway* instance are available to the *Application* before playback of a linear service commences. In such cases, the *Application* may provide the details of the discovered *Multicast gateway* instance using optional query parameters to the request URL at reference point **B**.

NOTE: The *Multicast rendezvous service* may alternatively be configured by another means, such as the dynamic registration method specified in clause 7.6.

7.5.1 Request URL format

The request URL shall comply with the following syntax:

http[s]://*<Host>*/*<ManifestPath>*[?*<field>*=*<value>*[&*<field>*=*<value>*]*]

The elements of the URL are specified in table 7.5.1-1 below. The query parameters carry various optional information useful for the *Multicast rendezvous* service when redirecting a request from the *Content playback* function to a suitable *Multicast gateway*.

Table 7.5.1-1: Syntax of request URL elements

URL element	Use	Data type	Description
Host	1	String	The FQDN (or the IP address) and optionally the port number of the *Multicast rendezvous service*.
ManifestPath	1	String	The resource path for retrieving the presentation manifest from the specified host.
field	0..n		
AToken	0..1	String	The value is an authentication token that authorizes access to the *Multicast rendezvous service*, if required by the system operator. This may have been included in the original presentation manifest URL, it may have been added by a third-party CDN broker as part of an earlier HTTP redirect URL, or it may be generated locally by the *Application*.
MGstatus	0..1	Integer	The value is the current status of the *Multicast gateway*. 0 = inactive 1 = active
MGid	0..1	String	The value is the port number of the *Multicast gateway*, optionally preceded by its IP address. The format shall be [IP address]:port.
MGhost	0..1	String	The value is the *Multicast gateway* host name.
Ori	0..1	String	The value is the host name (FQDN) of the original targeted host. As described in clause 7.3, the *Application* may substitute the original targeted host name (FQDN) with the local *Multicast rendezvous service* host name or address. Moreover, in case of relying on a third-party CDN broker, the latter indicates here the original targeted host name (FQDN) before redirecting the request to the *Multicast rendezvous service* as explained in clause 7.4.
NOTE: URI query components are required to comply with the *query* production specified in Appendix A of [15].			

7.5.2 Operation of Multicast rendezvous service

7.5.2.0 General

After receiving the presentation manifest request from the *Content playback* function (possibly following a third-party CDN broker redirection) the *Multicast rendezvous service* shall first check that the authentication token AToken is valid (if present).

Secondly, the *Multicast rendezvous service* shall identify the appropriate *Multicast gateway* to be associated with the request, either by examining the optional MGId or MGhost request query parameters, or else by consulting its internal configuration.

The *Multicast rendezvous service* shall check whether the status of the selected *Multicast gateway* is active, either by examining the optional MGstatus request query parameter, or else by consulting its internal configuration.

If the selected *Multicast gateway* is active, the *Multicast rendezvous service* shall check whether the requested presentation manifest is associated with a multicast session in the multicast session configuration (whether currently active or not).

- If the presentation manifest is associated with a multicast session in the multicast session configuration (i.e. the service can be delivered through multicast) the *Multicast rendezvous service* shall redirect the request to the selected *Multicast gateway*, as specified in clause 7.5.2.1 below.

```
HTTP/1.1 307 Temporary Redirect
Server: <Multicast gateway>
Location: http[s]://<Multicast gateway>/<ManifestPath>
```

- If the presentation manifest is not associated with a multicast session in the multicast session configuration (i.e. the service will not be delivered through multicast) then the *Multicast rendezvous service* shall redirect the request to the *Content hosting*, either by examining the optional Ori request query parameter, or else by consulting its internal configuration.

```
HTTP/1.1 307 Temporary Redirect
Server: <Content hosting>
Location: http[s]://<Content hosting>/<ManifestPath>
```

7.5.2.1 Redirecting the request to a Multicast gateway

In the case where it accepts a request, the *Multicast rendezvous service* shall return a *307 Temporary Redirect* response. The `Location` HTTP response header shall be a URL that may include a session identifier and/or a query parameter "piggybacking" a multicast gateway configuration instance document containing the multicast session corresponding to the requested presentation manifest.

NOTE: The latter allows configuration of the *Multicast gateway* on a session-by-session basis conforming to the Just-in-time multicast gateway session configuration method (see clause 10.1.2).

The redirect URL in the `Location` response header shall comply with the following syntax:

http[s]://<*Host*>[/<*Session ID*>]/<*ManifestPath*>[?conf=<*multicast session parameters*>]

Table 7.5.2.1-1: Syntax of redirect URL elements

URL element	Use	Data type	Description
Host	1	String	The IP address or FQDN of the Multicast gateway and optionally the port number (for example "router.example:8088" or "192.0.2.1:8088").
Session ID	0..1	String	A unique presentation session identifier communicated (and possibly generated) by the *Multicast rendezvous service* comprising one or more URL path elements.
ManifestPath	1	String	The resource path for retrieving the presentation manifest from the specified host.
conf	0..1	String	The multicast session parameters shall take the form of a multicast gateway configuration instance document (see clause 10.0) comprising one multicast session per clause 10.2.2. The document shall be compressed using Gzip [8] and *base64url*-encoded according to section 5 of IETF RFC 4648 [9] prior to inclusion as a URL query string parameter.
NOTE 1: URI query components are required to comply with the *query* production specified in Appendix A of [15]. NOTE 2: Implementations of the *Multicast gateway* should be able to process long URLs.			

7.5.2.2 Refusing the request

In the case where it refuses a request, the *Multicast rendezvous service* shall respond with one of the following HTTP response status codes:

Table 7.5.2.2-1: Multicast rendezvous service error responses

HTTP status code and reason phrase	Description
401 Unauthorized	The requesting *Content playback* function is not permitted to use the *Multicast rendezvous service*, for example because the authentication token supplied in the request URL is not valid.
404 Not Found	The requested presentation manifest is not known to the *Multicast rendezvous service*.

7.6 Dynamic registration of Multicast gateway with Provisioning function

7.6.0 Introduction

In some deployments, the presence and details of *Multicast gateway* instances are not known to the *Provisioning* function in advance nor, by extension, to the *Multicast rendezvous service*. In such cases, a *Multicast gateway* may perform dynamic registration with the *Provisioning* function as a side-effect of requesting its configuration using the out-of-band pulled configuration method over reference point **C_{MR}** specified in clause 10.1.2. This dynamic registration, in turn, results in the configuration of the *Multicast rendezvous service* being updated dynamically by the *Provisioning* function.

NOTE 1: The *Multicast rendezvous service* may alternatively be configured by another means, such as during the presentation manifest request, as specified in clause 7.5.1.

To support this dynamic registration procedure, the request for a multicast gateway configuration at reference point **C_{MR}** may nominate one or more subnets (using the *content-playback-subnet* query parameter specified in clause 7.6.1) from which *Content playback* requests to the *Multicast rendezvous service* may originate, where the requesting *Multicast gateway* is a valid redirection target for *Content playback* functions in these subnets.

The *Multicast gateway* shall nominate subnets that are visible to the *Multicast rendezvous service*. It shall not nominate subnets within the private [i.14][i.15], link-local [i.16][28] or loopback [28][i.17] ranges.

NOTE 2: When a *Multicast gateway* is deployed in the home gateway device (see clause 6.2) or terminal device (see clause 6.3), it is likely to be co-located with a NAT gateway or to be deployed on the local network side of a NAT gateway. In such cases it is the responsibility of the *Multicast gateway* to discover the public subnet on which the NAT gateway is currently operating.

The request for a multicast gateway configuration may also indicate (using the *redirect-base-url* query parameter specified in clause 7.6.1) a base URL for the *Multicast rendezvous service* to use when creating the redirect target to return to the *Content playback* function.

NOTE 3: The *hostname* part of the base URL does not need to be resolvable by the *Multicast rendezvous service*, only by the *Content playback* function that subsequently receives the redirect from the *Multicast rendezvous service*. In the case of a *Multicast gateway* deployed in the home gateway device (see clause 6.2) or terminal device (see clause 6.3), the hostname may resolve to a local network address for the *Multicast gateway*.

The format of the query string that carries these optional parameters is specified in clause 7.6.1 below.

The *Multicast gateway* performs periodic requests for its multicast gateway configuration, as specified in clause 10.1.2. Should the *Multicast gateway* cease making these requests, then the *Provisioning function* shall assume that the *Multicast gateway* is no longer operating and shall remove it from the *Multicast rendezvous service* configuration.

7.6.1 Request URL format

The request URL at reference point **C_{MR}** may include a query string that carries extra information, in the form of parameter=value pairs, to facilitate the *Multicast rendezvous service* redirecting *Content playback* functions to the appropriate *Multicast gateway*. This allows dynamic *Multicast gateway* instances to register themselves with a configured *Provisioning* system at run time. The permitted query parameters are as follows:

Table 7.6.1-1: Syntax of C_{MR} query parameters

Parameter name	Use	Data type	Description
content-playback-subnet	0..*	String	A subnet of *Content playback* function IP addresses that may be presented to the *Multicast rendezvous service*, encoded as either a "dotted decimal" IPv4 subnet string [27] or as a colon-separated IPv6 subnet string [28]. If the *Multicast gateway* supports *Content playback* access from multiple different subnets, and/or if the *Multicast gateway* is "dual stack", then the query string may include multiple instances of this parameter. If omitted, the set of subnets is configured by other means.
redirect-base-url	0..1	String	Base URL to be used by *Multicast rendezvous service* when redirecting *Content playback* requests at reference point **B**. If omitted, the relocation URL is configured by other means.

All parameter values in the request URL query string shall be encoded as described in section 3.4 of RFC 3986 [15].

8 Data plane operations

8.0 Introduction

8.0.0 General

This clause describes operation of the data plane of relevant logical functions in the reference architecture, namely at reference points **P_{in}'**, **O_{in}**, **M**, **A**, **A′**, **A″** and **L**.

8.0.1 Low-latency operation (informative)

Special provision is made in the following clauses to support low-latency operation in the data plane:

- When content is pushed into a *Multicast server* at reference point **P_{in}'**, the upload of an ingest object may commence before it has been completely encoded, encrypted and packaged by the sending *Content preparation* function.
- When content is pulled by a *Multicast server* from a *Content hosting* function at reference point **O_{in}**, an ingest object may be made available for download before it has been completely received from the *Content preparation* function.
- A *Multicast server* may pass an ingest object from its *Content ingest* subfunction to its *Multicast transmission* subfunction with minimal buffering to minimize additional internal processing latency.
- If the multicast media transport protocol in use supports a low-latency transmission mode, a *Multicast transmission* subfunction may start to form multicast transport objects and begin transmitting them at reference point **M** before an ingest object has been completely ingested by the *Content ingest* subfunction.
- A *Multicast gateway* may make a playback delivery object available for download by a *Content playback* function at reference point **L** before the corresponding multicast transport object has been completely received.
- A *Multicast gateway* may make early use of unicast repair operations at reference point **A** to ensure the timely availability of playback delivery objects to a *Content playback* function at reference point **L**.

NOTE: In order to support low-latency operation, the ingest objects need to be prepared with this in mind, for example as specified in clause 4.2.9 of [10].

8.1 Operation of Content preparation function

When the multicast server configuration indicates the push content ingest method (see clause 10.2.3.4) for a particular multicast transport session, the *Content preparation* function shall deliver ingest objects to the *Multicast server* at reference point **P_{in}'** in a timely manner with respect to the presentation timeline indicated in the presentation manifest. Ingest Interface 2, described in section 6 of the DASH-IF Live Media Ingest Protocol specification [i.4] may be used at this reference point, in which case the *Content packaging* subfunction of the *Content preparation* function plays the role of "Ingest source" and the *Multicast server* plays the role of "Receiving entity".

When the multicast server configuration indicates the pull content ingest method (see clause 10.2.3.4) for a particular multicast transport session, the *Content preparation* function shall deliver ingest objects to the *Content hosting* function at reference point **P_{in}** in accordance with the DVB DASH profile [10], the HTTP Live Streaming specification [i.5] or equivalent.

With either content ingest method the *Content preparation* function may use HTTP/1.1 chunked transfer coding (as specified in section 7.1 of IETF RFC 9112 [1]) at its discretion, but especially to support MPEG-DASH [i.2] content delivery with low latency and/or in cases where the length of a media object is not known at the start of the HTTP upload transaction. When using HTTP/2 [34] or HTTP/3 [35] to support these use cases, the *Content preparation* function should deliver each part of the media object in separate *DATA* frames as and when it becomes available.

8.2 Operation of Content hosting function

The *Content hosting* function shall make media objects available for retrieval by the *Multicast server* at reference point **O_{in}** in accordance with the DVB DASH profile [10], the HTTP Live Streaming specification [i.5] or equivalent. The *Content hosting* function shall also make media objects available on an identical basis at for retrieval by the *Multicast gateway* at reference point **A**, by the *Content playback* function at reference point **A′**, and by the *Unicast repair* function at reference point **A″**. The *Content hosting* function shall support HTTP byte range requests according to section 14 of IETF RFC 9110 [2].

The *Content hosting* subfunction may serve media objects using HTTP/1.1 chunked transfer coding [1] at its discretion, but especially to support low-latency content delivery requirements of MPEG-DASH [i.2] and/or in cases where the length of a media object is not known to the *Content hosting* function when an HTTP request for it is received. When

using HTTP/2 [34] or HTTP/3 [35] to support these use cases, the *Content hosting* function should serve each part of the media object in separate *DATA* frames as and when it becomes available.

8.3 Operation of Multicast server function

8.3.0 Introduction

The primary data plane function of a *Multicast server* is to receive ingest objects at a *Content ingest* subfunction and to produce multicast transport objects from a *Multicast transmission* subfunction. The *Content ingest* subfunction shall support the reception of each ingest object in the body of an HTTP message at reference point **P_{in}'** and/or at reference point **O_{in}**. Each received ingest object shall be mapped to one or more multicast transport object(s) according to clause 8.3.3 and transmitted by the *Multicast transmission* subfunction on the associated multicast transport session at reference point **M** in accordance with clause 8.3.4.

8.3.1 Push-based content ingest

When the multicast server configuration indicates the push content ingest method (see clause 10.2.3.4) for a particular multicast transport session, the *Multicast server* shall expect ingest objects (for example, MPEG-DASH initialisation segments and media segments) to be delivered to it at reference point **P_{in}'** for example using HTTP(S) according to the specification for Ingest Interface 2 in section 6 of the DASH-IF Live Media Ingest Protocol specification [i.4]. In particular, the *Multicast server* shall implement support for HTTP/1.1 chunked transfer coding [1] to support content delivery with low latency and/or in cases where the length of a media object is not known at the start of the HTTP upload transaction. When using HTTP/2 [34] or HTTP/3 [35], the *Multicast server* shall accept ingest objects spanning multiple *DATA* frames. In order to facilitate low-latency content delivery, the *Content ingest* subfunction first needs to have acquired (via reference point **P_{in}'** or **O_{in}**) the presentation manifest associated with the parent multicast session (see clause 10.2.2.2).

In the context of [i.4] the *Content packaging* subfunction of the *Content preparation* function shall play the role of "Ingest source" and the *Content ingest* subfunction of the *Multicast server* shall play the role of "Receiving entity".

- If the presentation manifest contains relative URLs for any supplementary ingest objects, such as MPEG-DASH initialisation segments, then they shall also be pushed via reference point **P_{in}'** to the appropriate relative URLs indicated in the presentation manifest.

NOTE: The use of relative paths in the presentation manifest is recommended by clause 7.3.2 of the DASH-IF Live Media Ingest Protocol specification [i.4].

- If the presentation manifest contains absolute URLs for any supplementary ingest objects, then they shall indicate the reference point **P_{in}'** URL to which the objects are to be pushed.

The *Content packaging* subfunction may deliver ingest objects to the *Content ingest* subfunction when the corresponding multicast transport session is in either the inactive or active state, as specified in clause 10.3.0. However, the media objects shall not be forwarded by the *Multicast transmission* subfunction at reference point **M** unless the multicast transport session concerned is in the active state.

8.3.2 Pull-based content ingest

When the multicast server configuration indicates the pull content ingest method (see clause 10.2.3.4) for a particular multicast transport session, the *Multicast server* shall be responsible for retrieving ingest objects (for example, MPEG-DASH initialisation segments and media segments) using HTTP(S) via reference point **O_{in}** according to the requirements of the DVB DASH profile [10], notably its clause 10.11, or the HTTP Live Streaming specification [i.5] or equivalent. In order to facilitate this, the *Content ingest* subfunction first needs to have acquired (via reference point **P_{in}'** or **O_{in}**) the presentation manifest associated with the parent multicast session (see clause 10.2.2.2).

In the case where the presentation manifest contains absolute URLs for any ingest objects, these objects shall be available for retrieval via reference point **O_{in}**.

8.3.3 Mapping of content ingest URL to multicast transport object URI

For each ingest object received by the *Multicast ingest* subfunction, the *Multicast server* shall map the ingest object URL to one or more multicast transport object URIs. Every multicast transport object URI shall be prefixed with the multicast transport object base URI specified for the target multicast transport session, as specified in clause 10.2.3.13. An example mapping can be found in clause E.1.

According to the requirements of the multicast media transport protocol, the multicast transport object URI may be suffixed with one or more query parameters.

In chunked transmission mode (see clause 8.3.4.1 below), where an ingest object is mapped to several multicast transport objects, the multicast transport object URI may be suffixed with additional path elements and/or fragment identifiers, as required by the multicast media transport protocol.

The mapping of content ingest URL to multicast transport object URI is otherwise unspecified.

8.3.4 Emission of multicast transport objects

8.3.4.0 General

The *Multicast transmission* subfunction is responsible for formatting received ingest objects into multicast transport objects according to one (or more) multicast media transport protocols and transmitting them as multicast packets at reference point **M**. The multicast media transport protocol used on each resulting multicast transport session shall be exactly one of those specified in the annexes of the present document.

The *Multicast transmission* subfunction shall emit multicast packets for a multicast transport session only when that multicast transport session is in the active state, as specified in clause 10.3.0.

The configuration parameters for a multicast transport session shall specify at least one multicast transport endpoint address in line with clause 10.2.3.9. A multicast media transport protocol may allow more than one transport endpoint address to be associated with each multicast transport session.

The *Multicast transmission* subfunction shall use information in the ingest objects and/or the presentation manifest to transmit multicast transport objects in such a way to enable their timely reception with respect to the presentation timeline.

The *Multicast transmission* subfunction shall not exceed the maximum bit rate for the multicast transport session signalled per clause 10.2.3.10.

NOTE: This maximum does not account for packet transmission "bursting" downstream of the *Multicast server*. It merely indicates an expectation given ideal network behaviour.

8.3.4.1 Multicast transmission mode

Two multicast transmission modes are defined by this specification and the mode in use for a particular multicast transport session is signalled in the multicast transport session parameters of the multicast server configuration (see clause 10.2.3.5).

- In **resource transmission mode**, one ingest object shall be mapped to one multicast transport object.
- In **chunked transmission mode**, one ingest object may be mapped to more than one multicast transport object.

NOTE: In the case where HTTP/1.1 chunked transfer coding [1] has been used to provide the ingest object at $\mathbf{P_{in}'}$ or $\mathbf{O_{in}}$, the mapping from received HTTP chunks to multicast transport objects need not necessarily be one-to-one.

Similarly, in the case where an ingest object spans multiple HTTP/2 [34] or HTTP/3 [35] *DATA* frames, the mapping from received *DATA* frames to multicast transport objects need not necessarily be one-to-one.

The formatting of multicast transport objects in these transmission modes shall be defined separately by each multicast media transport protocol.

8.3.4.2 Forward Error Correction

A multicast media transport protocol may specify the use (optional or otherwise) of Application Level Forward Error Correction (AL-FEC) to assist with the recovery of data carried in the payloads of multicast packets lost by the network in transit. The multicast media transport protocol may specify that AL-FEC repair packets are conveyed to the same multicast group as the packets they protect, or that they are addressed to a different multicast group. In either case, the endpoint address of AL-FEC packets shall be signalled according to clause 10.2.3.11.

8.3.4.3 Marking of Random Access Points in multicast transport objects

Some multicast transport objects begin with a Random Access Point and are therefore suitable points for a *Content playback* function to begin decoding the service component. A multicast media transport protocol may specify a means of marking such multicast transport objects, either in the protocol syntax itself, or else in metadata associated with the multicast transport object, to facilitate fast stream acquisition by a *Multicast gateway* as specified in clause 8.4.3.1 below.

NOTE: This is especially useful in chunked transmission mode where a multicast transport object represents part of an ingest object.

8.3.4.4 In-band carriage of ancillary multicast transport objects

Especially in the case of unidirectional operation mode, certain media objects (including, but not limited to, the presentation manifest and MPEG-DASH initialisation segments) may be transmitted repeatedly on one or more multicast transport sessions according to a carousel schedule. For example, this may be used to facilitate fast acquisition of a linear playback session by a *Multicast gateway*. In this case, the latest version of each ingest object should be transmitted as a multicast transport object alongside other multicast transport objects in the multicast transport session. The formatting of the repeated multicast transport objects is determined by the multicast media transport protocol, and the frequency of their repetition is a matter for individual *Multicast server* implementations.

The same multicast transport objects are transmitted repeatedly in the multicast transport session until such time as they change. The same multicast transport object identifier shall be used for each repetition of an unchanged multicast transport object such that a *Multicast gateway* can ignore repeated transmissions of objects that it has already successfully acquired.

When the *Multicast server* becomes aware that an ancillary ingest object has changed, the new version shall be used to replace the current ancillary multicast transport object in the multicast transport session(s) concerned. New multicast transport object identifier(s) shall be assigned to the replacement multicast transport object(s) according to the requirements of the applicable multicast transport protocol.

- In the case of the push-based content ingest method (see clause 8.3.1), it is the responsibility of the *Content preparation* function to publish updated ancillary ingest objects to the *Multicast server* in a timely manner.
- In the case of the pull-based content ingest method (see clause 8.3.2), it is the responsibility of the *Content ingest* subfunction to acquire the replacement ancillary ingest object as soon as possible after it has been published to the *Content hosting* function.
    - The *Content ingest* subfunction may be explicitly configured to check periodically for updates to ancillary ingest objects, as specified in clause 10.2.3.14.
    - In the absence of this configuration, the *Content ingest* subfunction may instead use HTTP cache control metadata or information carried in the presentation manifest to determine the appropriate revalidation period. For example, when **`MPD/`**`@minimumUpdatePeriod` is set to 0 in a DASH media presentation description, the *Content ingest* subfunction is implicitly required to check for an updated MPD once every media segment period, i.e. before fetching every media segment.

The *Content ingest* subfunction should use conditional HTTP `GET` requests to revalidate previously acquired ancillary ingest objects efficiently when it is configured to use the pull-based content ingest method. For example, according to IETF RFC 9110 [2] the `If-None-Match` request header may be used with a previously acquired `ETag` response header value, or otherwise an `If-Modified-Since` request header may be used with a previously acquired `Last-Modified` response header value.

8.3.4.5 Integrity protection of multicast transport objects

When a multicast transport session is configured to provide an integrity check for multicast transport objects as specified in clause 10.2.3.6, the *Multicast server* shall provide an integrity check for every multicast transport object that is delivered by that multicast transport session.

Generic aspects of integrity protection are specified in clause 12.1. The format and semantics of the integrity information are specific to the multicast media transport protocol in use, as specified in annexes to the present document.

8.3.4.6 Authenticity assertion of multicast transport objects

When a multicast transport session is configured to provide an authenticity assertion for multicast transport objects as specified in clause 10.2.3.6, the *Multicast server* shall provide an authenticity assertion for every multicast transport object that is delivered by that multicast transport session.

Generic aspects of authenticity assertion are specified in clause 12.2. The format and semantics of the authenticity information are specific to the multicast media transport protocol in use, as specified in annexes to the present document.

When authenticity information is provided, the *Multicast server* shall signal which public key is to be used by recipients to verify the authenticity of the metadata associated with each multicast transport object so protected by means of the subject key identifier of an X.509 certificate [40], as specified in clause 12.2.1.6.

NOTE: X.509 certificates may be carouselled in the multicast gateway configuration transport session at reference point **M** or may be made available from a key server accessible via reference point **A**.

The provisioning of the private and public keys at the *Multicast server* is beyond the scope of the present document.

8.3.5 Multicast gateway configuration transport session

The *Multicast server* may generate a special multicast transport session at reference point **M** to convey metadata and certain types of media objects to a population of *Multicast gateway* instances. The multicast transport objects conveyed in this multicast gateway configuration transport session may include (but are not limited to):

- Multicast gateway configuration instance document as specified in clause 10.1.2 (the "in-band configuration method"). The multicast transport object URI for this multicast transport object shall be set to the following fixed value, as specified in clause B.3:

  urn:dvb:metadata:cs:MulticastTransportObjectTypeCS:2021:gateway-configuration

  If the multicast transport protocol supports the signalling of a MIME media type [14], the value specified in clause 10.2.0 shall be used.

- Presentation manifest media objects. The multicast transport object URI of each presentation manifest shall be determined by the value of the **`PresentationManifestLocator`**`@transportObjectURI` attribute, if present in the multicast server configuration (see clause 10.2.2.2). If omitted, the *Multicast server* shall nominate a compliant transport object URI for the presentation manifest. The attribute shall always be present in multicast gateway configuration instance documents conveyed as specified in this clause to enable *Multicast gateway* instances to associate presentation manifests received at reference point **M** with the corresponding service component(s).

NOTE 1: In the case where a presentation manifest is conveyed in a multicast gateway configuration transport session, the *Multicast server* may manipulate the presentation timeline to compensate for the additional delay introduced by multicast transmission and/or resulting from downstream repair operations.

- Initialization segment media objects for the Representations described by MPEG-DASH presentation manifests (Media Presentation Description documents) [i.2].

- X.509 certificates [39] used for the purpose of asserting the authenticity of multicast transport objects in any multicast transport session emitted by the *Multicast server* as specified in clause 8.3.4.6, or as part of a chain of trust for another X.509 certificate used for that purpose.

The latest version of each multicast transport object should be transmitted repeatedly on the multicast gateway configuration transport session in order to facilitate their rapid acquisition by a *Multicast gateway*.

NOTE 2: The carousel repetition rates used for different types of multicast transport object on the multicast gateway configuration transport session are a matter for individual *Multicast server* implementations.

If a multicast gateway configuration transport session is advertised as being available in a deployed system, the *Multicast gateway* should subscribe to it and may cache the latest version of any or all the multicast transport objects it conveys.

8.4 Operation of Multicast gateway function

8.4.0 Introduction

The main purpose of a *Multicast gateway* is to convert multicast transport objects received at reference point **M** into playback delivery objects, as specified in clause 8.4.3 below. In order to do this, the multicast transport URI is mapped to a playback delivery URL, as specified in clause 8.4.4. The presentation manifest may need to be manipulated to achieve these aims, and this is specified in clause 8.4.1.

The *Multicast gateway* exposes playback delivery objects at reference point **L** (specified in clause 8.4.5) in order to service requests from the *Content playback* function.

8.4.1 Handling of presentation manifest

8.4.1.0 General

When a *Multicast gateway* receives an HTTP request for a presentation manifest from the *Content playback* function at reference point **L** it may need to acquire a copy of the corresponding HTTP resource if it does not already have a valid, unexpired copy cached locally, either from a multicast transport session (via reference point **M**, see clause 8.4.2) or from the multicast gateway configuration transport session (via reference point **M**, see clause 8.3.5) or from the *Content hosting* function (via reference point **A**). The *Multicast gateway* shall use the content playback path pattern specified in clause 10.2.2.2, if present in the multicast gateway configuration, to match the requested presentation manifest URL at reference point **L** with a multicast session present in the current multicast gateway configuration.

NOTE 1: In unidirectional operation mode where the *Multicast gateway* receives the presentation manifest only through reference point **M**, in the case of a cache miss (e.g. the *Multicast gateway* has just booted) the *Multicast gateway* should stall the relevant HTTP request at reference point **L** until the presentation manifest arrives.

It is an error for the *Content playback* function to request a presentation manifest from the *Multicast gateway* that is not present in the current multicast gateway configuration. In this case the *Multicast gateway* shall return the HTTP error code *404 (Not Found)*.

During a multicast session, the presentation manifest may be updated. The *Multicast gateway* should keep a copy of the presentation manifest in its *Asset storage* so that it can be served efficiently to the *Content playback* function on request.

The *Multicast gateway* may manipulate the presentation manifest so acquired prior to returning it to the *Content playback* function.

- The *Multicast gateway* should take any necessary steps to ensure that any HTTP redirection from the *Multicast rendezvous service* is "sticky" for the duration of the presentation session, such that the *Content playback* function updates its presentation manifest directly from the *Multicast gateway* rather than returning to the *Multicast rendezvous service*. This is further elaborated in clause 8.4.1.1 below.

- Where a service component described in the presentation manifest corresponds to an active multicast transport session in the current multicast gateway configuration and its URL (or its base URL) is absolute [15], the host part of its URL (or that of its base URL) shall be adjusted to point at the reference point **L** address of the *Multicast gateway* in accordance with clause 8.4.4 below. Clause E.4 provides an example of how the playback delivery URL is modified.

- Where a service component described in the presentation manifest corresponds to an active multicast transport session in the current multicast gateway configuration, a presentation session identifier may be inserted into its URL (or into its base URL) to allow the *Multicast gateway* to associate subsequent requests for playback delivery objects with the presentation manifest request.

NOTE 2: This may be useful in monitoring playback session performance separately for each *Content playback* instance.

- Where any service component referenced by the presentation manifest corresponds to an active multicast transport session in the current multicast gateway configuration, the presentation timeline signalled in the presentation manifest may be adjusted to allow additional time for Forward Error Correction and/or unicast repair operations (see clause 9) to be performed. This is further elaborated in clause 8.4.1.1 below.

NOTE 3: FEC repair can only be applied after all symbols (source and redundancy) for a source block have been received. The presentation timeline should therefore be delayed at the least by a period equivalent to the transmission time of one source block at the multicast transport session bit rate (signalled according to clause 10.2.3.10) plus the maximum time taken to effect the repair of one source block.

- Where the presentation manifest delivered at reference point **A** references media objects not yet transmitted over reference point **M**, the presentation timeline signalled in the presentation manifest should be adjusted to prevent requests by the *Media player* at reference point **L** for media objects that are not yet available in the *Multicast gateway* as delivery objects. This is further elaborated for the case of MPEG-DASH in clause 8.4.1.1 below.

NOTE 4: Modifying the presentation timeline in this way reduces the number of *Multicast gateway* cache misses and, hence, the number of unicast requests made at reference point **A**.

8.4.1.1 MPEG-DASH presentation manifest manipulation (informative)

The *Multicast gateway* may reduce the value of **MPD/**`@timeShiftBufferDepth` in the presentation manifest if it exceeds the capacity of its internal cache.

To ensure that any HTTP redirection from the *Multicast rendezvous service* is "sticky" for the duration of the presentation session (as described in clause 8.4.1.0 above), the *Multicast gateway* should replace any **MPD/Location** elements in the Media Presentation Description with new elements indicating the reference point **L** URL(s) of the presentation manifest. The *Multicast gateway* may add one or more **MPD/Location** elements to the presentation manifest if none are present.

NOTE 1: An MPEG-DASH Media Presentation Description may contain multiple **MPD/Location** elements according to clause A.11 of [i.2]. This may be useful in deployments where there is more than one *Multicast gateway* available to the *Content playback* function.

NOTE 2: The use of the **MPD/Location** element by the *Content playback* function is specified in clause 10.11 of ETSI TS 103 285.

NOTE 3: In the absence of any **MPD/Location** elements, the *Content playback* function should follow the MPD Location and Reference Resolution provisions specified in clause 3.2.15 of [i.12].

Depending on the value of the relevant **MulticastSession/**`@contentPlaybackAvailabilityOffset` attribute in the multicast gateway configuration (see clause 10.2.2.0), the *Multicast gateway* may need to modify the presentation timeline as follows:

- In the case of an MPEG-DASH Media Presentation Description with **SegmentTemplate** using *$Time$*, when receiving an updated presentation manifest either through reference point **A** or through reference point **M**, the *Multicast gateway* modifies the presentation manifest exposed over reference point **L** by removing the media segments it has not yet received over reference point **M**.

- In the case of an MPEG-DASH Media Presentation Description with **`SegmentTemplate`** using *$Number$*, the *Multicast gateway* may adjust the value of **`MPD/`**`@availabilityStartTime` to accommodate any latency introduced by the system.

NOTE 4: Modifying the presentation timeline in this way may reduce the usable cache in the *Multicast gateway*, necessitating a further reduction of **`MPD/`**`@timeShiftBufferDepth`.

8.4.2 Acquisition of multicast transport objects

8.4.2.0 General

A *Multicast gateway* may begin acquiring multicast transport objects from a multicast transport session before or after a request for a presentation manifest has been received from a *Content playback* function at reference point **L**.

To start acquiring multicast transport objects from a particular multicast transport session in the current multicast session configuration, the *Multicast reception* subfunction shall subscribe to the IP multicast group(s) indicated in the multicast transport endpoint address(es), as specified in clause 10.2.3.9.

A *Multicast gateway* may subscribe to a multicast transport session before that session is in the active state (see clause 10.3.0). A *Multicast gateway* should unsubscribe from a multicast transport session when that session enters the inactive state (see clause 10.3.0).

In the case where multicast packet loss is encountered by the *Multicast reception* subfunction, it may employ Forward Error Correction techniques (if signalled according to clause 10.2.3.11) and/or unicast repair as specified in clause 9 (if signalled according to clause 10.2.3.12) to recover the missing data.

A *Multicast reception* subfunction should verify the integrity and authenticity of received multicast transport objects if the availability of this metadata in the multicast transport session is signalled according to clause 10.2.3.6. Generic aspects of integrity protection and authenticity assertion are specified in clauses 12.1 and 12.2 respectively; integrity and authenticity provisions for individual multicast transport protocols are specified in annexes to the present document. The provisioning of the public key at the *Multicast gateway* used to verify authenticity is beyond the scope of the present document.

NOTE: The public key may, for example, be provisioned via reference point **C_{MR}**, or it may be available from the multicast gateway configuration transport session at reference point **M**, or it may be retrieved from a key server accessible via reference point **A**.

Unicast repair may be employed as described in the previous paragraph to acquire the corresponding media object in the case that either the integrity check or the authenticity check fails.

8.4.2.1 Acquisition of DASH initialisation segments

When a *Multicast gateway* receives an HTTP request for an initialisation segment from the *Content playback* function at reference point **L** it may need to acquire a copy of the corresponding HTTP resource if it does not already have a valid, unexpired copy cached locally. Initialisation segments shall be made available from one or more of the following sources:

- At reference point **M** from a multicast transport session (see clause 8.4.2.0 above).
- At reference point **M** from the multicast gateway configuration transport session (see clause 8.3.5).
- At reference point **A** from the *Content hosting* function (see clause 9).

NOTE: In unidirectional operation mode, where the *Multicast gateway* receives the initialisation segments only through reference point **M**, in the case of a cache miss (e.g. the *Multicast gateway* has just booted) the *Multicast gateway* should stall the relevant HTTP request at reference point **L** until the initialisation segments arrive.

During a multicast session, the initialisation segments may be updated. The *Multicast gateway* should keep a copy of the initialisation segments in its *Asset storage* so that they can be served efficiently to the *Content playback* function on request.

8.4.3 Construction of playback delivery objects

8.4.3.0 General

A *Multicast gateway* shall attempt to reconstruct playback delivery objects from unicast acquisition at reference point **A** and/or the multicast data packets it receives at reference point **M** according to the specification of the multicast media transport protocol signalled in the multicast media transport protocol parameters (see clause 10.2.3.8).

If chunked transmission mode has been specified for a multicast transport session, the *Multicast gateway* shall be responsible for recombining a set of multicast transport objects that comprise the media object into a single playback delivery object at reference point **L**. The recombination recipe (if any) shall be specified separately by each multicast media transport protocol.

Playback delivery objects whose integrity cannot be established by the *Multicast gateway* per clause 8.4.2 above should not be served at reference point **L**.

A *Multicast gateway* may cache the playback delivery objects that it has successfully reconstructed in its *Asset storage* subfunction.

8.4.3.1 Fast acquisition of playback session (informative)

To facilitate fast acquisition of a multicast transport session by a *Multicast gateway*, a *Multicast transmission* subfunction may mark some multicast transport objects with a Random Access Point marker, as specified in clause 8.3.4.3 above. In such cases, the *Multicast gateway* may construct a partial playback delivery object and deliver this at reference point **L**.

NOTE 1: This is particularly useful in chunked transmission mode (clause 8.3.4.1) where a *Multicast gateway* joins a multicast transport session part-way through transmission of a media object. It can wait for the start of the next multicast transport object with a Random Access Point marker and can use this to reconstruct a partial playback delivery object.

NOTE 2: It may be necessary for the *Multicast gateway* to manipulate the presentation manifest to preserve the timing model of the presentation, for example by indicating that the first segment delivered at reference point **L** is shorter than usual.

Alternatively, at the start of the presentation session, a *Multicast gateway* may fetch one or more partial or complete media objects via unicast at reference point **A**.

8.4.4 Mapping of multicast transport object URI to playback delivery object URL

For every service component listed in the presentation manifest that corresponds to a multicast transport session in the current multicast gateway configuration (whether currently active or not), the *Multicast gateway* shall expose a unique playback delivery object URL at reference point **L** for each media object available on that service component in each presentation session. The playback delivery object URL shall be the same irrespective of whether the underlying media object is acquired from an active multicast transport session at reference point **M**, from a unicast repair operation at reference point **A**, or any combination of these sources.

NOTE: At any instant in time, the *Multicast gateway* may have available for download at reference point **L** playback delivery objects in a sliding window that corresponds to a timeshift buffer indicated by the presentation manifest. However, a notional mapping exists for all media objects on relevant service components: past, present and future.

The format of the playback delivery object URL is at the discretion of the *Multicast gateway* implementation, but it may be algorithmically derived from the multicast transport object URI. Example mappings can be found in clause E.4.

In the case where the *Multicast gateway* is deployed in a terminal device (see clause 6.3), the form of the playback delivery object URL is at the discretion of the implementation.

If chunked transmission mode has been specified for a multicast transport session with each ingest object divided into several multicast transport objects, the *Multicast gateway* shall expose a single playback delivery object URL for each media object at reference point **L** that is consistent with the presentation manifest previously delivered.

Any presentation manifest served by the *Multicast gateway* shall be adjusted to reflect the structure of the playback object URL(s) chosen, in accordance with clause 8.4.1 above.

8.4.5 Serving of playback delivery objects

When a *Multicast gateway* receives an HTTP request for a playback delivery object from the *Content playback* function at reference point **L**, it shall ensure that the request corresponds to a service declared in the current multicast gateway configuration document.

NOTE 1: The means to determine which service a playback delivery object is a component of is implementation-dependent.

It is an error for the *Content playback* function to request a playback delivery object from the *Multicast gateway* that is not a component of a service present in the multicast gateway configuration. In this case, the *Multicast gateway* shall return the HTTP error code *404 (Not Found)*.

Where playback delivery objects acquired by the *Multicast gateway* are made available at reference point **L** in accordance with the DVB DASH profile [10] (notably its clause 10.11):

- The *Multicast gateway* is required to implement support for HTTP byte range requests [2] at the above reference point.
- The *Multicast gateway* is required to implement support for HTTP/1.1 chunked transfer coding [1] at the above reference point and may use it at its discretion, but especially to support content delivery with low latency and/or cases where the length of a playback delivery object is not known at the start of an HTTP download transaction.

NOTE 2: Where HTTP/1.1 chunked transfer coding is employed at reference point **L**, the arrangement of chunks in the *chunked-body* (see section 7.1 of IETF RFC 9112 [1]) need not have the same structure as those ingested by the *Content ingest* function at reference points $\mathbf{P_{in}'}$ or $\mathbf{O_{in}}$, nor need the HTTP chunks served at reference point **L** align with the boundaries of the multicast transport objects received at reference point **M** nor the boundaries of media objects retrieved at reference point **A**.

Likewise, when HTTP/2 [34] or HTTP/3 [35] are employed at reference point **L**, the boundaries of the *DATA* frames served may differ from those of the corresponding ingest objects at reference points $\mathbf{P_{in}'}$ or $\mathbf{O_{in}}$, the corresponding multicast transport objects received at reference point **M** or any media object repairs at reference point **A**.

Playback delivery objects may also be made available in accordance with the HTTP Live Streaming specification [i.5] or equivalent.

The *Multicast gateway* may support the Transport Layer Security protocol [7] at reference point **L**.

If chunked transmission mode has been specified for a multicast transport session and low-latency operation is specified in the presentation manifest, the *Multicast gateway* should make the playback delivery object available for download as soon as the playback object URL has been determined. However, the HTTP message body shall not be served until any necessary Forward Error Correction and/or unicast repair operations (see clause 9) have been completed.

8.5 Operation of Content playback function

The purpose of a *Content playback* function is to retrieve and subsequently render media objects from the *Content hosting* function (via reference point **B**), some of which may be redirected to a *Rendezvous service* (also reference point **B**) and/or a *Multicast gateway* (via reference point **L**).

Where these interactions comply with the DVB DASH profile [10] (notably its clause 10.11):

- The *Content playback* function is required to implement support for HTTP byte range requests [2] at the above reference points.
- The *Content playback* function is required to implement support for HTTP/1.1 chunked transfer coding [1] at the above reference points to support content retrieval with low latency and/or cases where the length of a playback delivery object is not known at the start of the HTTP download transaction.

Alternatively, the *Content playback* function may comply with the HTTP Live Streaming specification [i.5] or equivalent.

9 Unicast repair

9.0 Introduction

In cases where a multicast transport object is not received intact by the *Multicast reception* subfunction by the required deadline and the multicast packet loss could not be repaired by the *Multicast gateway* using Forward Error Correction, repair of the media object should be attempted using one of the unicast repair protocols specified in this clause. For example:

1) Multicast transport packets are received by the *Multicast reception* subfunction, but they are in some way corrupt or unusable.
2) Multicast transport packets do not arrive in time to be usable.
3) No multicast transport packets have been received for the media object in question.

An HTTP-based repair protocol is specified in clause 9.2. Unicast repair at reference point **U** is not specified in this clause.

The unicast repair procedure shall not be used in case of unidirectional operation.

If Forward Error Correction is provided as part of a multicast transport session and is used by the *Multicast gateway*, the *Multicast gateway* should wait until the arrival of the last packet of an FEC block including repair symbols, and should attempt to recover the lost packets using these repair symbols, before triggering the unicast repair procedure.

9.1 Triggering unicast repair

The unicast repair procedure may be triggered by the *Multicast gateway* when any of the following conditions occur:

- As soon as packet loss is detected, in the case where no Forward Error Correction is provided as part of the multicast transport session, or when FEC is not used by the *Multicast gateway*.
- If no multicast transport packets have been received within the period of time indicated by the transport object reception timeout specified in clause 10.2.3.12.
- If a timeout timer specified by the multicast media transport protocol expires.
- In order to ensure that a playback delivery object is made available at reference point **L** in a timely manner with respect to the timing constraints of the presentation manifest as modified by the *Multicast server* and/or by the *Multicast gateway*.
- The end of a multicast session (as specified in clause 10.2.3.3) is reached.

The multicast media transport protocol may specify additional requirements or conditions for when the unicast repair procedure is triggered.

Once a unicast repair operation has been triggered, the *Multicast gateway* delays the unicast repair procedure as follows:

- If a fixed back-off period is indicated in the unicast repair parameters for the multicast transport session (see clause 10.2.3.12), the *Multicast gateway* shall delay the unicast repair procedure by this period of time.
- If a random back-off period is indicated in the unicast repair parameters for the multicast transport session (see clause 10.2.3.12), the *Multicast gateway* shall delay the unicast repair procedure by a randomly chosen period of time between 0 and the random back-off period.

NOTE: For a linear service that requires low-latency delivery of media objects, the values of the fixed back-off period and random back-off period should be chosen with care.

For example, if Forward Error Correction is not used in a multicast transport session or FEC is not used by the *Multicast gateway*, the unicast repair procedure will start at the latest:

min{transportObjectReceptionTimeout, multicast media transport protocol-specific object timeout timer}
+ fixedBackOffPeriod + randomBackOffPeriod

9.2 HTTP-based repair protocol

9.2.0 General

The *Multicast gateway* and the *Content hosting* function shall support the use of HTTP/1.1 [1] as the unicast repair protocol at reference point **A**. The *Multicast gateway* and the *Content hosting* function may additionally support the use of HTTP/2 as specified in IETF RFC 9113 [34] and/or HTTP/3 as specified in IETF RFC 9114 [35] as the unicast repair protocol at this reference point. The *Multicast gateway* and the *Content hosting* may use HTTPS for HTTP/1.1 or HTTP/2; the use of HTTPS is mandatory for HTTP/3.

Irrespective of the HTTP protocol version used, the *Multicast gateway* and the *Content hosting* function shall support the use of byte range requests as specified in IETF RFC 9110 [2].

When the individual gaps in the received media object can be identified by the *Multicast gateway*, it should make an HTTP byte range request containing one or more byte ranges. The request and response messages are formatted as specified in clause 9.2.4 below.

If multiple unicast repair endpoints are specified in the multicast session parameters as described in clause 10.2.3.13, the *Multicast gateway* shall choose one of them to send the byte range request(s). If this request is unsuccessful, the *Multicast gateway* may retry the repair procedure using one or more of the remaining unicast repair endpoints.

9.2.1 Selection of unicast repair base URL

Where more than one unicast repair base URL prefix is indicated for a multicast transport session, the *Multicast gateway* shall select exactly one of these to perform the unicast repair operation. The basis on which this selection is made is implementation-specific. For example, the following algorithm describes one possible implementation:

1) Place the unicast repair base URLs declared for the multicast transport session in an ordered list indexed by consecutive positive integers.
2) Sum the non-zero relative weights of all the **`UnicastRepairParameters/BaseURL`** elements.
3) Generate a random integer between 1 and the sum calculated in the previous step and use it to index the list constructed in step 1.

In the below example, three unicast repair base URLs are declared with relative weights of 3, 5 and 2 respectively:

```
<UnicastRepairParameters [...]>
  <BaseURL relativeWeight="3">http://cdn1.example.com/content/</BaseURL>
  <BaseURL relativeWeight="5">http://cdn2.example.com/content/</BaseURL>
  <BaseURL relativeWeight="2">http://cdn3.example.com/content/</BaseURL>
```

```
</UnicastRepairParameters>
```

If a random value of 6 is calculated, for example, then the second unicast repair base URL (*cdn2*) is selected, as shown in figure 9.2.1-1:

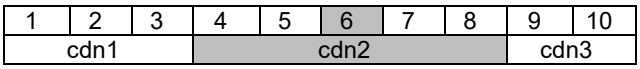


Figure 9.2.1-1: Unicast repair parameters example

9.2.2 Mapping of multicast transport object URI to unicast repair URL

In order to construct the unicast repair URL for a given multicast transport object, the *Multicast gateway* shall perform the following operations on the multicast transport object URI:

1. Remove the prefix (if any) indicated by the multicast transport object base URI specified in clause 10.2.3.13.
2. Prepend the unicast repair base URL prefix (if any) selected in clause 9.2.1 above.

An example can be found in clause E.2.

NOTE: HTTPS may be used in the constructed unicast repair URL.

9.2.3 Construction of unicast request URL when no object metadata has been received

When no object metadata has been received for a media object, the *Multicast gateway* shall construct a URL for the purpose of requesting the missing object based on the relevant presentation manifest acquired at reference point **A** or **M**. The unicast request URL shall be of a form that can be addressed to the *Content hosting* function – either directly at reference point **A** or else (as a result of redirecting the *Content playback* function per section 15.4 of [2]) at reference point **A′** – that is consistent with the requirements of the presentation manifest.

9.2.4 Message format

To request a complete media object, the *Multicast gateway* should use a conventional HTTP `GET` request. The request URL shall be constructed according to clause 9.2.2 or clause 9.2.3, as appropriate.

To request the missing portion(s) of the media object, the *Multicast gateway* should use the partial HTTP `GET` request with the `Range` request header as described in section 14 of IETF RFC 9110 [2]. To improve efficiency, the *Multicast gateway* may use a single partial HTTP `GET` message to request multiple byte ranges.

If missing data is identified in multiple multicast transport objects, the *Multicast gateway* should use separate HTTP `GET` requests (partial or otherwise) to repair each object separately:

- If there is an entity tag (e.g. `Etag` response header) present in the metadata for the media object, its value shall be used in the `If-Range` header of the conditional HTTP `GET` request as described in section 13.1.5 of IETF RFC 9110 [2]. Strong comparison shall be applied when comparing entity tags as described in section 8.8.3.2 of IETF RFC 9110 [2].
- Otherwise, the *Multicast gateway* shall send a request message without the `If-Range` header.

The HTTP response message shall be formatted as described in the section 15.3.6 of IETF RFC 9110 [2]. The *Content hosting* function may redirect the request to another server.

- The integrity of the response message may be protected using HTTP digest fields (i.e. `Content-Digest` and/or `Repr-Digest` fields) as specified in RFC 9350 [20]. If these are present, the *Multicast gateway* should validate them before further processing the payload of the response message.
- The authenticity of the HTTP fields in the response message (including any HTTP digest fields) may be asserted using HTTP message signatures (i.e. `Signature-Input` and `Signature` fields) as specified in RFC 9421 [38]. If these are present, the *Multicast gateway* should validate them before further processing the payload of the response message.

Example request and response messages can be found in clause E.3.

10 Multicast session configuration

10.0 Introduction

The *Multicast server* and the *Multicast gateway* need to have a common view on the set of multicast transport sessions that are currently defined in a deployed system so that the latter knows which multicast transport sessions it can subscribe to, as specified in clause 8.4.2. The concrete embodiment of this logical state is referred to in the present specification as the **multicast session configuration** and is realized as an XML instance document - the **multicast session configuration instance document** - whose data model is specified in clause 10.2 below and whose schemas are specified in annex A:

- In the case of the *Multicast server*, the current configuration is referred to as the **multicast server configuration** and is embodied in a **multicast server configuration instance document**. An example can be found in clause C.1.
- For the *Multicast gateway*, the current configuration is the **multicast gateway configuration** and is embodied in a **multicast gateway configuration instance document**. An example can be found in clause C.3.

Further worked examples can be found in clause 5 of [i.13].

Certain configuration parameters of interest to the *Multicast gateway* may be included in the multicast server configuration but not interpreted by the *Multicast server*. These are intended to be transmitted to the *Multicast gateway* in a multicast gateway configuration transport session (see the "In-band configuration method" in clause 10.1.2 below).

10.1 Control system arrangement

10.1.1 Configuration of Multicast server

A *Multicast server* shall be configured using exactly one of the two following methods:

1) **Out-of-band pushed configuration method** whereby the *Network control* function delivers a multicast server configuration instance document instance to the *Multicast server* across reference point **C_{MS}**. The procedure for this method is specified in clause 10.4.2.1.

2) **Out-of-band pulled configuration method** whereby the *Multicast server* periodically polls the *Network control* function for the current multicast server configuration document instance across reference point **C_{MS}**. The procedure for this method is specified in clause 10.4.2.2.

10.1.2 Configuration of Multicast gateway

In a given deployed system, all *Multicast gateway* instances shall be configured using one or more of following methods:

1) **Out-of-band pushed configuration method** whereby the *Network control* function delivers a multicast gateway configuration document instance across reference point **C_{MR}**. The procedure for this method is specified in clause 10.4.4.1.

2) **Out-of-band pulled configuration method** whereby the *Multicast gateway* periodically polls the *Network control* function for the current multicast gateway configuration document instance across reference point **C_{MR}**. The procedure for this method is specified in clause 10.4.4.2.

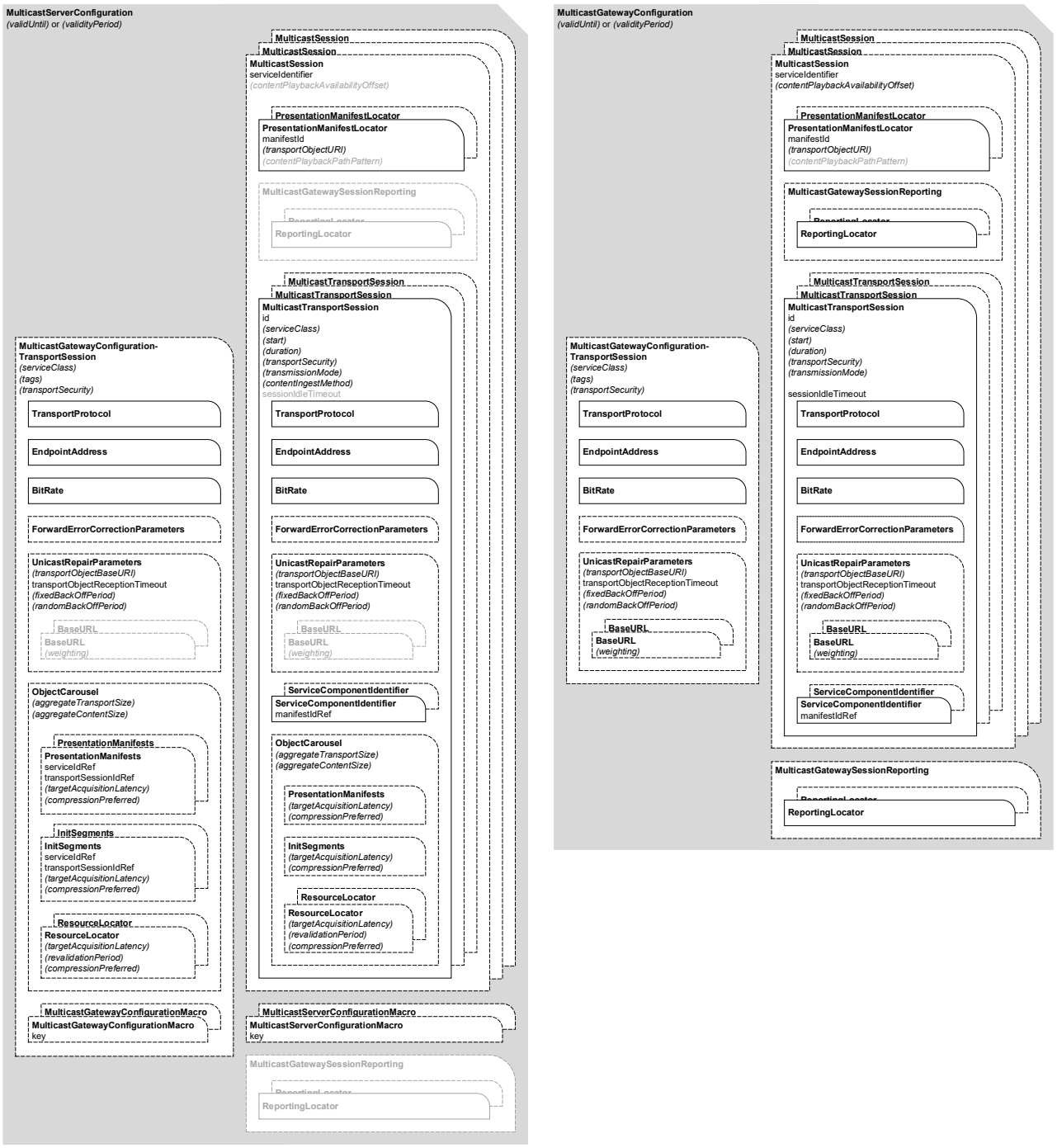
3) **In-band configuration method** whereby the *Multicast server* carousels the current multicast gateway configuration instance document instance via a configured multicast gateway configuration transport session at reference point **M** in accordance with clause 8.3.5. In this case the *Multicast server* shall be responsible for generating the multicast gateway configuration instance document based on its current multicast server configuration received over reference point **C_{MS}**. The procedure for this method is specified in clause 10.4.5.

NOTE: If the *Multicast rendezvous service* is co-located with the *Multicast gateway* (this is mandatory in case of unidirectional operation) then it can consume the multicast gateway configuration instance document and can therefore be configured through the in-band configuration method.

4) **Just-in-time configuration method** whereby the *Multicast rendezvous service* includes a single set of multicast session parameters in its redirect response to a *Content playback* function when a presentation manifest is requested via reference point **B**. The procedure for this method is specified in clause 7.5.

10.2 Multicast session configuration instance document data model

10.2.0 Overview



NOTE: Only principal elements and selected attributes are depicted. Dotted lines indicate optional elements. Parenthesised italics denote optional attributes. Items in grey are passed through from the multicast server configuration to the multicast gateway configuration but are not used by the *Multicast server*.

Figure 10.2-1: Overview of the multicast server configuration instance document (left) and multicast gateway configuration instance document (right)

Multicast session configuration instance documents shall be signalled using the following MIME media type [14]:

application/xml+dvb-mabr-session-configuration

10.2.1 Document root element

10.2.1.0 General

A multicast session configuration instance document describes an intended configuration of multicast sessions and multicast transport sessions in a deployed system.

Multicast session configuration instance documents shall comply with the XML schema definitions in annex A. Documents complying with the present version of this specification shall declare the following baseline XML schema name space URI:

urn:dvb:metadata:MulticastSessionConfiguration:2024

The root element for a multicast server configuration instance document shall be **`MulticastServerConfiguration`** as specified in clause 10.2.1.1 below.

The root element for a multicast gateway configuration instance document shall be **`MulticastGatewayConfiguration`** as specified in clause 10.2.1.2 below.

A multicast session configuration instance document shall set the `@schemaVersion` attribute of the root element to the value specified in clause A.0.

In system deployments employing the *Multicast gateway* in-band configuration method (see clause 10.1.2 above) the multicast session configuration shall specify one or more multicast gateway configuration transport sessions according to clause 10.2.5 below. When this configuration method is used, one or more **`MulticastServerConfigurationMacro`** elements may be present in a multicast server configuration instance document, as specified in clause 10.2.1.1 below, allowing the *Multicast server* to generate and transmit multiple variant multicast gateway configurations. The usage of this element is further specified in clause 10.2.5.2.

A multicast session configuration instance document shall specify zero or more multicast sessions in the system according to clause 10.2.2 below.

Multicast session configuration instance documents may have an associated validity expressed as either:

1) A fixed period of time since the document was received, using the `@validityPeriod` attribute of the document root element. The value of this attribute shall comply with clause 5.5.2 of ISO 8601-1 [11], but excluding the alternative format specified in clause 5.5.2.4 of [11]. After this period has elapsed the document and its contents shall be deemed to have expired.

2) As an absolute point in time, using the `@validUntil` attribute of the document root element. The value of this attribute shall comply with the *TimePoint* datatype specified in clause 6.4.3 of MPEG-7 Part 5 [12]. At this point in time the document and its contents shall by deemed to have expired.

NOTE 1: The MPEG-7 *TimePoint* datatype is a restricted subset of ISO 8601-1 clause 5.4 [11].

NOTE 2: It is not possible to indicate the use of Coordinated Universal Time (UTC) using the UTC designator *Z* with this datatype. Per clause 6.4.3 of MPEG-7 Part 5 [12], if no time zone is specified a time zone of 00:00 is assumed rather than local time.

A multicast session configuration instance document that includes both of the above validity attributes shall be deemed to expire at whichever indicates the later expiry.

If neither attribute is specified, the expiry of a received multicast session configuration instance document may be determined by the use of other metadata, such as HTTP `Cache-Control` or `Expires` response headers [5].

In any case, the multicast session configuration contained in the instance document shall be deemed to be null and void after the point of expiry, and the recipient should discard that configuration.

A multicast gateway configuration instance document carouselled in a multicast gateway configuration transport session at reference point **M** (see clause 8.3.5) shall not include the `@validityPeriod` attribute.

A *Multicast gateway* may be configured to report metrics for all configured multicast sessions to one or more external *Service reporting capture* subfunctions.

NOTE 1: Reporting on an individual multicast session may be declared independently of reporting for all multicast sessions (see clause 10.2.2.3) and the two levels of reporting may therefore operate in parallel.

NOTE 2: The target *Service reporting capture* subfunctions may be operated by different business entities.

Reporting instance documents are periodically submitted to the declared set of *Service reporting capture* subfunctions only by a randomly selected sample of *Multicast gateway* instances in the deployed system. Each *Multicast gateway* determines whether it will send reports to a given *Service reporting capture* subfunction by independent random choice, weighted according to a configurable sample proportion.

One or more reporting endpoints shall be declared inside the **`MulticastGatewaySessionReporting`** child element of **`MulticastSession`**, each endpoint being specified as a URL in the content of a separate **`ReportingLocator`** element. The use of this endpoint locator is specified in clause 11.0:

- The `@proportion` attribute should be used to indicate what sample of *Multicast gateway* instances in the deployed system report to the endpoint specified in the enclosing **`ReportingLocator`** element. Omitting this attribute indicates that all *Multicast gateway* instances should report (corresponding to a proportion of 1.0).
- The `@period` attribute shall be used to indicate the time gap between consecutive reports being submitted by the *Multicast gateway*. The value 0 indicates that periodic reporting is disabled, in which case reporting occurs whenever an event is to be reported as defined in clause 11.2.2.
- The `@randomDelay` attribute shall be used by the *Multicast gateway* as an additional delay after the above time gap.
- The `@reportSessionRunningEvents` flag controls whether Playback session running events (see clause 11.2.2.2) are to be included in the reporting document instance.

When the *Multicast gateway* in-band configuration method is used at reference point **M** (per clauses 8.3.5 and 10.2.5) the **`MulticastGatewaySessionReporting`** element and all its children shall be copied from the multicast server configuration instance document into the corresponding multicast session of the multicast gateway configuration instance document. For other *Multicast gateway* configuration methods, the **`MulticastGatewaySessionReporting`** element should be omitted from the multicast server configuration instance document.

10.2.1.1 MulticastServerConfiguration root element

The syntax of the **`MulticastServerConfiguration`** element (the root element of a multicast server configuration instance document) is specified in table 10.2.1.1-1 below.

Table 10.2.1.1-1: MulticastServerConfiguration element syntax

Element or attribute name	Use	Data type	Clause	Description
`MulticastServerConfiguration`			10.2.1.0	Root element of the multicast server configuration instance document.
`@schemaVersion`	1	Unsigned Integer	10.2.1.0	Schema version number for this multicast server configuration instance document, as specified in clause A.0.
`@validityPeriod`	0..1	Duration	10.2.1.0	Time period after receipt for which this instance document is valid. If this attribute is present, then `@validUntil` should not be present.
`@validUntil`	0..1	dateTime	10.2.1.0	Deadline after which this instance document ceases to be valid. If this attribute is present, then `@validityPeriod` should not be present.
`MulticastGatewayConfiguration TransportSession`	0..n		10.2.5, 8.3.5	Container for multicast gateway configuration transport session parameters.
`MulticastSession`	0..n		10.2.2	Container for multicast session parameters.
`MulticastServerConfiguration Macro`	0..n	String	10.2.1.0	The value to substitute in place of any matched macro keys found when processing the multicast server configuration.
`@key`	1	Name Token string	10.2.1.0	Macro key to be matched within the multicast server configuration.
`MulticastGatewaySession Reporting`	0..1		10.2.1.0	Container for reporting parameters.
`ReportingLocator`	1..n	URI String	10.2.1.0	Container for a multicast gateway reporting endpoint.
`@proportion`	0..1	Decimal	10.2.1.0	Proportion of *Multicast gateway* instances that should send session reports to the specified endpoint, expressed as a decimal value between 0.0 and 1.0.
`@period`	1	Duration String	10.2.1.0	Session reporting periodicity expressed in seconds. The value 0 disables periodic reporting and reports are then sent only when significant events occur.
`@randomDelay`	1	Unsigned Integer	10.2.1.0	An additional random period that a *Multicast gateway* should delay between sending playback session reports.
`@reportSessionRunningEvents`	0..1	Boolean	10.2.1.0	Determines whether Playback session running events are reported.

10.2.1.2 MulticastGatewayConfiguration root element

The syntax of the **`MulticastGatewayConfiguration`** element (the root element of a multicast gateway configuration instance document) is specified in table 10.2.1.2-1 below.

Table 10.2.1.2-1: MulticastGatewayConfiguration element syntax

Element or attribute name	Use	Data type	Clause	Description
`MulticastGatewayConfiguration`			10.2.1.0	Root element of the multicast gateway configuration instance document.
`@schemaVersion`	1	Unsigned Integer	10.2.1.0	Schema version number for this multicast gateway configuration instance document, as specified in clause A.0.
`@validityPeriod`	0..1	Duration	10.2.1.0	Time period after receipt for which this instance document is valid. If this attribute is present, then `@validUntil` should not be present. If the *Multicast gateway* has received the multicast gateway configuration via a cached resource and this attribute is present, then the value shall be reduced by the cache age.
`@validUntil`	0..1	dateTime	10.2.1.0	Deadline after which this instance document ceases to be valid. If this attribute is present, then `@validityPeriod` should not be present.
`MulticastGatewayConfiguration TransportSession`	0..n		10.2.5	Container for multicast gateway configuration transport session parameters.
`MulticastSession`	0..n		10.2.2	Container for multicast session parameters.
`MulticastGatewaySessionReporting`	0..1		10.2.1.0	Container for reporting parameters.
`ReportingLocator`	1..n	URI String	10.2.1.0	Container for a multicast gateway reporting endpoint.
`@proportion`	0..1	Decimal	10.2.1.0	Proportion of *Multicast gateway* instances that should send session reports to the specified endpoint, expressed as a decimal value between 0.0 and 1.0.
`@period`	1	Duration String	10.2.1.0	Session reporting periodicity expressed in seconds. The value 0 disables periodic reporting and reports are then sent only when significant events occur.
`@randomDelay`	1	Unsigned Integer	10.2.1.0	An additional random period that a *Multicast gateway* should delay between sending playback session reports.
`@reportSessionRunningEvents`	0..1	Boolean	10.2.1.0	Determines whether Playback session running events are reported.

10.2.2 Multicast session parameters

10.2.2.0 General

A multicast session describes a set of multicast transport sessions (specified in clause 10.2.3 below) that convey components of a particular linear service. The multicast session parameters are used by a *Multicast server* to configure its *Multicast transmission* subfunction. The multicast session parameters are used by a *Multicast gateway* to configure its *Multicast reception* subfunction.

Each multicast session is specified using a separate **`MulticastSession`** element in the multicast session configuration instance document.

A multicast session shall be assigned a service identifier that is unique within the scope of the deployed system and this is conveyed in the `@serviceIdentifier` attribute. This service identifier may be used to cross-reference descriptive metadata for the linear service in question, such as that specified in [i.6].

To account for delay in the transmission and repair of multicast transport objects, the *Multicast gateway* may be instructed to delay the presentation timeline (e.g. by manipulating the presentation manifest or by delaying the advertised availability of the corresponding playback delivery objects). The `@contentPlaybackAvailabilityOffset` attribute may be used to specify the length of this delay. The value of this attribute shall be a time interval, as specified in clause 5.5.2 of ISO 8601-1 [11], but excluding the alternative format specified in clause 5.5.2.4 of [11]. If omitted, the delay shall be assumed to be zero. When the *Multicast gateway* in-band configuration method is used at reference point **M** (per clauses 8.3.5 and 10.2.5) this attribute and its value shall be copied from the multicast server configuration instance document into the corresponding multicast session of the multicast gateway configuration instance document. For other *Multicast gateway* configuration methods, this attribute should be omitted from the multicast server configuration instance document.

10.2.2.1 MulticastSession element

The syntax of the **MulticastSession** element is specified in table 10.2.2.1-1 below:

Table 10.2.2.1-1: MulticastSession element syntax

Element or attribute name	Use	Data type	Clause	Description
MulticastSession			10.2.2	
@serviceIdentifier	1	URI string	10.2.2.0	Service identifier for the logical service with which this session is associated.
@contentPlaybackAvailabilityOffset	0..1	Duration string	10.2.2.0	Availability time offset adjustment applied to the original presentation manifest when passed to instances of the *Content playback* function.
PresentationManifestLocator	1..n	URI string	10.2.2.2	URL of a presentation manifest for the linear service.
@manifestId	1	Name Token string	10.2.2.2	Uniquely identifies this presentation manifest within the scope of a multicast session.
@contentType	1	MPEG-7 *mimeType*	10.2.2.2	The MIME content type of this presentation manifest.
@transportObjectURI	0..1	URI string	10.2.2.2	Multicast transport object URI to be used when conveying this presentation manifest in an in-band carousel of multicast transport objects.
@contentPlaybackPathPattern	0..1	String	10.2.2.2	A pattern string used by the *Multicast gateway* to match the path part of presentation manifest request URLs from a *Content playback* function with this multicast session. May contain any number of wildcard characters at any position.
MulticastGatewaySessionReporting	0..1		10.2.2.3	Container for multicast session reporting parameters.
ReportingLocator	1..n	URI String	10.2.2.3	Container for a multicast gateway reporting endpoint.
@proportion	0..1	Decimal	10.2.2.3	Proportion of *Multicast gateway* instances that should send session reports to the specified endpoint, expressed as a decimal value between 0.0 and 1.0.
@period	1	Duration string	10.2.2.3	Session reporting periodicity expressed in seconds. The value 0 disables periodic reporting and reports are then sent only when significant events occur.
@randomDelay	1	Unsigned Integer	10.2.2.3	An additional random period that a *Multicast gateway* should delay between sending playback session reports.
@reportSessionRunningEvents	0..1	Boolean	10.2.2.3	Determines whether Playback session running events are reported.
MulticastTransportSession	0..n		10.2.3	Container for multicast transport session parameters.

10.2.2.2 Presentation manifest locator

The URLs of presentation manifests related to the linear service shall be conveyed in one or more **PresentationManifestLocator** child elements of **MulticastSession**.

NOTE 1: This plurality allows presentation manifests of different types that reference a common stream of media objects to be associated with a single set of multicast transport sessions.

Every **PresentationManifestLocator** element shall carry an identifier in its @manifestId attribute that is unique within the scope of its parent **MulticastSession**. This identifier is used when associating service components with multicast transport sessions, as specified in clause 10.2.4.

The MIME media type [14] of the presentation manifest shall be indicated using the @contentType attribute of the **PresentationManifestLocator** element as follows:

Presentation manifest type	MIME media type(s)
DASH Media Presentation Description (MPD)	*application/dash+xml*
HLS Master Playlist	*application/vnd.apple.mpegURL* or *audio/mpegurl*

The multicast transport object URI of the presentation manifest may be specified by means of the optional @transportObjectURI attribute. The attribute value shall be unique within the scope of the multicast session configuration.

NOTE 2: This attribute allows a *Provisioning* function to decide the multicast transport object URI for a presentation manifest and then to signal its value in the multicast server configuration delivered at reference point **C_{MS}** and in the multicast gateway configuration delivered at reference point **C_{MR}**. It also allows the multicast transport object URI to be signalled to the *Multicast gateway* in a multicast gateway configuration transport session (see clause 8.3.5).

The **PresentationManifestLocator** element may carry an optional @contentPlaybackPathPattern attribute which the *Multicast gateway* uses as specified in clause 8.4.1.0. This attribute carries a pattern string that is matched against the path component of the presentation manifest request URL from the *Content playback* function at reference point **L**. The path component used in this comparison includes the leading "/" character, as required by section 3.3 of [15], and therefore the pattern string shall also include a leading "/" character.

The pattern string may include one or more asterisk ("*") or question mark ("?") wildcard characters.

- A "?" wildcard character matches any single character from the set *pchar* (as defined in appendix A of [15]) at the corresponding position in the presentation manifest request path.
- An "*" wildcard character matches any sequence of characters from the set *pchar* at the corresponding position in the presentation manifest request path component.

If the request target includes the literal characters "$", "*" or "?", then these literals shall be prefixed with the "$" escape character in the pattern string.

When the **PresentationManifestLocator** element is carried in a multicast server configuration:

- If the presentation manifest is to be acquired by the *Content ingest* subfunction of the *Multicast server* using the push content ingest method (see clause 8.3.1) the element content shall be the reference point **P_{in}'** URL on the *Multicast server* to which the presentation manifest is to be pushed.
- If the presentation manifest is to be acquired by the *Content ingest* subfunction of the *Multicast server* using the pull content ingest method (see clause 8.3.2) the element content shall be the reference point **O_{in}** URL on the *Content hosting* function from which the presentation manifest is to be retrieved.

NOTE 3: The intention to push or pull the presentation manifest is independent of the content ingest method (see clause 10.2.3.4).

When the **PresentationManifestLocator** element is carried in a multicast gateway configuration:

- If the presentation manifest is available for unicast retrieval and/or repair via reference point **A**, the element content shall be its URL on the *Content hosting* function.

NOTE 4: The URL at reference point **A** may differ from that at reference point **O_{in}**.

- Otherwise (for example if reference point **A** is not present in the deployment) the element content shall be empty and the @contentPlaybackPathPattern attribute shall be present and non-empty.

10.2.2.3 Multicast gateway session reporting parameters

A *Multicast gateway* may be configured to report metrics for a single multicast session to one or more external *Service reporting capture* subfunctions.

NOTE 1: Reporting on an individual multicast session may be declared independently of reporting for all multicast sessions (see clause 10.2.1.0) and the two levels of reporting may therefore operate in parallel.

NOTE 2: The target *Service reporting capture* subfunctions may be operated by different business entities.

Reporting instance documents are periodically submitted to the declared set of *Service reporting capture* subfunctions only by a randomly selected sample of *Multicast gateway* instances in the deployed system. Each *Multicast gateway* determines whether it will send reports to a given *Service reporting capture* subfunction by independent random choice, weighted according to a configurable sample proportion.

One or more reporting endpoints shall be declared inside the **`MulticastGatewaySessionReporting`** child element of **`MulticastSession`**, each endpoint being specified as a URL in the content of a separate **`ReportingLocator`** element. The use of this endpoint locator is specified in clause 11.1:

- The `@proportion` attribute should be used to indicate what sample of *Multicast gateway* instances in the deployed system report to the endpoint specified in the enclosing **`ReportingLocator`** element. Omitting this attribute indicates that all *Multicast gateway* instances should report (corresponding to a proportion of 1.0).
- The `@period` attribute shall be used to indicate the time gap between consecutive reports being submitted by the *Multicast gateway*. The value 0 indicates that periodic reporting is disabled, in which case reporting occurs whenever an event is to be reported as defined in clause 11.2.2.
- The `@randomDelay` attribute shall be used by the *Multicast gateway* as an additional delay after the above time gap.
- The `@reportSessionRunningEvents` flag controls whether **Playback session running** events (see clause 11.2.2.2) are to be included in the reporting document instance.

When the *Multicast gateway* in-band configuration method is used at reference point **M** (per clauses 8.3.5 and 10.2.5) the **`MulticastGatewaySessionReporting`** element and all its children shall be copied from the multicast server configuration instance document into the corresponding multicast session of the multicast gateway configuration instance document. For other *Multicast gateway* configuration methods, the **`MulticastGatewaySessionReporting`** element should be omitted from the multicast server configuration instance document.

10.2.3 Multicast transport session parameters

10.2.3.0 General

The parameters of each multicast transport session are captured in a separate **`MulticastTransportSession`** child element of **`MulticastSession`**. Each such multicast transport session is explicitly associated with service component(s) in the presentation manifest(s), as described in clause 10.2.4 below.

10.2.3.1 MulticastTransportSession element

The syntax of the **`MulticastTransportSession`** element is specified in table 10.2.3.1-1 below:

Table 10.2.3.1-1: MulticastTransportSession element syntax

Element or attribute name	Use	Data type	Clause	Description
`MulticastTransportSession`			10.2.3	Container for multicast transport session parameters.
`@id`	1	Name Token string	10.2.3.2	Uniquely identifies a multicast transport session within the scope of its parent multicast session. The assignment of this value is implementation-specific.

Element or attribute name	Use	Data type	Clause	Description
@serviceClass	0..1	MPEG-7 *termReference*	10.2.3.2A	A fully-qualified term identifier URI from the vocabulary specified in clause 9 of [41] or from another vocabulary indicating the class of information conveyed by this multicast transport session.
@start	0..1	MPEG-7 *TimePoint*	10.3.1	Start time of the multicast transport session.
@duration	0..1	Duration string	10.3.1	Duration of the multicast transport session.
@contentIngestMethod	0..1	String	10.2.3.4	Indicates how ingest objects for this multicast transport session are made available to the *Multicast server*. If a *Multicast gateway* receives this attribute, it shall be ignored.
@transmissionMode	0..1	String	10.2.3.5	Indicates whether ingest objects for this multicast transport session are mapped to one or more multicast transport objects. If present, it shall take the value of *resource* or *chunked*.
@transportSecurity	0..1	String	10.2.3.6	Indicates which security features are enabled in this multicast transport session.
@sessionIdleTimeout	1	Integer	10.2.3.7	The period of time a *Multicast gateway* should wait for new packets on this multicast transport session before assuming the session is inactive. *Multicast server* functions shall ignore this attribute.
TransportProtocol	1		10.2.3.8	Container for multicast media transport protocol parameters.
@protocolIdentifier	1	MPEG-7 *termReference*	10.2.3.8	A fully-qualified term identifier URI from *MulticastTransportProtocolCS* (clause B.1) specifying the conformant multicast media transport protocol used for this multicast transport session, or a URI specifying a non-conformant multicast media transport protocol.
@protocolVersion	1	String	10.2.3.8	Indicates the version of the multicast media transport protocol.
EndpointAddress	1..n		10.2.3.9	Container for the addressing parameters of this multicast transport session.
NetworkSourceAddress	0..1	String	10.2.3.9	A host name or IP address specifying the multicast group source address for use with source-specific multicast transport.
NetworkDestinationGroup Address	1	IP address string	10.2.3.9	An IP address specifying a multicast group for the multicast transport session.
TransportDestinationPort	1	Unsigned 16-bit integer	10.2.3.9	The UDP port number of the multicast transport session.
MediaTransportSession Identifier	0..1	Positive integer	10.2.3.9	An opaque transport session identifier specified by the multicast media transport protocol.
BitRate	1		10.2.3.10	Container specifying the bit rate(s) of the multicast transport session across all declared endpoint addresses and including any FEC overheads.
@average	0..1	Positive integer	10.2.3.10	The average bit rate of the multicast transport session.
@maximum	1	Positive integer	10.2.3.10	The maximum bit rate of the multicast transport session.

Element or attribute name	Use	Data type	Clause	Description
ForwardErrorCorrection Parameters	0..n		10.2.3.11	Container for the parameters of the FEC repair blocks protecting the multicast transport session.
SchemeIdentifier	1	MPEG-7 *termReference*	10.2.3.11	A fully-qualified term identifier URI from *ForwardErrorCorrectionSchemeCS* (clause B.2) specifying the scheme of the FEC repair blocks protecting the multicast transport session.
OverheadPercentage	1	Positive integer	10.2.3.11	Percentage of FEC overhead compared with source packets. For example, a value of 20 means 20 % overhead, 100 means there is a repair packet for every source packet. Values greater than 100 are permitted.
EndpointAddress	0..n		10.2.3.9	Container for the addressing parameters of FEC repair blocks protecting this multicast transport session. May be omitted if FEC repair packets are carried in band with the multicast transport session.
NetworkSourceAddress	0..1	String	10.2.3.9	An optional host name or IP address specifying the source address of FEC repair packets protecting this multicast transport session.
NetworkDestinationGroup Address	1	IP address string	10.2.3.9	An IP address specifying the multicast group for FEC repair packets protecting this multicast transport session.
TransportDestinationPort	1	Unsigned 16-bit integer	10.2.3.9	The UDP port number of FEC repair packets protecting this multicast transport session.
MediaTransportSession Identifier	0..1	Positive integer	10.2.3.9	An opaque transport session identifier of FEC repair packets protecting this multicast transport session.

Element or attribute name	Use	Data type	Clause	Description
UnicastRepairParameters	0..1		10.2.3.12	Container for parameters pertaining to unicast repair.
@transportObjectBaseURI	0..1	URI string	10.2.3.13	Multicast transport object base URI. The base URI of all multicast transport objects conveyed in this multicast transport session.
@transportObjectReception Timeout	1	Unsigned integer	10.2.3.12	The period of time (in milliseconds) that a *Multicast gateway* should wait for a packet relating to a multicast transport object before assuming that the object transmission is over.
@fixedBackOffPeriod	0..1	Unsigned integer	10.2.3.12	The minimum number of milliseconds that the *Multicast gateway* shall wait after the end of object transmission before attempting unicast repair.
@randomBackOffPeriod	0..1	Unsigned integer	10.2.3.12	Maximum number of milliseconds for the random delay period that the *Multicast gateway* shall wait in addition to @fixedBackOffPeriod.
BaseURL	0..n	URI string	10.2.3.13	Unicast repair base URL prefix. The base path of a unicast repair endpoint.
@relativeWeight	0..1	Unsigned integer	10.2.3.13	A relative weighting for this unicast repair endpoint.
ObjectCarousel	0..1		10.2.3.14.0	Container for parameters pertaining to the in-band carousel of ancillary multicast transport objects.
@aggregateTransportSize	0..1	Unsigned integer	10.2.3.14.0	The combined size of all transport objects described by this carousel, excluding any associated metadata and transport protocol overhead.
@aggregateContentSize	0..1	Unsigned integer	10.2.3.14.0	The combined size of all transport objects described by this carousel, excluding any associated metadata and transport protocol overhead and excluding any content encoding (such as compression). May be omitted if no content encoding is applied to the multicast transport objects.
PresentationManifests	0..1		10.2.3.14.1	The presence of this element indicates that presentation manifests shall be conveyed in the multicast transport session when the parent multicast session is active.
@targetAcquisitionLatency	0..1	Duration string	10.2.3.14.0	Repetition rate of presentation manifests in the multicast transport session.
@compressionPreferred	0..1	Boolean	10.2.3.14.0	Expression of preference to the *Multicast server* that compression of the resulting multicast transport object(s) is desired by the *Provisioning* function. Only permitted in multicast server configuration instance documents.
InitSegments	0..1		10.2.3.14.2	The presence of this element indicates that initialisation segments shall be conveyed in the multicast transport session when the parent multicast session is active.
@targetAcquisitionLatency	0..1	Duration string	10.2.3.14.0	Repetition rate of initialisation segments in the multicast transport session.

Element or attribute name	Use	Data type	Clause	Description
@compressionPreferred	0..1	Boolean	10.2.3.14.0	Expression of preference to the *Multicast server* that compression of the resulting multicast transport object(s) is desired by the *Provisioning* function. Only permitted in multicast server configuration instance documents.
ResourceLocator	0..n	URI string	10.2.3.14.3	The URL of a resource that shall be conveyed in the multicast transport session when the parent multicast session is active.
@targetAcquisitionLatency	0..1	Duration string	10.2.3.14.0	Repetition rate of the resource in the multicast transport session.
@revalidationPeriod	0..1	Duration string	10.2.3.14.3	The period of time that a *Multicast server* should wait before revalidating the resource.
@compressionPreferred	0..1	Boolean	10.2.3.14.0	Expression of preference to the *Multicast server* that compression of the resulting multicast transport object(s) is desired by the *Provisioning* function. Only permitted in multicast server configuration instance documents.
ServiceComponentIdentifier	1..n		10.2.4	Linkages between this multicast transport session and service components(s) in presentation manifest(s) declared by the parent multicast session.

10.2.3.2 Multicast transport session identifier

Every multicast transport session shall be assigned an identifier, indicated using the `@id` attribute of the **`MulticastTransportSession`** element. This identifier shall be unique within the scope of the enclosing multicast session.

NOTE: This identifier is used, in combination with the `@serviceIdentifier` attribute of the parent multicast session, to manually (de)activate the multicast transport session, as specified in clause 10.3.2.

10.2.3.2A Service Class

The type of content carried by a multicast transport session should be signalled using the optional `@serviceClass` attribute of the **`MulticastTransportSession`** element.

DVB-I service instances shall be signalled according to clause 9 of [41].

A *Multicast gateway* should ignore multicast transport sessions described using an unknown service class term identifier.

10.2.3.3 Multicast transport session timing

The start time and duration of the multicast transport session may be specified using the `@start` and `@duration` attributes respectively of the **`MulticastTransportSession`** element. The values of these attributes shall be set in accordance with clause 10.3.1.

If desired, the multicast transport session start time shall be indicated using the `@start` attribute. The value of this attribute shall comply with the *TimePoint* datatype, as specified in clause 6.4.3 of MPEG-7 Part 5 [12].

NOTE 1: The MPEG-7 *TimePoint* datatype is a restricted subset of ISO 8601-1, clause 5.4 [11].

NOTE 2: With this datatype it is not possible to indicate Coordinated Universal Time (UTC) using the UTC designator *Z*. Per clause 6.4.3 of MPEG-7 Part 5 [12], if no time zone is specified a time zone of 00:00 is assumed rather than local time.

If desired, the multicast transport session duration shall be indicated using the `@duration` attribute.

10.2.3.4 Content ingest method

The means by which ingest objects are to be acquired by the *Multicast server* may be indicated in a multicast server configuration instance document using the `@contentIngestMethod` attribute of the **`MulticastTransportSession`** element as follows:

- **Push content ingest method.** If ingest objects are to be pushed to the *Content ingest* subfunction at reference point $\mathbf{P_{in}'}$ (as specified in clause 8.3.1) then the attribute shall have the value *push*.
- **Pull content ingest method.** If ingest objects are to be pulled by the *Content ingest* subfunction from a *Content hosting* function at reference point $\mathbf{O_{in}}$ (as specified in clause 8.3.2) then the attribute shall have the value *pull*.

If this attribute is omitted, the pull content ingest method shall be assumed by a *Multicast server*.

The `@contentIngestMethod` attribute shall not be present in multicast gateway configuration instance documents.

10.2.3.5 Multicast transmission mode

The means by which a *Multicast server* is to map ingest objects to multicast transport objects may be indicated in a multicast server configuration instance document using the `@transmissionMode` attribute of the **`MulticastTransportSession`** element as follows:

- **Resource transmission mode.** If one ingest object is mapped to one multicast transport object then the attribute shall have the value *resource*.
- **Chunked transmission mode.** If one ingest object is mapped to more than one multicast transport objects (e.g. each HTTP chunk of an ingest object mapped to a separate multicast transport object) then the attribute shall have the value *chunked*.

If this attribute is omitted, resource transmission mode shall be assumed.

The `@transmissionMode` attribute should be present in multicast gateway configuration instance documents to signal the multicast transmission mode to the *Multicast reception* subfunction of the *Multicast gateway*.

10.2.3.6 Multicast transport security mode

The *Multicast transmission* subfunction may add metadata to each multicast transport object in order to preserve the integrity and/or authenticity of media objects in transit. The transport security mode is indicated using the `@transportSecurity` attribute of the **`MulticastTransportSession`** element as follows:

- **No security.** The absence of security is explicitly indicated using the value *none*.
- **Integrity only.** If an integrity check is provided for every multicast transport object in the multicast transport session, such as an object checksum, this shall be indicated using the attribute value *integrity*.
- **Authenticity and integrity.** If both authenticity and integrity checks are provided, this shall be indicated by the attribute value *integrityAndAuthenticity*.

The interpretation of these attribute values shall be specified by each multicast media transport protocol.

If this attribute is omitted, no security shall be assumed.

The `@transportSecurity` attribute should be present in multicast gateway configuration instance documents to signal the multicast transport security mode to the *Multicast reception* subfunction of the *Multicast gateway*.

10.2.3.7 Multicast transport session idle timeout

The maximum expected inter-packet time interval for a multicast transport session shall be specified in the `@sessionIdleTimeout` attribute of the **`MulticastTransportSession`** element. The value of the attribute shall be expressed in milliseconds. This attribute takes precedence over any other timeout value for the multicast transport session.

The *Multicast reception* subfunction of a *Multicast gateway* may unsubscribe from a multicast transport session if it has not received a packet at the configured multicast transport endpoint address (clause 10.2.3.9 below) for this time period in order to release reserved resources associated with the multicast transport session.

NOTE: When a transport session times out unicast repair may be triggered immediately, unless otherwise specified by the multicast media transport protocol, and still subject to the values of the `@fixedBackOffPeriod` and `@randomBackOffPeriod` attributes specified in clause 10.2.3.12 below.

10.2.3.8 Multicast media transport protocol parameters

The multicast media transport protocol used for this multicast transport session shall be specified using the **`TransportProtocol`** child element of the **`MulticastTransportSession`** element. This information enables a Multicast gateway to determine whether it can consume the multicast transport session.

- The `@protocolIdentifier` attribute of this element shall be used to unambiguously identify the multicast media transport protocol in use. Implementations conforming with the present document shall signal a controlled term from the MPEG-7 Classification Scheme *MulticastTransportProtocolCS* specified in clause B.1.
- The `@protocolVersion` attribute shall indicate the major version number of the multicast media transport protocol in use.

10.2.3.9 Multicast transport endpoint address

The transport endpoint address(es) used for a multicast transport session shall be specified using the **`EndpointAddress`** child element of the **`MulticastTransportSession`** element.

NOTE: Certain multicast media transport protocols may permit a service component to be transmitted on more than one transport address, as specified in clause 8.3.4.0. In such cases, the *Multicast reception* subfunction of a *Multicast gateway* needs to subscribe to all specified transport addresses in order to completely receive all relevant multicast transport objects.

The host name or source network IP address of the system originating the multicast transport session shall be specified in the **`NetworkSourceAddress`** child element in the case of source-specific multicast, and the destination group network address shall be specified in the **`NetworkDestinationGroupAddress`** child element:

- IPv4 network addresses shall be expressed using the "quad-dotted decimal" string notation.
- IPv6 network addresses shall be expressed using the colon-separated string notation recommended in IETF RFC 5952 [13].

The transport protocol destination port number of the multicast transport session shall be specified in the **`TransportDestinationPort`** child element. It shall be an integer between 1 and 65535 inclusive.

A multicast media transport protocol session identifier may additionally be specified in the **`MediaTransportSessionIdentifier`** child element. It shall be a positive integer.

10.2.3.10 Multicast transport session bit rate

The bit rate of this multicast transport session shall be specified using the **`BitRate`** child element of the **`MulticastTransportSession`** element as follows:

- The maximum bit rate shall be indicated in the `@maximum` attribute.
- The average bit rate may additionally be indicated in the `@average` attribute.

Bit rate values in the above attributes shall be expressed in bits per second.

The values shall indicate the bit rate of the multicast media transport protocol data units conveyed by the underlying transport protocol (e.g. carried in the payload field of UDP datagrams) across all endpoints declared for this multicast transport session, including any FEC repair packets addressed to the same destination group network address.

10.2.3.11 Forward Error Correction parameters

A multicast transport session may provide Application Level Forward Error Correction (AL-FEC) repair packets to assist with multicast packet loss, as specified in clause 8.3.4.2. These repair packets may be transmitted on the same endpoint address as the packets they are intended to repair, or on a different endpoint address.

Where Forward Error Correction repair packets are available as part of a multicast transport session, the parameters for receiving and interpreting them shall be specified using the optional **`ForwardErrorCorrectionParameters`** child element of the **`MulticastTransportSession`** element as follows:

- The AL-FEC scheme shall be indicated using the **`SchemeIdentifier`** child element. Its value shall be a fully-qualified term identifier from the *ForwardErrorCorrectionSchemeCS* controlled vocabulary specified in clause B.2.

NOTE: The set of valid AL-FEC schemes is specified separately by each multicast media transport protocol.

- The percentage overhead of AL-FEC repair packets compared with source packets shall be indicated using the **`OverheadPercentage`** child element.

- If the endpoint address(es) to which AL-FEC repair packets are sent differ from those of the enclosing multicast transport session, they shall be indicated using one or more **`EndpointAddress`** child elements, as specified in clause 10.2.3.9 above.

Each multicast media transport protocol shall specify what is meant when the **`ForwardErrorCorrectionParameters`** element is omitted.

10.2.3.12 Unicast repair parameters

Clause 9 specifies how a *Multicast gateway* performs unicast repair at reference point **A**. Where such unicast repair is available, the parameters governing these procedures shall be specified using the **`UnicastRepairParameters`** child element of the **`MulticastTransportSession`** element as follows:

NOTE: In the case of unidirectional system deployments this child element should be omitted.

- **Transport object reception timeout.** The `@transportObjectReceptionTimeout` attribute shall specify a timeout period, expressed in milliseconds. This shall signal the maximum time a *Multicast gateway* should wait for the arrival of the next (or final) data packet relating to a particular multicast transport object before resorting to Forward Error Correction or a unicast repair operation.

- **Fixed back-off period.** The `@fixedBackOffPeriod` attribute may be provided to specify an additional fixed delay, expressed in milliseconds, that a *Multicast gateway* shall wait after the above object completion timeout before commencing a unicast repair operation. A value of 0 implies that there is no additional fixed delay. If this attribute is omitted, the value 0 shall be assumed.

- **Random back-off period.** The `@randomBackOffPeriod` attribute may be provided to specify an additional random delay, expressed in milliseconds, that a *Multicast gateway* shall wait after the above fixed back-off period before commencing a unicast repair operation. The *Multicast gateway* should select a different random back-off period between 0 and the specified random back-off period for each multicast transmission object. A value of 0 implies that there is no additional random delay. If this attribute is omitted, the value 0 shall be assumed.

When the *Multicast gateway* in-band configuration method is used at reference point **M** (per clauses 8.3.5 and 10.2.5) the **`UnicastRepairParameters`** element and all its children shall be copied from the multicast server configuration instance document into the corresponding multicast transport session of the multicast gateway configuration instance document. For other *Multicast gateway* configuration methods, the **`UnicastRepairParameters`** element should be omitted from the multicast server configuration instance document.

10.2.3.13 Multicast gateway path mapping parameters

A *Multicast gateway* may be configured to transform the URI associated with a multicast transport object at reference point **M** into a different URL that can be used for unicast repair operations at reference point **A**. To perform this transformation, a **multicast transport object base URI** prefix in the multicast transport object URI is substituted with a **unicast repair base URL** prefix, as specified in clause 9.2.2.

Where present, the multicast transport object base URI for this multicast transport session shall be indicated using the `@transportObjectBaseURI` attribute of the **`UnicastRepairParameters`** element:

- The multicast transport object base URI may be present in multicast server configuration instance documents. Omission shall indicate that the multicast transport object URI is to be assigned by the *Multicast server* implementation for the multicast transport session in question.
- The transport object base URI may be present in multicast gateway configuration instance documents. Omission shall indicate that the multicast transport object base URI is the empty string for the multicast transport session in question.
- The multicast transport object base URI value shall be an absolute URI as defined in section 4.3 of IETF RFC 3986 [15]. It shall not contain any query or fragment component specified for other purposes by the present document.

A **`UnicastRepairParameters`** element may declare a (possibly empty) set of unicast repair base URLs, and each one shall be indicated as the content of a separate **`BaseURL`** child element. The value of this element shall be an absolute URI as defined in section 4.3 of IETF RFC 3986 [15]. It shall not contain any query or fragment component specified for other purposes by the present document.

NOTE 1: The way in which a *Multicast gateway* chooses between multiple unicast repair base URLs is specified in clause 9.2.1.

If no **`BaseURL`** child elements are declared, the unicast repair URL for any given multicast transport object shall be determined as specified in clause 9.2.2.

A relative weight may be associated with each unicast repair base URL using the optional `@relativeWeight` attribute. If present, this attribute shall have an integer value of zero or greater. The use of this attribute in choosing between multiple unicast repair base URLs is specified in clause 9. Setting the value of this attribute to zero shall indicate that the unicast repair base URL in question is to be ignored by the *Multicast gateway*.

NOTE 2: This latter indication can be used to temporarily disable the use of a particular unicast repair base URL for operational reasons.

Omitting the `@relativeWeight` attribute shall have the same meaning at setting its value to 1.

NOTE 3: Thus, if all **`BaseURL`** child elements omit the `@relativeWeight` attribute then the corresponding unicast repair base URLs are all deemed to have equal weight.

10.2.3.14 In-band carousel of ancillary multicast transport objects

10.2.3.14.0 General

The presence of the optional **`ObjectCarousel`** child element of the **`MulticastTransportSession`** element indicates that the *Multicast server* shall compile and transmit a carousel of ancillary multicast transport objects as part of a multicast transport session, as specified in clause 8.3.4.4.

The total size of the transport objects that are carouselled may be signalled using the optional `@aggregateTransportSize` and `@aggregateContentSize` attributes of the **`ObjectCarousel`** element. Both attributes shall exclude the size of any associated metadata or protocol overhead in their value.

- The `@aggregateTransportSize` attribute indicates the size of the transport objects as transmitted at reference point **M**.
- The `@aggregateContentSize` attribute indicates the size of the transport objects when any content encoding (such as compression) has been removed from them. This attribute may be omitted if no content encoding is applied to the multicast transport objects in the carousel.

These attributes should be present in a multicast gateway configuration instance document to enable a *Multicast gateway* to decide whether it has sufficient resources to decode the object carousel.

These attributes may be present in a multicast server configuration instance document (for example, when declaring a object carousel that is not generated by the *Multicast server*), but their values may be ignored by the *Multicast server* and recalculated when it generates a corresponding multicast gateway configuration instance document.

Three different types of ancillary media object may be carouselled:

- Presentation manifests, as specified in clause 10.2.3.14.1.
- Initialisation segments, as specified in clause 10.2.3.14.2.
- Other resources, as specified in clause 10.2.3.14.3.

For all types of ancillary media object, the optional `@targetAcquisitionLatency` attribute indicates how frequently the multicast transport objects are to be repeated in the multicast transport session so that they can be acquired by a *Multicast gateway* within the indicated target time. If this attribute is omitted, the multicast transport objects should be repeated as frequently as necessary for successful operation of the system such that the multicast transport session does not exceed its declared bit rate (see clause 10.2.3.10).

For all types of ancillary media object, the revalidation provisions of clause 8.3.4.4 shall apply unless overridden by additional configuration parameters specified in the following subclauses.

For all types of ancillary media object, setting the value of the optional `@compressionPreferred` attribute to *true* in a multicast server configuration instance document indicates to the *Multicast server* that compression of the corresponding multicast transport object described is desired by the *Provisioning* function.

NOTE 1: The compression codec used by the *Multicast server* is dependent on the multicast transport protocol configured for the multicast transport session.

NOTE 2: The *Multicast server* may choose not to compress multicast transport objects, for example if the multicast transport protocol does not specify a compression codec, if the *Multicast server* does not implement any of the compression codecs specified by the configured multicast transport protocol, if there are insufficient resources to perform the compression operation, or if compression would not produce a substantive reduction in object size for the effort required.

The `@compressionPreferred` attribute shall not be present in multicast gateway configuration instance documents.

10.2.3.14.1 In-band carousel parameters for presentation manifests

If the optional **`PresentationManifests`** child element is present, the presentation manifests for all service components declared in this multicast transport session shall be carouselled.

- For DASH service components (configured per clause 10.2.4.1), the referenced Media Presentation Description shall be carouselled.
- For HLS service components (configured per clause 10.2.4.2), the master playlist plus the referenced media playlist for that service component shall be carouselled.
- For hybrid multicast transport sessions that have both DASH- and HLS-based service components, all of the abovementioned ancillary media objects shall be carouselled.

10.2.3.14.2 In-band carousel parameters for initialisation segments

If the optional **`InitSegments`** child element is present, the initialisation segments for all service components declared in this multicast transport session shall be carouselled.

- For DASH service components (configured per clause 10.2.4.1), only the initialisation segment(s) for the currently active Period of the Representation(s) mapped to the relevant service components shall be carouselled.
- For HLS service components (configured per clause 10.2.4.2), only the media initialization section(s) referenced via the *EXT-X-MAP* tag from the relevant media playlist(s) mapped to the relevant service component(s) shall be carouselled.
- For hybrid multicast transport sessions that have both DASH- and HLS-based service components, all of the abovementioned ancillary media objects shall be carouselled.

10.2.3.14.3 In-band carousel parameters for other resources

If the optional **`ResourceLocator`** child element is present, the resource at the URL indicated in the content of that child element shall be carouselled.

If the optional `@revalidationPeriod` attribute is present, the *Content ingest* subfunction shall revalidate the ancillary ingest object with the *Content hosting* function at the indicated frequency.

10.2.4 Association between multicast transport session and service component

10.2.4.0 General

Every multicast transport session shall be associated with one or more service components in a presentation manifest associated with the parent multicast session. Each such association shall be realized by means of a separate **`ServiceComponentIdentifier`** child element. The presentation manifest shall be referenced by assigning the `@manifestId` value of the target **`PresentationManifestLocator`** (see clause 10.2.2.2) to the `@manifestIdRef` attribute of this element. The schema type and form of this element depends on the type of presentation manifest referenced.

In the case where the presentation manifest is not of a type described in any of the following clauses:

- The `@xsi:type` attribute of the **`ServiceComponentIdentifier`** element shall be set to the value *GenericComponentIdentifierType*.
- The value of the `@componentIdentifier` attribute shall uniquely identify a service component in the presentation.

If multiple service components of a linear service are carried by the same multicast transport session, then each component shall be explicitly identified using a separate **`ServiceComponentIdentifier`** element.

A multicast transport session may carry multiple **ServiceComponentIdentifier** elements with different values of the @xsi:type attribute, for example if an MPEG-DASH and an HLS presentation share the same multicast transport objects.

A multicast transport session may carry multiple **ServiceComponentIdentifier** elements with the same value of @xsi:type, for example if there are two MPEG-DASH presentations associated with a linear service that include overlapping but non-identical sets of representations.

10.2.4.1 Service component identification for DASH presentations

In the case where the referenced presentation manifest is a DASH media presentation description (MPD) [i.2], the @xsi:type attribute of the **ServiceComponentIdentifier** element shall be set to the value *DASHComponentIdentifierType*. The syntax of this schema type is specified in table 10.2.4.1-1 below.

Table 10.2.4.1-1: DASHComponentIdentifier syntax

Element or attribute name	Use	Data type	Description
ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType"			Container unambiguously identifying an MPEG-DASH representation whose media objects are conveyed by the parent multicast transport session.
@manifestIdRef	1	Name Token string	A cross-reference to a **PresentationManifestLocator/**@manifestId in the parent multicast session that is an MPD.
@periodIdentifier	1	String	A **Period/**@id from the referenced MPD.
@adaptationSetIdentifier	1	Unsigned integer	An **AdaptationSet/**@id from the referenced MPD.
@representationIdentifier	1	String	A **Representation/**@id from the referenced MPD.
NOTE: All **Period** elements in an MPD of type *dynamic* contain the optional @id attribute, as required by clause 5.3.2.2 of the MPEG-DASH specification [i.2]).			

10.2.4.2 Service component identification for HLS presentations

In the case where the referenced presentation manifest is an HLS Master Playlist (.m3u8) [i.5], the @xsi:type attribute of the **ServiceComponentIdentifier** element shall be set to the value *HLSComponentIdentifierType*. The syntax of this schema type is specified in table 10.2.4.2-1 below.

Table 10.2.4.2-1: HLSComponentIdentifierType syntax

Element or attribute name	Use	Data type	Description
ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType"			Container identifying an HLS Media Playlist whose media objects are conveyed by the parent multicast transport session.
@manifestIdRef	1	Name Token string	A cross-reference to a **PresentationManifestLocator/**@manifestId in the parent multicast session that is an HLS Master Playlist.
@mediaPlaylistLocator	1	URI string	The absolute URL of an HLS Media Playlist appearing in the referenced HLS Master Playlist.

10.2.5 Multicast gateway configuration transport session parameters

10.2.5.0 General

In the case where the population of *Multicast gateway* instances is configured using the in-band configuration method at reference point **M** specified in clause 8.3.5 and clause 10.4.5, one or more **`MulticastGatewayConfiguration TransportSession`** elements shall be present in the multicast server configuration instance document. The *Multicast server* shall carousel the current multicast gateway configuration instance document on all specified multicast gateway configuration transport sessions.

In addition, when the in-band configuration method is used, a "bootstrap" multicast gateway configuration instance document may be provided at reference point **C_{MR}** containing one or more **`MulticastGatewayConfiguration TransportSession`** elements. An example can be found in clause C.2 and further worked examples can be found in clause 5 of [i.13]. In this case, the *Multicast gateway* shall receive additional multicast gateway configuration instance documents at reference point **M** on one of the specified multicast gateway configuration transport sessions.

NOTE 1: In the case of unidirectional system deployments an alternative bootstrapping mechanism is required to deliver the information conveyed in the **`MulticastGatewayConfigurationTransportSession`** element to the terminal device. The specification of this mechanism lies beyond the scope of the present document.

The elements and attributes of the **`MulticastGatewayConfigurationTransportSession`** element are a subset of those appearing in the **`MulticastTransportSession`** element specified in clause 10.2.3 above, plus additional elements specified in the following subclauses, as summarised below, with the following constraints and extensions.

The service class of the multicast gateway configuration transport session is indicated as specified in clause 10.2.3.2A.

The applicability of this multicast gateway configuration transport session may be indicated using the optional `@tags` attribute, the value of which shall be a list of URI strings.

NOTE 2: The service class and tags allow a *Multicast gateway* to select multicast gateway configuration transport session(s) that are relevant to it.

NOTE 3: The assignment and usage of tag values is beyond the scope of the present document.

The multicast media transport protocol used for the multicast gateway configuration transport session shall be specified per clause 10.2.3.8.

The transport endpoint address(es) used for the multicast gateway configuration transport session shall be specified per clause 10.2.3.9.

The bit rate of the multicast gateway configuration transport session shall be specified per clause 10.2.3.10.

The Forward Error Correction parameters of the multicast gateway configuration transport session shall be specified in the same manner as described in clause 10.2.3.11.

The unicast repair parameters of the multicast gateway configuration transport session shall be specified in the same manner as described in clause 10.2.3.12 and 10.2.3.13.

The in-band carousel of ancillary multicast transport objects to be conveyed in the multicast gateway configuration transport session shall be specified in the same manner as described in clause 10.2.3.14 with the following additions specified in clause 10.2.5.1 below:

- Child elements may optionally reference a particular multicast session by quoting its service identifier in a `@serviceIdRef` attribute.

NOTE 4: If the `@serviceIdRef` attribute is omitted, the scope of the child element is all active multicast sessions.

- Child elements may optionally reference a particular multicast transport session within the above referenced multicast session by quoting its identifier in a `@transportSessionIdRef` attribute.

NOTE 5: If the `@transportSessionIdRef` attribute is omitted, the scope of the child element is all multicast transport sessions within the referenced multicast session.

When carried in a multicast server configuration instance document, the **MulticastGatewayConfigurationTransportSession** may also carry one or more **MulticastGatewayConfigurationMacro** elements. This usage of this element is specified in clause 10.2.5.2.

The @compressionPreferred attribute of **ObjectCarousel** child elements shall not be present when a **MulticastGatewayConfigurationTransportSession** is carried in a multicast gateway configuration instance document.

10.2.5.1 MulticastGatewayConfigurationTransportSession element

The syntax of the **MulticastGatewayConfigurationTransportSession** element is specified in table 10.2.5.1-1 below. The semantics of the attributes and child elements are identical to the corresponding attributes and child elements of the **MulticastTransportSession** element specified in clause 10.2.3.

Table 10.2.5.1-1: MulticastGatewayConfigurationTransportSession syntax

Element or attribute name	Use	Data type	Clause	Description
MulticastGatewayConfiguration TransportSession			10.2.5	Container for multicast gateway configuration transport session parameters.
@serviceClass	0..1	MPEG-7 *termReference*	10.2.3.2A	A fully-qualified term identifier URI from the vocabulary specified in clause 9 of [41] or from another vocabulary indicating the class of information conveyed by this multicast transport session.
@tags	0..1	List of URI string	10.2.5.0	A whitespace-separated list of URIs [15] that indicate the applicability of this multicast gateway configuration transport session to the *Multicast gateway*.
@transportSecurity	0..1	String	10.2.3.6	Indicates which security features are enabled in this multicast gateway configuration transport session.
TransportProtocol	1		10.2.3.8	Container for multicast media transport protocol parameters.
@protocolIdentifier	1	MPEG-7 *termReference*	10.2.3.8	A fully-qualified term identifier URI from *MulticastTransportProtocolCS* (clause B.1) specifying the conformant multicast media transport protocol used for this multicast gateway configuration transport session, or a URI specifying a non-conformant multicast media transport protocol.
@protocolVersion	1	String	10.2.3.8	Indicates the version of the multicast media transport protocol.
EndpointAddress	1..n		10.2.3.9	Container for the addressing parameters of this multicast gateway configuration transport session.
NetworkSourceAddress	0..1	String	10.2.3.9	A host name or IP address specifying the multicast group source address for use with source-specific multicast transport.
NetworkDestinationGroup Address	1	IP address string	10.2.3.9	An IP address specifying a multicast group for the multicast gateway configuration transport session.
TransportDestinationPort	1	Unsigned 16-bit integer	10.2.3.9	The UDP port number of the multicast gateway configuration transport session.
MediaTransportSession Identifier	0..1	Positive integer	10.2.3.9	An opaque transport session identifier specified by the multicast media transport protocol.

Element or attribute name	Use	Data type	Clause	Description
BitRate	1		10.2.3.10	Container specifying the bit rate(s) of the multicast gateway configuration transport session across all declared endpoint addresses and including any FEC overheads.
@average	0..1	Positive integer	10.2.3.10	The average bit rate of the transport session.
@maximum	1	Positive integer	10.2.3.10	The maximum bit rate of the transport session.
ForwardErrorCorrection Parameters	0..n		10.2.3.11	Container for the parameters of the FEC repair blocks protecting the multicast gateway configuration transport session.
SchemeIdentifier	1	String	10.2.3.11	A fully-qualified term identifier URI from *ForwardErrorCorrectionSchemeCS* (clause B.2) specifying the scheme of the FEC repair blocks protecting the multicast gateway configuration transport session.
OverheadPercentage	1	Positive integer	10.2.3.11	Percentage of FEC overhead compared with source packets. For example, a value of 20 means 20 % overhead, 100 means there is a repair packet for every source packet. Values greater than 100 are permitted.
EndpointAddress	0..n		10.2.3.9	Container for the addressing parameters of FEC repair blocks protecting this multicast gateway configuration transport session. May be omitted if FEC repair packets are carried in band with the multicast gateway configuration transport session.
NetworkSourceAddress	0..1	String	10.2.3.9	Host name or IP address specifying the source address of FEC repair packets protecting this multicast gateway configuration transport session.
NetworkDestinationGroup Address	1	IP address string	10.2.3.9	An IP address specifying the multicast group for FEC repair packets protecting this multicast gateway configuration transport session.
TransportDestinationPort	1	Unsigned 16-bit integer	10.2.3.9	The UDP port number of FEC repair packets protecting this multicast gateway configuration transport session.
MediaTransportSession Identifier	0..1	Positive integer	10.2.3.9	An opaque transport session identifier of FEC repair packets protecting this multicast gateway configuration transport session.

Element or attribute name	Use	Data type	Clause	Description
UnicastRepairParameters	0..1		10.2.3.12	Container for parameters pertaining to a unicast repair.
@transportObjectBaseURI	0..1	URI string	10.2.3.13	Multicast transport object base URI. The base URI of all multicast transport objects conveyed in this multicast gateway configuration transport session.
@transportObjectReceptionTimeout	1	Unsigned integer	10.2.3.12	The period of time (in milliseconds) that a *Multicast gateway* should wait for a packet relating to a multicast transport object before assuming that the object transmission is over.
@fixedBackOffPeriod	0..1	Unsigned integer	10.2.3.12	The minimum number of milliseconds that the *Multicast gateway* shall wait after the end of object transmission before attempting unicast repair.
@randomBackOffPeriod	0..1	Unsigned integer	10.2.3.12	Maximum number of milliseconds for the random delay period that the *Multicast gateway* shall wait in addition to @fixedBackOffPeriod.
BaseURL	0..n	URI string	10.2.3.13	Unicast repair base URL prefix. The base path of a unicast repair endpoint.
@relativeWeight	0..1	Unsigned integer	10.2.3.13	A relative weighting for this unicast repair endpoint.
ObjectCarousel	0..1		10.2.3.14.0	Container for parameters pertaining to the in-band carousel of ancillary multicast transport objects.
@aggregateTransportSize	0..1	Unsigned integer	10.2.3.14.0	The combined size of all transport objects described by this carousel, excluding any associated metadata and transport protocol overhead.
@aggregateContentSize	0..1	Unsigned integer	10.2.3.14.0	The combined size of all transport objects described by this carousel, excluding any associated metadata and transport protocol overhead and excluding any content encoding (such as compression). May be omitted if no content encoding is applied to the multicast transport objects.
PresentationManifests	0..n		10.2.3.14.1	The presence of this element indicates that presentation manifests shall be conveyed in the multicast gateway configuration transport session when the referenced multicast session is active.
@serviceIdRef	0..1	URI string	10.2.5.0	A reference to the service identifier of a multicast session in the current multicast session configuration whose presentation manifest(s) are to be conveyed when the referenced multicast session is active. Omitting this attribute indicates that presentation manifests for all active multicast sessions are to be conveyed.
@transportSessionIdRef	0..1	URI string	10.2.5.0	A reference to a multicast transport session identifier within the scope of the referenced multicast session whose presentation manifest(s) are to be conveyed when the referenced multicast session is active. Omitting this attribute indicates that presentation manifests for all multicast transport sessions in the referenced multicast session are to be conveyed when it is active. It is an error to supply this attribute if the @serviceIdRef attribute above is omitted.

Element or attribute name	Use	Data type	Clause	Description
@targetAcquisitionLatency	0..1	Duration string	10.2.3.14.0	Repetition rate of presentation manifests in the multicast gateway configuration transport session.
@compressionPreferred	0..1	Boolean	10.2.3.14.0	Expression of preference to the *Multicast server* that compression of the resulting multicast transport object(s) is desired by the *Provisioning* function. Only permitted in multicast server configuration instance documents.
InitSegments	0..n	URI string	10.2.3.14.2	The presence of this element indicates that initialisation segments shall be conveyed in the multicast gateway configuration transport session when the referenced multicast session is active.
@serviceIdRef	0..1	URI string	10.2.5.0	A reference to the service identifier of a multicast session in the current multicast session configuration whose initialisation segment(s) are to be conveyed when the referenced multicast session is active. Omitting this attribute indicates that initialisation segments for all active multicast sessions are to be conveyed.
@transportSessionIdRef	0..1	URI string	10.2.5.0	A reference to a multicast transport session identifier within the scope of the referenced multicast session whose presentation manifest(s) are to be conveyed when the referenced multicast session is active. Omitting this attribute indicates that initialisation segments for all multicast transport sessions in the referenced multicast session are to be conveyed when it is active. It is an error to supply this attribute if the `@serviceIdRef` attribute above is omitted.
@targetAcquisitionLatency	0..1	Duration string	10.2.3.14.0	Repetition rate of initialisation segments in the multicast gateway configuration transport session.
@compressionPreferred	0..1	Boolean	10.2.3.14.0	Expression of preference to the *Multicast server* that compression of the resulting multicast transport object(s) is desired by the *Provisioning* function. Only permitted in multicast server configuration instance documents.
ResourceLocator	0..n	URI string	10.2.3.14.3	The URL of a resource that shall be conveyed in the multicast gateway configuration transport session.
@targetAcquisitionLatency	0..1	Duration string	10.2.3.14.0	Repetition rate of the resource in the multicast gateway configuration transport session.
@revalidationPeriod	0..1	Duration string	10.2.3.14.3	The period of time that a *Multicast server* should wait before revalidating the resource.
@compressionPreferred	0..1	Boolean	10.2.3.14.0	Expression of preference to the *Multicast server* that compression of the resulting multicast transport object(s) is desired by the *Provisioning* function. Only permitted in multicast server configuration instance documents.
MulticastGatewayConfiguration Macro	0..n	String	10.2.5.2	The value to substitute in place of any matched macro key found when the *Multicast server* generates a multicast gateway configuration.

Element or attribute name			Use	Data type	Clause	Description
		@key	1	Name Token string	10.2.5.2	Macro key to be matched in the multicast server configuration when the *Multicast server* generates a multicast gateway configuration.

10.2.5.2 Multicast session configuration macro expansion

In the case where the *Multicast gateway* is configured by means of the in-band configuration method specified in clause 10.1.2, the *Multicast server* is responsible for generating one or more multicast gateway configurations based on its current multicast server configuration. In some deployments, the URLs of objects requested from the *Content hosting* function may differ between reference points **O_{in}** and **A**, and in the case of a *Multicast server* configured to use the push-based content ingest method over reference point **P_{in}'** (as specified in clause 8.3.1) the URLs should be different from those at reference point **A**. In these cases, the *Multicast server* may use the macro definition elements **`MulticastServerConfigurationMacro`** and **`MulticastGatewayConfigurationMacro`** in combination to set the correct reference point **A** URLs in each multicast gateway configuration. These extension elements shall comply with the XML schema definition in clause A.2.

NOTE: Each multicast gateway configuration to be generated is described by a separate **`MulticastGatewayConfigurationTransportSession`** and each such element and may therefore define different expansions of the same macro key.

When the macro definition elements are present in a multicast server configuration instance document, the document may also carry macro keys in any element or attribute of data type URI String. Macro keys are name token strings prefixed and suffixed with a single unescaped "$" delimiter character. When processing the multicast server configuration, the *Multicast server* shall match such keys to the `@key` attribute of a macro definition element and shall replace the macro key, as well as its leading and trailing "$" delimiters, with the content of the matched macro definition element.

- The *Multicast server* shall apply macro expansion to the content of elements and attributes defined as data type URI String in clauses 10.2.2.1 and 10.2.3.1 before attempting to acquire ingest objects referenced by them.
- When generating a multicast gateway configuration from the multicast server configuration, the *Multicast server* shall apply macro expansion to the content of elements and attributes defined as data type URI String in clauses 10.2.2.1, 10.2.3.1 and 10.2.5.1.

The *Multicast server* shall use the original multicast server configuration received over reference point **C_{MS}** when performing macro expansion, and not a previous copy of the multicast server configuration that has already had macro expansion performed on it.

The *Multicast gateway* is not expected to perform any macro expansion. Any macros present in the multicast server configuration shall be fully expanded in the multicast gateway configuration instance document: it is an error for unexpanded macro keys to be present in a multicast gateway configuration.

10.3 Life-cycle of multicast transport sessions

10.3.0 General

Individual multicast transport sessions in the multicast session configuration are in one of two states:

- In the **active** state, multicast media transport protocol packets may be transmitted by the *Multicast server* according to the current set of parameters specifying that multicast transport session.
- In the **inactive** state, multicast media transport protocol packets shall not be transmitted by the *Multicast server*.

A multicast transport session may be permanently in the active state or it may be (de)activated by one of two different methods, described in the following clauses.

10.3.1 Timed activation of multicast transport session

A multicast session configuration document may include parameters specifying a start time and/or duration for any of the multicast transport sessions it describes, as specified in clause 10.2.3.3.

In the general case, a multicast transport session shall become active in the deployed system at the indicated start time. Once active, a multicast transport session shall remain active for a period of time equal to that indicated by its duration.

If the start time is in the past, the multicast transport session shall be deemed to be active immediately on receipt of the multicast session configuration document unless the start time plus the duration is also in the past, in which case the multicast transport session shall be inactive.

If a start time is specified but no duration, the multicast transport session shall become active at the specified start time and shall remain active until further notice.

If a duration is specified but no start time, the multicast transport session shall be deemed to be active immediately on receipt of the multicast session configuration document and shall remain active for a period of time equal to the time of receipt plus the duration.

If neither a start time nor a duration is specified for it, a multicast transport session shall remain in its current state until this state is modified by manual (de)activation, as specified in clause 10.3.2. The initial state for such multicast transport sessions shall be inactive.

NOTE: If it is important that all instances of the *Multicast gateway* have a common view of the end time of a multicast transport session, both the `@start` and `@duration` attributes should be supplied with values in the multicast gateway configuration instance document.

10.3.2 Manual (de)activation of multicast transport session

Regardless of whether a start time and duration has been specified in the current multicast session configuration document, a multicast transport session may be activated by the procedure specified in clause 10.4.3.1 and deactivated by the procedure specified in clause 10.4.3.2.

In both cases, the multicast transport session(s) affected are indicated by quoting the value of their `@id` attribute in combination with the `@serviceIdentifier` of their parent multicast session.

10.4 Configuration and control procedures

10.4.0 General

The procedures for configuring and controlling system functions at reference points **C_{MS}** and **C_{MR}** shall use HTTP request and response messages, as specified in IETF RFC 9110 [2]. The use of TLS [7] to secure the interactions is recommended. Implementations shall support at least HTTP/1.1 [1] and may additionally support HTTP/2 [34] and/or HTTP/3 [35].

The structure of target resource URI paths in HTTP requests shall follow the following general format:

{rootPrefix}/*{endpointName}*/*{endpointVersion}*/*{endpointSpecificSuffix}*

where *{rootPrefix}* shall be an implementation-specific URI prefix according to the generic syntax specified in IETF RFC 3986 [15] with either *http* or *https* as the scheme part. *{endpointSpecificSuffix}* is a set of one or more path elements and/or query parameters specified in the following clauses.

10.4.1 Error responses

When an error occurs as the result of a request at reference point **C_{MS}** or **C_{MR}**, the most appropriate status code from table 10.4.1-1 below shall be returned to the requestor:

Table 10.4.1-1: HTTP error response status codes

Response status code and reason phrase	Applicable methods	Summary description	Normative specification
400 Bad Request	`GET`, `PUT`, `POST`	The parameters supplied in the request were incorrect for the endpoint specified by the target resource URI.	IETF RFC 9110 [2], section 15.5.1
401 Unauthorized	`GET`, `PUT`, `POST`	The request lacks a valid set of authentication credentials in the `Authorization` request header to access the target resource URI.	IETF RFC 9110 [2], section 15.5.2
403 Forbidden	`GET`, `PUT`, `POST`	The request message is well formed, and any authentication credentials supplied are valid, but the endpoint refuses to authorize the request for other reasons.	IETF RFC 9110 [2], section 15.5.4
404 Not Found	`GET`, `PUT`, `POST`	The target resource URI in the request message is not recognized by the endpoint.	IETF RFC 9110 [2], section 15.5.5
405 Method Not Allowed	(Any)	The method in the request is not supported by the target resource URI.	IETF RFC 9110 [2], section 15.5.6
406 Not Acceptable	`GET`	The endpoint was unable to fulfil the request within the constraints of the supplied content negotiation request headers.	IETF RFC 9110 [2], section 15.5.7
411 Length Required	`PUT`	The endpoint refuses to accept a request without a `Content-Length` header.	IETF RFC 9110 [2], section 15.5.12
413 Content Too Large	`PUT`	The endpoint refuses to accept the request because the request content size declared in the `Content-Length` header is too large, or because the request content exceeds the indicated length.	IETF RFC 9110 [2], section 15.5.14
415 Unsupported Media Type	`PUT`	The `Content-Type` request header did not contain a value required by the specified endpoint.	IETF RFC 9110 [2], section 15.5.16
429 Too Many Requests	`GET`, `PUT`, `POST`	The endpoint is experiencing high load that prevented it from fulfilling the request. The `Retry-After` response header (see section 10.2.3 in [2]) may be provided in the response message to indicate how long the client should wait before reattempting the request.	IETF RFC 6585 [16], section 4
500 Internal Server Error	`GET`, `PUT`, `POST`	The endpoint encountered an unexpected error that prevented it from fulfilling the request.	IETF RFC 9110 [2] section 15.6.1
503 Service Unavailable	`GET`, `PUT`, `POST`	The endpoint was unable to process the request. The `Retry-After` response header (see section 10.2.3 in [2]) may be provided in the response message to indicate how long the client should wait before reattempting the request.	IETF RFC 9110 [2], section 15.6.4

10.4.2 Multicast server configuration procedures

10.4.2.0 General

This clause specifies the procedures for configuring a *Multicast server* at reference point **C_{MS}**:

- The *{endpointName}* URL element shall be dvb-multicast-server.
- The *{endpointVersion}* URL element shall be v1.

10.4.2.1 Out-of-band pushed multicast server configuration method

This procedure is invoked by a *Network control* subfunction on a *Multicast server* function to push a new multicast server configuration to the latter. To this end, the *Multicast server* function shall expose a RESTful resource [i.8] at reference point **C_{MS}** representing its current configuration.

The *{endpointSpecificSuffix}* URL element shall be the path element configuration, identifying the RESTful resource to be manipulated.

Request message	
Request method	*PUT*
Target resource URI	*{rootPrefix}*/dvb-multicast-server/v1/configuration
`Content-type` request header	*text/xml*
Request message body	A multicast server configuration instance document according to the definition in clause 10.0 and the syntax specified in clause 10.2.
Response message	
Success status code	*204 No Content*
Success response message body	Empty

A success status code in the response message signifies that the multicast server configuration has been replaced with that contained in the instance document supplied in the request message body.

10.4.2.2 Out-of-band pulled multicast server configuration method

This procedure is invoked by a *Multicast server* function on a *Network control* subfunction to retrieve the latest multicast server configuration from the latter. To this end, the *Network control* subfunction shall expose a RESTful resource [i.8] at reference point **C_{MS}** representing the current multicast server configuration in the system.

NOTE: Configuration of the *{rootPrefix}* to be used by the *Multicast server* is beyond the scope of the present specification.

The *{endpointSpecificSuffix}* URL element shall be the path element configuration, identifying the RESTful resource made available for retrieval.

Request message	
Request method	`GET`
Target resource URI	*{rootPrefix}*/dvb-multicast-server/v1/configuration
Request message body	Empty
Response message	
Success status code	*200 OK*
Success `Content-type` request header	*text/xml*
Success response message body	A multicast server configuration instance document according to the definition in clause 10.0 and the syntax specified in clause 10.2.

A success status code in the response message signifies that the current multicast server configuration is represented by the instance document supplied in the response message body.

10.4.3 Multicast server control procedures

10.4.3.0 General

This clause specifies the procedures for controlling a *Multicast server* at reference point **C_{MS}**:

- The *{endpointName}* URL element shall be *dvb-multicast-server*.
- The *{endpointVersion}* URL element shall be *v1*.

10.4.3.1 Multicast transport session activation

This procedure is invoked by a *Network control* subfunction on a *Multicast server* function to activate a single multicast transport session or to activate all multicast transport sessions in a particular multicast session. For this purpose, the *Multicast server* function shall expose a remote procedure call at reference point **C_{MS}**.

The *{endpointSpecificSuffix}* URL element shall be the path element activate, the name of the remote procedure call:

- To activate a single multicast transport session, the first form of the target resource URI below shall be used, indicating the service identifier of the parent multicast session (specified in clause 10.2.2.0) as the value of the *service* query parameter and the transport session identifier (specified in clause 10.2.3.2) as the value of the *transport-session* query parameter.
- To activate all currently inactive multicast transport sessions in a particular multicast session, the second form of the target resource URI below shall be used, indicating only the service identifier of the multicast session (clause 10.2.2.0) as the value of the *service* query parameter.

Request message	
Request method	`POST`
Target resource URI *(first form)*	*{rootPrefix}*/dvb-multicast-server/v1/activate? service=*{service-id}*&transport-session=*{transport-session-id}*
Target resource URI *(second form)*	*{rootPrefix}*/dvb-multicast-server/v1/activate?service=*{service-id}*
`Content-type` request header	Omitted
Request message body	Empty
Response message	
Success status code	*204 No Content*
Success response message body	Empty
NOTE: URI query components are required to comply with the *query* production specified in Appendix A of [15].	

A success status code in the response message signifies that the multicast transport session(s) specified in the target resource URI of the request is/are now in the active state.

10.4.3.2 Multicast transport session deactivation

This procedure is invoked by a *Network control* subfunction on a *Multicast server* function to deactivate a single multicast transport session or to deactivate all multicast transport sessions in a particular multicast session. For this purpose, the *Multicast server* function shall expose a remote procedure call at reference point **C_{MS}**.

The *{endpointSpecificSuffix}* URL element shall be the path element deactivate, the name of the remote procedure call.

- To deactivate a single multicast transport session, the first form of the target resource URI below shall be used, indicating the service identifier of the parent multicast session (specified in clause 10.2.2.0) as the value of the *service* query parameter and the transport session identifier (specified in clause 10.2.3.2) as the value of the *transport-session* query parameter.
- To deactivate all currently active multicast transport sessions in a particular multicast session, the second form of the target resource URI below shall be used, indicating only the service identifier of the multicast session (clause 10.2.2.0) as the value of the *service* query parameter.

Request message	
Request method	`POST`
Target resource URI *(first form)*	*{rootPrefix}*/dvb-multicast-server/v1/deactivate? service=*{service-id}*&transport-session=*{transport-session-id}*
Target resource URI *(second form)*	*{rootPrefix}*/dvb-multicast-server/v1/deactivate?service=*{service-id}*
`Content-type` request header	Omitted
Request message body	Empty
Response message	
Success status code	*204 No Content*
Success response message body	Empty
NOTE: URI query components are required to comply with the *query* production specified in appendix A of [15].	

A success status code in the response message signifies that the multicast transport session(s) specified in the target resource URI of the request is/are now in the inactive state.

10.4.4 Multicast gateway configuration procedures

10.4.4.0 General

This clause specifies the procedures for configuring a *Multicast gateway* at reference point **C_{MR}**:

- The *{endpointName}* URL element shall be dvb-multicast-gateway.
- The *{endpointVersion}* URL element shall be v1.

10.4.4.1 Out-of-band pushed multicast gateway configuration method

This procedure is invoked by a *Network control* subfunction on a *Multicast gateway* function to push a new multicast gateway configuration to the latter. To this end, the *Multicast gateway* function shall expose a RESTful resource [i.8] at reference point **C_{MR}** representing its current configuration.

The *{endpointSpecificSuffix}* URL element shall be the path element configuration, identifying the RESTful resource to be manipulated.

Request message	
Request method	`PUT`
Target resource URI	*{rootPrefix}*/dvb-multicast-gateway/v1/configuration
`Content-type` request header	*text/xml*
Request message body	A multicast gateway configuration instance document according to the definition in clause 10.0 and the syntax specified in clause 10.2.
Response message	
Success status code	*204 No Content*
Success response message body	Empty

A success status code in the response message signifies that the multicast gateway configuration has been replaced with that contained in the instance document supplied in the request message body.

10.4.4.2 Out-of-band pulled multicast gateway configuration method

This procedure is invoked by a *Multicast gateway* function on a *Network control* subfunction to retrieve the latest multicast gateway configuration from the latter. To this end, the *Network control* subfunction shall expose a RESTful resource [i.8] at reference point **C_{MR}** representing the current multicast gateway configuration in the system.

NOTE: Configuration of the *{rootPrefix}* to be used by each *Multicast gateway* instance is beyond the scope of the present document.

The *{endpointSpecificSuffix}* URL element shall be the path element configuration, identifying the RESTful resource made available for retrieval.

The request target may optionally include query parameters for dynamic *Multicast gateway* registration as specified in clause 7.6, presented below as the second form of target resource URI.

Request message	
Request method	`GET`
Target resource URI (first form)	*{rootPrefix}*/dvb-multicast-gateway/v2/configuration
Target resource URI (second form)	*{rootPrefix}*/dvb-multicast-gateway/v2/configuration? [content-playback-subnet=*{player-subnet}*&] [redirect-base-url=*{gateway-base-url}*&]…
Request message body	Empty
Response message	
Success status code	*200 OK*
Success `Content-type` request header	*text/xml*
Success response message body	A multicast gateway configuration instance document according to the definition in clause 10.0 and the syntax specified in clause 10.2.

A success status code in the response message signifies that the current multicast gateway configuration is represented by the instance document supplied in the response message body.

10.4.5 In-band multicast gateway configuration method

When this method is used to configure a *Multicast gateway* (or *Multicast Rendezvous service*) the *Multicast server* shall transmit a multicast gateway configuration transport session as specified in clause 8.3.5. The parameters of this transport session need to be communicated to the *Multicast gateway* so that it can subscribe to the appropriate multicast endpoint address(es). The parameters may be encapsulated in a simple "bootstrap" multicast gateway configuration instance document containing only a **`MulticastGatewayConfigurationTransportSession`** element, as specified in clause 10.2.5. This document may be provided at reference point **C_{MR}** using the out-of-band pushed method (clause 10.4.4.1 above) or the out-of-band pulled method (clause 10.4.4.2 above) or through an alternative configuration method such as TR-069 [i.9] or TR-369 [i.10].

11 Reporting interactions

11.0 Overview

According to clause 5.3.5.5, the *Multicast gateway* includes a *Service reporting* subfunction (see also figure 5.2-1). As explained in clause 5.3.6.1, this supplies service reporting information via reference point **R_S** to the *Service reporting capture* function which may, in turn, export service reporting information to the *Content Provider reporting capture* function via reference point **R_{CP}**.

The configuration for the *Service reporting* subfunction is described in clauses 10.2.1, 10.2.2.1 and 10.2.2.3. The session reporting parameters govern the way the *Multicast gateway* provides service reporting information (e.g. the sample of *Multicast gateway* instances that report and the periodicity of reporting).

This clause defines the set of metrics and events to be exposed as service reporting information over either reference point **R_S** or **R_{CP}**. The metrics exposed are cumulative statistics since the beginning of the reported playback session(s).

Service reporting information is collated by the *Multicast gateway* and formatted as a service reporting information instance document, the syntax of which is specified in clause 11.1. Depending on the multicast session configuration, each service reporting information instance document may aggregate metrics and/or events for current playback sessions relating to all multicast sessions (see clause 10.2.1) or for playback sessions relating to one particular multicast session (see clause 10.2.2.3).

Service reporting information instance documents are sent by the *Service reporting* subfunction of the *Multicast gateway* to a *Service reporting capture* subfunction of the *Provisioning* function using the protocol specified in clause 11.2. As specified in clause 10.2.2.3, the *Service reporting* subfunction submits reporting instance documents either regularly (on the basis of the reporting period) and/or detection of events.

The instance document reports a series of events related to one or more playback sessions. The service reporting information is described by three top-level event types that indicate the state of a playback session:

1. A **Playback session started** event is generated whenever the *Content playback* function interacts with the *Multicast gateway* after having been redirected by the *Multicast rendezvous service*.

2. A **Playback session ended** event is generated whenever an existing playback session ends due to either the end of the media presentation or the session being terminated.

3. A **Playback session running** event corresponds to any other moment the *Multicast gateway* needs to submit a new report according to the configured reporting period or as a result of one of the events specified in clause 11.1.2.2.

11.1 Service reporting information instance document format

11.1.0 General

The reporting information instance document shall be formatted according to JSON [36].

11.1.1 Service reporting information instance document syntax

The syntax of the service reporting information instance document is specified in annex N and the semantics are specified in table 11.1.1-1 below.

Table 11.1.1-1: Service reporting information instance document syntax

Object or property name	Use	Data type	Clause	Description
report				
timestamp	1	MPEG-7 *TimePoint*	11.1.2.1	Date/time (UTC) when this service reporting information instance document was generated.
gateway-id	1	String	11.1.2.1	Opaque identifier that uniquely identifies the *Multicast gateway* instance in the system.
gateway-description	0..1	String	11.1.2.1	Description of the *Multicast gateway* instance.
events	1..n	Array of object literals	11.1.2.2	A set of events to be reported.
timestamp	1	MPEG-7 *TimePoint*	11.1.2.2	Date/time (UTC) when this event occurred.
playback-session-id	1	String	11.1.2.2	Unique identifier for the playback session, possibly communicated by the *Content playback* function. Associates the event with a playback session reported in the *sessions* list below.
type	1	String (enum)	11.1.2.2	Type of event: *session-started*, *heartbeat*, *service-component-switch*, *multicast-join*, *multicast-leave*, *object-delivery*, *session-ended*.
object-delivery-status	0..1	String (enum)	11.1.2.2	Delivery status of a media object. Present only if the event type is *object-delivery*.
service-component-identifier	0..1	Object	11.1.2.2	An object that uniquely identifies a service component in the presentation manifest associated with this playback session. Present only if the event type is *service-component-switch*.
multicast-endpoint-address	0..1	Object	11.1.2.2	Present only if the event type is *multicast-join* or *multicast-leave*.
source	0..1	String	11.1.2.2	If the multicast transport session uses source-specific multicast transport, the host name or IP address specifying the multicast group source address.
group	1	String	11.1.2.2	IP address specifying the multicast group for the multicast transport session.
port	1	Unsigned Integer	11.1.2.2	UDP port number of the multicast transport session.
transport-session-id	0..1	Positive Integer	11.1.2.2	Multicast transport session identifier, if configured.
playback-sessions	1..n	Array of object literals	11.1.2.3	Table of playback sessions currently served by the *Multicast gateway*.

Object or property name	Use	Data type	Clause	Description
playback-session-id	1	URI string	11.1.2.3	Unique identifier for the playback session, possibly communicated by the *Content playback* function.
service-id	1	URI string	11.1.2.3	Service identifier of the logical service.
manifest-id	1	String	11.1.2.3	Unique identifier of the presentation manifest used by the *Multicast gateway* to deliver the presentation session.
manifest-url	1	URI string	11.1.2.3	Presentation manifest URL requested by the *Content playback* function at reference point **L**.
user-agent	0..1	String	11.1.2.3	Unspecified string identifying the user agent.
in-session-metrics	0..1	Object	11.1.2.4	Present only for **Playback session running** or **Playback session ended** events.
errors-b	0..1	Unsigned Integer	11.1.2.4	Number of errors so far during the playback session at reference point **B**.
errors-l	0..1	Unsigned Integer	11.1.2.4	Number of errors returned by the *Multicast gateway* instance so far during the playback session at reference point **L** during the playback session.
errors-a	0..1	Unsigned Integer	11.1.2.4	Number of errors so far during the playback session at reference point **A**.
errors-m	0..1	Unsigned Integer	11.1.2.4	Number of errors so far during the playback session at reference point **M**.
errors-u	0..1	Unsigned Integer	11.1.2.4	Number of errors so far during the playback session at reference point **U**.
cache-rep-a	1	Unsigned Integer	11.1.2.4	Number of bytes of unicast repair data received so far during the playback session at reference point **A**.
cache-rep-f	1	Unsigned Integer	11.1.2.4	Number of bytes of lost multicast transport object data successfully recovered so far during the playback session through the use of AL-FEC.
cache-rep-u	0..1	Unsigned Integer	11.1.2.4	Number of bytes of unicast repair data received so far during the playback session at reference point **U**.
rep-switch-nb	1	Unsigned Integer	11.1.2.4	Number of representation changes so far during the playback session.
segment-req-nb	0..1	Unsigned Integer	11.1.2.4	Number of media object requests received so far during the playback session at reference point **L**.
bytes-received-m	1	Unsigned Integer	11.1.2.4	Number of bytes received so far during the playback session at reference point **M**.
bytes-received-a	1	Unsigned Integer	11.1.2.4	Number of bytes received so far during the playback session at reference point **A**.
bytes-received-l	0..1	Unsigned Integer	11.1.2.4	Number of bytes received so far during the playback session at reference point **L**.
bytes-sent-l	0..1	Unsigned Integer	11.1.2.4	Number of bytes delivered so far during the playback session at reference point **L**.
session-end-metrics	0..1	Object	11.1.2.5	Present only for playback session ended events.
service-component-information	1..N	Array of object literals	11.1.2.5	
service-component-identifier	1	Object	11.1.2.5	An object that uniquely identifies a service component in the presentation manifest associated with the playback session.
multicast-endpoint-address	0..N	Array of object literals	11.1.2.5	Multicast endpoint address of the service component.
source	0..1	String	11.1.2.5	Host name or IP address of the multicast group source. Present only for source-specific multicast.
group	1	String	11.1.2.5	IP address of the multicast group for the multicast transport session.
port	1	Unsigned Integer	11.1.2.5	UDP destination port number of the multicast transport session.
transport-session-id	0..1	Positive Integer	11.1.2.5	Multicast transport session identifier, if configured.
segment-duration	0..1	Positive integer	11.1.2.5	Duration of segments in milliseconds as declared in the presentation manifest for the service component.

Object or property name	Use	Data type	Clause	Description
total-bytes	1	Unsigned Integer	11.1.2.5	Number of bytes delivered over reference point **L** for the service component.
bit-rate	1	Positive Integer	11.1.2.5	The bit rate of the service component as conveyed in the presentation manifest, expressed in bits per second.
cache-miss-expired	1	Unsigned Integer	11.1.2.5	Total number of media objects for this service component retrieved at reference point **A** because the media object expired from the *Asset storage* subfunction, for example because playback is significantly behind the multicast live edge.
cache-miss-incomplete	1	Unsigned Integer	11.1.2.5	Total number of media objects retrieved at reference point **A** because they could not be repaired using reference points **A** and/or **U** after being only partly received at reference point **M**.
cache-miss-filter	1	Unsigned Integer	11.1.2.5	Total number of media objects fetched at reference point **A** and corresponding to an inactive multicast transport session.
cache-miss-nodata-m	1	Unsigned Integer	11.1.2.5	Total number of media objects fetched at reference point **A** because no data related to that playback session was received via reference point **M**.
cache-miss-nodata-s	1	Unsigned Integer	11.1.2.5	Total number of media objects fetched at reference point **A** because they were not received via reference point **M** although other data was received in relation with that playback session.
cache-miss-nodata-j	1	Unsigned Integer	11.1.2.5	Total number of media objects fetched at reference point **A** before the *Multicast gateway* subscribed to the multicast transport session.
cache-miss-nodata-o	1	Unsigned Integer	11.1.2.5	Total number of media objects fetched at reference point **A** due to any other unspecified reason.
cache-miss-nodata-l	0..1	Unsigned Integer	11.1.2.5	Total number of requests at reference point **L** that the *Multicast gateway* could not fulfil because the requested playback delivery object was not received in time via reference point **M** and nor could it be retrieved via reference point **A**.
cache-hit-m	1	Unsigned Integer	11.1.2.5	Total number of media objects retrieved from the *Asset storage* cache received via reference point **M**.
cache-hit-a	1	Unsigned Integer	11.1.2.5	Total number of media objects retrieved from the *Asset storage* cache received via reference point **A**.
cache-hit-mr	1	Unsigned Integer	11.1.2.5	Total number of media objects retrieved from the *Asset storage* cache received via reference point **M** but repaired over reference point **A** or reference point **U**.

11.1.2 Reporting elements

11.1.2.0 Introduction

This clause describes the reporting elements that comprise the structure of a service reporting information instance document.

11.1.2.1 Report information

The document root object *report* provides general information about the *Multicast gateway* that is reporting, including its identifier, and the date at which the report was generated.

The value of *gateway-id* shall be assigned by the System operator.

A textual description of the *Multicast gateway* may be reported in the *gateway-description* property.

11.1.2.2 Event information

The *events* array shall provide a list of reporting events and carries several name–value pairs. Depending on the event type, it may also carry additional JSON objects, as specified below.

The *timestamp* property shall carry the date and time that this event was generated by the *Multicast gateway*. The value of this property shall comply with the *TimePoint* datatype specified in clause 6.4.3 of MPEG-7 Part 5 [12].

The *playback-session-id* property is a unique identifier for the playback session that is used to associate the event with one of the playback sessions listed in the same service reporting instance document (see clause 11.1.2.3).

The type of event shall be indicated by the value of the *type* property as follows:

1. A **Playback session started** event shall be identified by the *type* value *session-started*, and the element of the *events* array shall contain no additional objects.

2. A **Playback session ended** event shall be identified by the *type* value *session-ended*, and the element of the *events* array then contains *session-end-metrics* object as described in clause 11.1.2.4 and may contain additional *in-session-metrics* object as described in clause 11.1.2.5.

3. **Playback session running** events shall be identified by the *type* values listed below, and the member of the *events* array describing the event shall include an *in-sesssion-metrics* element as described in clause 11.1.2.3.

    a) *heartbeat*: Sent periodically, according to the reporting configuration. This event is used when there is no other event to report.

    b) *service-component-switch*: The playback session switched to a different service component. This event is accompanied by the *service-component-identifier* object indicating which service component the *Multicast gateway* switched to.

    c) *multicast-join*: The *Multicast gateway* joined a multicast group. This event is accompanied by a *multicast-endpoint-address* object indicating the multicast group joined.

    d) *multicast-leave*: The *Multicast gateway* left a multicast group. This event is accompanied by a *multicast-endpoint-address* object indicating the multicast group left.

    e) *object-delivery*: This event is accompanied by an enumerated value *object-delivery-status* as specified in table 11.1.2.2-1 below, indicating how the media object was delivered to the *Multicast gateway*.

Table 11.1.2.2-1: Media object delivery status enumeration

Status	Description
cache-hit-m	Media object was completely received intact via reference point **M**.
cache-hit-a	Media object was already present in the cache having been previously fetched from unicast via reference point **A**.
cache-hit-mr	Media object was partly received via reference point **M** but was successfully repaired via reference point **A** or reference point **U**.
cache-miss-expired	Media object was fetched at reference point **A** because, although present in the *Asset storage* subfunction, it had expired.
cache-miss-incomplete	Media object was partly received at reference point **M** but cannot be repaired because the repair service is not available.
cache-miss-filter	Media object was fetched at reference point **A** during an inactive multicast session.
cache-miss-nodata-s	Media object was fetched at reference point **A** because no data for it was received via reference point **M**.
cache-miss-nodata-m	Media object was fetched at reference point **A** because no data for it was received via reference point **M** although other data was received in relation to the playback session in question.
cache-miss-nodata-j	Media object was fetched at reference point **A** before the *Multicast gateway* subscribed to the multicast transport session.
cache-miss-nodata-o	Media object was fetched at reference point **A** for any other reason.

11.1.2.3 Session information

The *playback-sessions* array shall provide a list of metrics associated with each ongoing playback session at the reporting *Multicast gateway*. Each member of this array shall be identified using a *playback-session-id* property which carries a unique session identifier for the playback session.

NOTE 1: The assignment of this value is implementation-specific and out of scope for the present document.

NOTE 2: Members of the *events* array (see clause 11.1.2.2) refer to a playback session using the *playback-session-id* value.

The service identifier for the multicast session that this playback session pertains to shall be carried in the *service-id* property. This shall be the value carried by the **`MulticastSession`**`@serviceIdentifier` attribute in the multicast session configuration, as specified in clause 10.2.2.0.

The unique identifier of the presentation manifest that the *Multicast gateway* used to deliver the presentation session shall be carried in the *manifest-id* property. This shall be the value carried by the **`PresentationManifestLocator`** `@manifestId` attribute in the multicast session configuration, as specified in clause 10.2.2.2.

The URL used by the *Content playback* function to request the presentation manifest at reference point **L** for this presentation session shall be carried in the *manifest-url* property.

The *user-agent* property may be present and carries the user agent string of the *Content playback* function presented at reference point **L**, as specified in section 5.5.3 of [2].

11.1.2.4 In-session metrics

When a given event has a *type* property indicating that it is a **Playback session running** event as specified in clause 11.1.2.2, the event shall contain an *in-session-metrics* object. This object carries metrics covering a single presentation session. The metrics are accumulated since the beginning of the presentation session.

- The *errors-b* property is only used when the *Multicast rendezvous service* function is co-located with the *Multicast gateway* as described in clause 7.2, and reports the number of errors detected so far by the *Multicast rendezvous service* at reference point **B**. Any request by the *Content playback* function that either fails due to a connection error or which results in the *Multicast rendezvous service* generating a HTTP response code in the 4xx client error or 5xx server error range is considered an error.

- The *errors-l* property reports the number of errors detected so far by the *Multicast gateway* at reference point **L**. An error is counted whenever an attempt to return a requested object failed due to either a connection error or due to returning an HTTP response code in the 4xx client error or 5xx server error range.

- The *errors-a* property reports the number of errors detected so far by the *Multicast gateway* at reference point **A**. Any request by the *Multicast gateway* that either fails due to a connection error or receives a response from the *Content hosting* function with a HTTP response code in the 4xx client error or 5xx server error range is considered an error.

- The *errors-m* property reports the number of errors detected so far by the *Multicast gateway* at reference point **M**. An error is counted whenever an attempt to receive an object at reference point **M** fails (e.g. IGMP failure, multicast interface is down).

- The *errors-u* property reports the number of errors detected so far by the *Multicast gateway* at reference point **U**. The definition of an error at reference point **U** is dependent on the unicast repair protocol used, and shall be defined by the multicast transport protocol in use.

- The *cache-rep-f* property reports the number of multicast transport object bytes recovered using AL-FEC repair data received at reference point **M** as described in clause 8.3.4.

- The *cache-rep-u* property reports the number of multicast transport object bytes recovered from the *Unicast repair service* at reference point **U**.

- The *rep-switch-nb* property reports the number of representation switches.

- The *segment-req-nb* property reports the number of media object requests received by the *Multicast gateway* from the *Content playback* function at reference point **L**.

- The *bytes-received-m* property reports the total number of multicast transport protocol bytes received by the *Multicast gateway* from the *Multicast server* at reference point **M**.

- The *bytes-received-a* property reports the total number of application protocol bytes (i.e., headers, body and framing) received by the *Multicast gateway* from the *Content hosting* function in HTTP response messages at reference point **A**.

- The *bytes-received-l* property reports the total number of application protocol bytes (i.e., headers, bodies and framing) received by the *Multicast gateway* in HTTP request messages from the *Content playback* function at reference point **L**.

- The *bytes-sent-l* property reports the total number of application protocol bytes (i.e., headers, bodies and framing) sent by the *Multicast gateway* to the *Content playback* function in HTTP response messages at reference point **L**.

11.1.2.5 Session end metrics

The *session-end-metrics* element shall be present when the *Multicast gateway* considers a given streaming session has ended. The *session-end-metrics* element shall contain an array of *service-component-information* objects, with each element representing a service component in the presentation manifest that was consumed during the streaming session. The service component shall be identified by means of a *service-component-identifier* child object of the *service-component-info* object.

In the case where the service component is part of an MPEG-DASH presentation, the *service-component-identifier* shall be realised by a *dash-component-identifier* object. The syntax of this object is specified in table 11.1.2.5-1 below.

Table 11.1.2.5-1: dash-component-identifier syntax

Element or attribute name	Use	Data type	Description
`dash-component-identifier`			Container unambiguously identifying an MPEG-DASH representation.
`period-id`	1	String	A **`Period/`**`@id` from the MPD referenced in the `manifest-id` attribute of the parent structure.
`adaptation-set-id`	1	Unsigned integer	An **`AdaptationSet/`**`@id` from the MPD referenced in the `manifest-id` attribute of the parent structure.
`representation-id`	1	String	A **`Representation/`**`@id` from the MPD referenced in the `manifest-id` attribute of the parent structure.
NOTE: All **`Period`** elements in an MPD of type *dynamic* contain the optional `@id` attribute, as required by clause 5.3.2.2 of the MPEG-DASH specification [i.2]).			

In the case where the service component is part of a HLS presentation, the *service-component-identifier* shall be realised by a *hls-component-identifier* object. The syntax of this object is specified in table 11.1.2.5-2 below.

Table 11.1.2.5-2: hls-component-identifier syntax

Element or attribute name	Use	Data type	Description
`hls-component-identifier`			Container identifying an HLS Media Playlist.
`media-playlist-locator`	1	URI string	The absolute URL of an HLS Media Playlist appearing in the HLS Master Playlist referenced in the `manifest-id` attribute of the parent structure.

Each *service-component-information* object in the array shall also contain the following metadata properties:

- The *bit-rate* property as published in the presentation manifest.
- The optional *multicast-endpoint-address* property is an array of object literals which mirrors the **`EndpointAddress`** element in the multicast transport session configuration described in clause 10.2.3.9, and each object literal in the array carries the following key/value pairs:
  - The *source* property shall be present only when the multicast transport session for the session uses source-specific multicast. It shall carry a host name or string literal specifying the multicast group source address that the *Multicast gateway* used to retrieve objects for this service component.
  - The *group* property shall always be present and shall carry the multicast group address used to retrieve objects for this service component.
  - The *port* property shall always be present and shall carry the UDP port number of the multicast group used to retrieve objects for this service component.
  - The *transport-id* property is only present when the multicast transport session used a multicast media transport protocol session identifier, as specified in clause 10.2.3.9.
- The *segment-duration* property as published in the presentation manifest for this service component.

Each *service-component-information* object in the array shall also report the following metrics:

- The *cache-miss-expired* property indicates the number of times a *Content playback* function requested a media segment belonging to this service component that was too old given the availability indicated in the presentation manifest.
- The *cache-miss-incomplete* property indicates the total number of media objects belonging to this service component fetched by the *Multicast gateway* at reference point **A** because they were partly received at reference point **M** and could not be repaired using reference points **A** and/or **U**.
- The *cache-miss-filter* property indicates the number of media objects request received while the multicast transport session corresponding to this service component was inactive (per clause 10.3). When the *Multicast gateway* does not know the status (i.e. active or inactive) of the related multicast transport session then it shall assume that the transport session is active and therefore the *cache-miss-nodata-m* counter shall be used instead.
- The *cache-miss-nodata-m* property indicates the total number of media objects belonging to this service component fetched by the *Multicast gateway* at reference point **A** because no data related to that playback session was received via reference point **M**.
- The *cache-miss-nodata-s* property indicates the total number of media objects belonging to this service component fetched by the *Multicast gateway* at reference point **A** because they were not received via reference point **M** although other data was received via reference point **M** for the playback session.
- The *cache-miss-nodata-j* property indicates the total number of media objects belonging to this service component fetched by the *Multicast gateway* at reference point **A** because the *Multicast gateway* had not yet subscribed to the multicast transport session.
- The *cache-miss-nodata-o* property indicates the total number of media objects belonging to this service component fetched by the *Multicast gateway* at reference point **A** due to any other unspecified reason.

- The *cache-miss-nodata-l* property indicates the total number of requests at reference point **L** in relation to this service component that the *Multicast gateway* could not fulfil because the requested playback delivery object was neither received in time via reference point **M** nor could it be retrieved via reference point **A**.
- The *cache-hit-m* property corresponds to the total number of media objects retrieved from the *Asset storage* cache in relation to this service component that were received via reference point **M**.
- The *cache-hit-a* property corresponds to the total number of media objects retrieved from the *Asset storage* cache in relation to this service component that were received via reference point **A**.
- The *cache-hit-mr* property corresponds to the total number of media objects retrieved from the *Asset storage* cache in relation to this service component that were received via reference point **M** and repaired via reference point **A** and/or reference point **U**.
- The *total-bytes* property corresponds to the number of application protocol bytes (i.e., headers, bodies and framing) that were delivered in HTTP request and response messages over reference point **L** for this service component.

11.2 Reporting procedure

11.2.0 Protocol

Service reporting instance documents shall be submitted to the *Service reporting capture* function at reference point $\mathbf{R_S}$ or to the *Content Provider reporting capture* function via reference point $\mathbf{R_{CP}}$ using the HTTP `POST` method according to the format specified in clause 11.1.

Reference points $\mathbf{R_{CP}}$ or $\mathbf{R_S}$ shall support HTTPS.

11.2.1 Request URL format

The HTTP request URL shall comply with the following syntax:

http[s]://<*Host*>/dvb/mabr/reportingInformationInstance

For example:

```
POST /dvb/mabr/reportingInformationInstance HTTP/1.1
Host: dvb.example
Content-Type: application/json
Content-Length : <size>

<JSON formatted reporting information instance document>
```

12 Security

12.1 Integrity protection of multicast transport objects

12.1.0 Introduction

The integrity of multicast transport objects may be protected using HTTP digest fields as specified in RFC 9530 [20], adapted to work in the context of multicast delivery. The profile of [20] is specified in clause 12.1.1, and a further profile of [20] for use with individual HTTP chunks is specified in clause 12.1.2. Additional security considerations are specified in clause 12.1.3.

12.1.1 Profile of HTTP digest fields

RFC 9530 [20] is profiled for use in unidirectional multicast delivery as follows:

- The *Multicast gateway* and *Unicast repair service* functions shall support at least the *sha-256* and *sha-512* hash algorithms as specified in the IANA registry of Hash Algorithms for HTTP Digest Fields [42].
- Because the *Multicast gateway* does not make an explicit request for a media object to the *Multicast server*, the `Want-Repr-Digest` and `Want-Content-Digest` HTTP fields shall not be used.
- The `Content-Digest` HTTP field shall be used to carry a message digest pertaining to the payload data conveyed in the multicast transport object after applying any content encoding signalled in a `Content-Encoding` HTTP header. The `Content-Digest` HTTP field may be carried in a trailer section of the HTTP message per section 6.5 of RFC 9110 [2].
- Where the payload data of the multicast transport object maps to a partial media object or where the media object is carried with some form of additional encoding (e.g. GZip [8]), the `Repr-Digest` HTTP field may additionally be used to carry a digest of the complete unencoded media object. For example, where a single ingest object is transmitted as a series of multicast transport objects, each one with a specified byte range as defined in section 14.1 of RFC 9110 [2], the multicast transport object carrying the final byte range of the media object may include a `Repr-Digest` HTTP field to enable a *Multicast gateway* to verify the integrity of the reconstructed media object. The `Repr-Digest` HTTP field may be carried in a trailer section of the HTTP message per section 6.5 of RFC 9110 [2].

NOTE: When a media object is carried complete and without any change to its form or encoding, the digest hash values carried by a `Content-Digest` header and `Repr-Digest` header are equal, rendering the latter redundant.

12.1.2 Profile of HTTP digest fields for HTTP chunked transfer coding

Where a multicast transport object is sent using HTTP chunked transfer coding as specified in section 7.1 of RFC 9112 [1], the *Multicast server* may not have the full ingest object available to compute the digest value before the first chunk is ready for transmission at reference point **M**. In such cases, the `chunk-content-digest` HTTP/1.1 chunk extension specified in this clause may be used to protect the integrity of individual HTTP chunks. It is specified as an HTTP/1.1 chunk extension (see section 7.1.1 of RFC 9112 [1]).

The `chunk-content-digest` chunk extension is modelled on the `Content-Digest` HTTP header specified in RFC 9530 [20] and carries a message digest pertaining to the payload data conveyed in the associated HTTP chunk. The syntax of the chunk extension is specified in listing 12.1.2-1 below in terms of Augmented Backus-Naur Form (ABNF) notation as specified in RFC 5234 [40]:

Listing 12.1.2-1: Augmented Backus-Naur Form syntax of `chunk-content-digest` HTTP/1.1 chunk extension

```
chunk-content-digest    = chunk-name "=" digest
chunk-name          = "chunk-content-digest"
digest              = DQUOTE hash-key ":" digest-value ":"
DQUOTE
hash-key                = string
string              = 1*(ALPHA / DIGIT / "-")
digest-value            = base64-str
base64-str          = 1*(ALPHA / DIGIT / "+" / "/") *("=")
```

An example of a HTTP chunked transfer encoded message including the `chunk-content-digest` chunk extension is given below. The optional `Repr-Digest` field is included as a trailer field after the final chunk, enabling the recipient to check the integrity of the whole received object. The expected presence of the `Repr-Digest` trailer field is advertised by the `Trailer` header field.

Example 12.1.2-1: HTTP/1.1 chunked transfer coded message including the `chunk-content-digest` chunk extension

```
HTTP/1.1 200 OK\r\n
Content-Type: video/mp4\r\n
Transfer-Encoding: chunked\r\n
Trailer: Repr-Digest\r\n
\r\n
14117;chunk-content-digest="sha-256=:jx3J8QAr2GW1y4GZ/GLLHDAz2DBc4x59Nidm19VLl9A=:"\r\n
<14117 bytes of message body>\r\n
9874;chunk-content-digest="sha-256=:AOqEnZCUWbmo297En6sY147TJXe6JiwNHUOqyUqWWY8=:"\r\n
<9874 bytes of message body>\r\n
8974;chunk-content-digest="sha-256=:cLWUSA+XWB7JED+VkvKh3Gk+ku2qym1QrpsigR/ELfw=:"\r\n
<8974 bytes of message body>\r\n
0\r\n
Repr-Digest: sha-256=:uPssR17KN74VGWJUs95pV+RuCEq24986AuuoeJYI4mY=:\r\n
```

12.1.3 Additional security considerations for HTTP digest fields

Content digest hash algorithms considered insecure according to the criteria specified in section 5 of RFC 9530 [20] shall not be used in combination with authenticity assertion (see clause 12.2.1.4).

12.2 Authenticity assertion of multicast transport objects

12.2.0 Introduction

The assertion of authenticity is based on HTTP message signatures as specified in RFC 9421 [38]. A baseline profile of [38] is specified in clause 12.2.1 and a further profile for use with individual HTTP chunks is specified in clause 12.2.2. Additional security considerations are specified in clause 12.2.3.

12.2.1 Profile of HTTP message signatures

12.2.1.0 General

RFC 9421 [38] is profiled for use in unidirectional multicast delivery as follows:

- The creation of the signature base shall be performed as described in section 2.5 of [38].
- The creation and verification of signatures shall be performed as described in sections 3.1 and 3.2 respectively of [38].
- Because the *Multicast gateway* does not make an explicit request for a media object to the *Multicast server*, the request–response signature binding specified in section 2.4 of [38] does not apply to this profile of the message signatures specification.

Additional profiling of the base message signatures specification is specified in the following clauses.

12.2.1.1 Derived components

Section 2.2 of RFC 9421 [38] specifies how the HTTP message signature may protect other elements of metadata – called derived components – beyond the HTTP fields present in the form of headers and trailers. Unless otherwise specified by the multicast transport protocol, the following derived components shall not form a message component as part of the signature mechanism defined in this profile:

- *@method*: There is no direct mapping for an HTTP request method to a multicast transport object.
- *@status*: There is no equivalent of an HTTP status code specified for a multicast transport object.

Conversely, the following derived components specified in section 2.2 of RFC 9421 [38] may form a message component, with the specified contents:

- *@target-uri*: The content shall be the complete multicast transport object URI, including any fragment identifier component.

NOTE 1: This expands the definition of this derived component beyond that in [38].

- *@scheme*: The content shall be the scheme component of the multicast transport object URI.
- *@authority*: The content shall be the authority component of the multicast transport object URI, if present.
- *@request-target*: The content shall be the path and query components of the multicast transport object URI.
- *@path*: The content shall be the path component of the multicast transport object URI.
- *@query*: The content shall be all query components in the multicast transport object URI, if present.
- *@query-param*: The content shall be a specific named query parameter from the multicast transport object URI.

NOTE 2: Implementations should avoid use of the *@request-target* derived component to refer to all URI components after the authority because [38] does not account for the fragment component.

In addition to those permitted above, this specification also introduces the following additional derived components:

- *@fragment*: The content shall be the fragment identifier component of the multicast transport object URI as defined in section 3.5 of RFC 3986 [15]. The component value is the entire normalised fragment identifier component excluding the leading "#" character.

For example, the following multicast transport object URI would result in the specified signature base line:

Table 12.2.1-1: Fragment signature base line example

Multicast transport object URI	tag:multicastserver.isp.net,2019-01-01:mts100,3922:segment1234.m4s**#2**
Signature base line	"@fragment": **#2**

12.2.1.2 Signature components

If present, the following multicast transport object metadata shall be protected:

- Signature parameters.
- Content digest field(s) per clause 12.1.1 that use a standard hash algorithm as described in clause 12.1.1.

It is possible that a bad actor on the network path may be able to change elements of the multicast transport object metadata to exploit vulnerabilities in a *Multicast gateway* implementation. For example, a modification of the multicast transport object URI may cause a change in the unicast repair URL, leading to a unicast repair request being directed to another host, depending on how the values carried in the multicast session configuration are transformed by a given *Multicast gateway* implementation. By signing more components than the minimum required set specified above, a *Multicast gateway* is able to check the validity of those additionally protected metadata components before acting upon their contents.

The following multicast transport object metadata should additionally be protected:

- Multicast transport object URI.
- Any other protocol identifier which uniquely identifies a singular multicast transport object, such as the multicast transport object identifier.

12.2.1.3 Signature parameters

The following profiles the parameters of the `Signature` HTTP field:

- The ***alg*** signature parameter shall be included as a signature parameter to signal to the *Multicast gateway* and *Unicast repair service* functions which algorithm is used to verify the signature.
- The ***keyid*** signature parameter shall identify the public key that is used to verify the signature by means of the subject key identifier of an X.509 certificate [39], as specified in clause 12.2.1.6, encoded as a string of hexadecimal characters.

12.2.1.4 Digest hashing algorithms covered by a message signature

Section 5 of RFC 9530 [20] specifies that only secure hashing algorithms with an "active" status in the IANA registry of Hash Algorithms for HTTP Digest Fields [42] (such as ***sha-256*** and ***sha-512***) are to be used in the content digest that is protected by a message signature for the purposes of checking the authenticity of a multicast transport object. In the context of this profile, a message signature may protect an insecure content digest or checksum value in addition to a secure content digest.

12.2.1.5 Message signature algorithms

Section 3.3 of RFC 9421 [38] specifies the HTTP Message Signature Algorithms Registry. Implementations conforming to this specification shall support at least the following algorithms from that registry:

- For asymmetric signature operations, at least the "RSASSA-PKCS1-v1_5 using SHA-256" algorithm as specified in section 3.3.2 of [38] shall be supported.
- For symmetric signature operations, at least the "HMAC using SHA-256" algorithm as specified in section 3.3.3 of [38] shall be supported.

NOTE: Clause 12.2.3 contains additional discussion on the security risks associated with symmetric signature algorithms.

Configuration or negotiation of which signature algorithm is to be used for a given multicast transport object signature are out of the scope of the present document. The algorithm selected shall be signalled as specified in clause 12.2.1.3.

12.2.1.6 Certificate verification

When an asymmetric signature algorithm is in use, the public key shall take the form of an X.509 v3 certificate with a subject key identifier extension as specified in section 4.2.1.2 of RFC 5280 [39].

If present, the X.509 Common Name (CN) field or one of the Subject Alternative Name (SAN) fields in the X.509 v3 certificate as specified in section 4.2.1.6 of RFC 5280 [39] shall match either:

- The hostname carried in the **`NetworkSourceAddress`** element of the multicast transport session configuration as specified in clause 10.2.3.9 (if present), or:
- The hostname used by the *Multicast server* in the multicast transport object URI.

If the public certificate used to verify the authenticity of the multicast transport objects is transmitted via reference point **M**, then the certificate itself may be prone to tampering by a bad actor on the network path. To protect against this risk, certificates used to verify the authenticity of multicast transport objects shall themselves be verified using the certification path validation procedures described in section 6 of RFC 5280 [39] before they are used for that purpose. The provisioning and verification of any intermediate certificates or trusted root certificate to perform the certificate path validation is beyond the scope of this specification.

12.2.2 Profile of HTTP message signatures for HTTP chunked transfer coding

Where a multicast transport object is sent using HTTP chunked transfer coding as specified in section 7.1 of RFC 9112 [1], the *Multicast server* may not have the full complement of metadata fields (such as the message digest) available to compute the message signature value before the first chunk is ready for transmission at reference point **M**. In such cases, the `Chunked-Signature-Input` HTTP field and associated `chunk-signature` HTTP/1.1 chunk extension specified in this clause may be used to assert the authenticity protection of individual HTTP chunks. The latter is specified as an HTTP/1.1 chunk extension (see section 7.1.1 of RFC 9112 [1]).

The `Chunked-Signature-Input` HTTP field is modelled on the `Signature-Input` HTTP header specified in RFC 9421 [38] and carries an ordered list of components protected by the chunk signature. Each chunk signature in a given multicast transport object shall protect the same components. At a minimum, each chunk signature shall protect the `Chunked-Signature-Input` (in place of the `Signature-Input` HTTP header) as well as a chunk digest as specified in clause 12.1.2. Each chunk signature should also protect the multicast transport object URI.

In addition to the derived components specified in clause 12.2.1.1, this clause introduces the concept of *chunked components* to apply to the `Chunked-Signature-Input` HTTP field. A chunked component that begins with a hash "#" character indicates a chunk extension that shall both be present with every HTTP chunk and shall be protected by the chunk signature. The ***chunk-size*** indicator may be protected with the special ***#@chunk-size*** derived chunk component identifier.

The `chunk-signature` HTTP chunk extension is modelled on the `Signature` HTTP header specified in RFC 9421 [38] and carries a message signature protecting all the fields and chunk extensions described by the `Chunked-Signature-Input` HTTP field specified above. The syntax of the chunk extension is specified in listing 12.2.2-1 below in terms of Augmented Backus-Naur Form (ABNF) notation as specified in RFC 5234 [40]:

Listing 12.2.2-1: Augmented Backus-Naur Form syntax of `chunk-signature` HTTP/1.1 chunk extension

```
chunk-signature     = chunk-name "=" signature
chunk-name          = "chunk-signature"
signature               = DQUOTE signature-key "=:" signature-
value ":" DQUOTE
signature-key           = string
string              = 1*(ALPHA / DIGIT / "-")
signature-value     = base64-str
base64-str          = 1*(ALPHA / DIGIT / "+" / "/") *("=")
```

An example of a HTTP chunked transfer coded message including the `chunk-signature` chunk extension is given below, along with the `chunked-content-digest` chunk extension specified in clause 12.1.2, with line wrapping using a single backslash ("\") character as per RFC 8792 [i.20]. The optional `Repr-Digest`, `Signature-Input` and `Signature` fields are included in the trailer section after the final chunk, enabling the recipient to check the integrity and authenticity of the whole received object. The expected presence of these trailer fields is advertised by the `Trailer` header field.

Example 12.2.2-1: HTTP/1.1 chunked transfer coded message including the `chunk-signature` chunk extension

```
HTTP/1.1 200 OK\r\n
Content-Type: video/mp4\r\n
Transfer-Encoding: chunked\r\n
Chunked-Signature-Input: chunk-sig=("#chunk-content-digest" "@target-uri");created=1671450156\
    ;keyid="888ABA7B0ADF37D1F97492877A68D5CE527B9FD2";alg="rsa-pss-sha512"\r\n
Trailer: Repr-Digest, Signature-Input, Signature\r\n
\r\n
14117;chunk-content-digest="sha-256=:jx3J8QAr2GW1y4GZ/GLLHDAz2DBc4x59Nidm19VLl9A=:";\
    chunk-signature="chunk-sig=:U7maQ0ZXxF2gWzPU0Btzh3FiPxc9eyh+YHqcnkcl/MzXlK8QpsSzYv7+DWA1lIEe\
        bqnzVo7THL9ASUFEouEI4Y3Q9O5taTlyFlkKGRzH1WA5gf7ifzpR0m2+ch+K1DB3ngozzDYsurbDsrs+85hAZIG\
        qL7ehODMiW3hvBCD57F0ezMsBMGMLZuUyTdEDNT7cRN6+3eYM4w2gXrUHUCqAeSlafwSdHUrUETTYIH5iNCDDOC\
        f3Q5TmE4sk8VZfN+Gw3LFG3bM1hEr6mOI22amLp3ZVC7UkVNBQjfcP5JKtoAyclwi2YEHmlUQFqah19br0ycynG\
        XCnOthnZt63huxllQ==:"\r\n
<14117 bytes of message body>\r\n
9874;chunk-content-digest="sha-256=:AOqEnZCUWbmo297En6sY147TJXe6JiwNHUOqyUqWWY8=:";\
    chunk-signature="chunk-sig=:dKYyJbEuFry5sLUIwFt8qxJCDX3d/BnetDQ1q+nM/8t/LHcHg0k2SZunFmSmH0el\
        2bMBmKoIiqXaZrkCwDpSvn2WV9AKyCNqnttE1k3rpEjzr9EHWXFCWcBu\CLo7nqD2J4irvrFtjskwSsZfGs9jn2\
        WkVFULqeXwIG2E5KL9tmH1kx3tFmH+yb1LlvX9wtaoCkk9N4FXm5nNg20G0TUwEFUGRuVK70n8ge/hIhLem2nc5\
        66T5R7ZJkC4+fTlLD0fZPVRrbahS7HCfuHJvbFx9PsSkLur9C/6BW6xILfOjaXhzbIvKhWiMGOnLMShafj4dlVB\
        nUh8+KsR5nKMMBAD5Q==:"\r\n
<9874 bytes of message body>\r\n
8974;chunk-content-digest="sha-256=:cLWUSA+XWB7JED+VkvKh3Gk+ku2qym1QrpsigR/ELfw=:";\
    chunk-signature="chunk-sig=:VqRNpz/mya3VCHsZELE/whyKmK9Gkw3hlz0bXztoo80NTIRExxMdeUz/8jse1pON\
        NSufaL+r8YXdAz+XVVipbePm0i3mL3ebpzaVA9X8FY7vSSbb2JaS7BciB5rxfviMzHIl0Z75cmFYyTtnX3KFwQn\
        qOc9YJDjbgIL7JYf9wYDqy5TjjOp8+2CoUXeRwmgYfJCqAb72wdcbMO9Hp2Ys4m8oMoeVJSrUO4P6PmGKG0zQFu\
        HSsbcQKEiVhhKP7vGX6T91Wtmt3wfS+xlVFbya2DWlaU584j3hjJ9Z3KoNetouWwglxOoJpA2A0ZNENHma/8VRs\
        WHzT0a8YeDZn7w3XA==:"\r\n
<8974 bytes of message body>\r\n
0\r\n
Repr-Digest: sha-256=:uPssR17KN74VGWJUs95pV+RuCEq24986AuuoeJYI4mY=:
Signature-Input: repr-sig=("@target-uri" "repr-digest");created=1671450156\
    ;keyid=" 888ABA7B0ADF37D1F97492877A68D5CE527B9FD2";alg="rsa-pss-sha512"\r\n
Signature: repr-
sig=:Wc1A2LmgDObuwol182eM2s0wy6fgHwL3Fg1B5M5rmQASSfJDwfsnxzS5Tfn6v2XTMM72iJLv7pnV\
    N+ryGvXAaKU3oWSGOVJ7kEblomVpbg76dDkucxc7R/GHgpGcXaT7/Dj4XLR9+XHmNMvNryeh5RLlPz1Ia0llqhBQvJ+m\
    Mqu0zaOyIwc2EbhTZ3RSq591YksAVX+pMM24bMB6o5ul6CrG4An0YGbKMOlFiv6+QOxMf72ZB0DsjaNDmr2dB4i4rYsM\
    RrTge/mEL1jrelLhFwhIhqsv6mOJ6pSqgUuNvQ+3lcKBiqokKFAguJrKnpzHk4cLntn08xD9rzjTGEe/lA==:\r\n
\r\n
```

The signature input for the first `Chunk-Content-Digest` chunk extension in the above example is given below, with line wrapping using a single backslash ("\") character as per RFC 8792 [i.20]:

Example 12.2.2-2: Signature input base line

```
"#chunk-content-digest": sha-
256=:jx3J8QAr2GW1y4GZ/GLLHDAz2DBc4x59Nidm19VLl9A=:
"@target-uri": tag:multicastserver.isp.net,2022-12-
19:mts100,3922:segment1234.m4s
"@signature-params": ("#chunk-content-digest" "@target-
uri");created=1671450156;\
    keyid="888ABA7B0ADF37D1F97492877A68D5CE527B9FD2";alg="rsa-pss-
sha512"
```

12.2.3 Additional security considerations for HTTP message signatures

If the multicast transport security mode for a multicast transport session is specified as *integrityAndAuthenticity* according to clause 10.2.3.6, and a multicast transport object is received via that multicast transport session without a usable message signature the *Multicast gateway* shall consider the entire multicast transport object lost and shall instead retrieve it via reference point **A** (if it is available in the deployment).

As described in sections 7.3.2 and 7.3.3 of RFC 9421 [38], the use of symmetric signature algorithms presents a significant security risk: if an attacker can compromise the key used in the signing process, it is able to impersonate the signer. Therefore, implementations or deployments conforming to the present document that choose to use a symmetric signature algorithm to assert the authenticity of multicast transport objects should implement additional steps to ensure that the shared secret used is not used by a bad actor in the network to impersonate the *Multicast server*. The mechanism to do this is not specified in the present document.

Annex A (normative): Multicast session configuration schema

A.0 General

The XML Schemas for the multicast session configuration instance document specified in clause 10.2 are reproduced in this annex.

- The extensibility schema is specified in clause A.1.
- The baseline multicast session configuration schema is specified in clause A.2.

The version of the baseline multicast session configuration schema, indicated in the `@version` attribute of its root element, shall be the highest value listed in the first column of table A.0-1 below. For each version number, the table specifies the set of schema name spaces that a conformant *Multicast gateway* implementation shall support in order to successfully parse a multicast gateway configuration instance document indicating that version number in the `@schemaVersion` attribute of its root element as specified in clause A.1.

Table A.0-1: Schema version history

Schema version number	Required schema namespaces	Schema definition reference
1	urn:dvb:metadata:MulticastSessionConfiguration:2019	—
2	urn:dvb:metadata:Extensibility:2024	Clause A.1
	urn:dvb:metadata:MulticastSessionConfiguration:2024	Clause A.2

The schema version number shall be incremented by 1 by every future published version of the present document in which new element(s) or attribute(s) are added to the schema according to the extensibility mechanism specified in clause A.1. The new version number shall be added to table A.0-1 above and the complete set of schema namespaces that a conformant *Multicast gateway* implementation is required to support shall be listed against this version number.

A.1 Extending the multicast session configuration instance document data model

A.1.1 Extensibility schema

The schema specified in this clause is assumed to have the filename "extensibility_2024-04-24.xsd" for the purpose of import by other schemas.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Extensibility:2024"
xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:dvb:metadata:Extensibility:2024"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:attribute name="schemaVersion" type="xs:unsignedInt"/>
    <xs:element name="NamespaceDelimiter">
        <xs:complexType/>
    </xs:element>
</xs:schema>
```

All multicast session configuration instance documents shall declare their conformance with a particular version of the schema specified in clause A.2 by including the `@schemaVersion` attribute in their root element. The value of this attribute shall be one of those listed in the first column of table A.0-1.

A.1.2 Extension elements

The multicast session configuration schema permits any number of *extension elements* to be present at designated *extension points* in a multicast session configuration instance document by including the XML Schema declaration **xs:any** shown in listing A.1.2-1 at these points in the baseline XML Schema specified in clause A.2. If present, this shall appear last in any schema element or schema type declaration and the enclosing schema declaration shall be an **xs:sequence** element, as shown.

Listing A.1.2-1: Schema declaration for extension elements

```
<xs:sequence>
    <!– Other elements... -->
    <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
</xs:sequence>
```

To comply with the **xs:any** declaration in listing A.1.2-1, extension elements shall be defined in an XML Schema namespace that is different from the baseline XML Schema namespace specified in clause 10.2.1.0.

Extension elements may be used to include implementation-specific data in a multicast session configuration instance document. The effect of the **xs:any** declaration in listing A.1.2-1 is that such *private extension elements* are ignored by *Multicast gateway* implementations that are compliant only with the syntax specified in clauses 10.2 of the present document and the XML Schema specified in clause A.2.

A.1.3 Standardised extension elements

To maintain forward and backward compatibility between *Multicast gateway* and the *Multicast server* implementations based on different versions of the present document, extensions to the multicast session configuration instance document data model that are specified in the present document shall appear only at the designated extension points listed in clause A.1.5. These *standardised extension elements* shall also be defined in their own dedicated namespaces.

In addition:

1. The set of standardised extension elements specified in a given version of the present document shall be declared in the same XML Schema namespace as each other, and this namespace shall be different from that used in all other versions of the present document.

2. The schema namespace of the baseline XML schema shall not be modified when standardised extension element declarations are added to it by a revision of the present document.

3. Instead, the schema's version number (declared in its **xs:schema**`@version` attribute) shall be incremented by 1 when standardised extension elements are added to the baseline schema. A new entry shall be added to table A.0-1 listing the set of XML Schema namespaces valid with new schema version. This list shall be the set of XML Schema namespaces valid with the previous schema version plus the XML Schema namespace for the standardised extension elements specified in the revision of the present document.

4. The new set of standardised extension elements shall be terminated by a **NamespaceDelimiter** element, as specified in clause A.1.1. This element shall be declared with the attributes `minOccurs="1"` and `maxOccurs="1"`.

5. The new set of standardised extension elements (including the terminal **NamespaceDelimiter** element) shall be inserted after all existing elements in the parent **xs:sequence** element (including any previous extension elements, also terminated by a **NamespaceDelimiter** element), but immediately before the **xs:any** declaration in listing A.1.2-1 (which always appears last in the declaration of a schema element or schema type).

The net result of the above is that a single **NamespaceDelimiter** element always appears at the boundary between standardised extension elements of different XML Schema namespaces specified in different versions of the present

document, and a single **NamespaceDelimiter** element always appears at the boundary between the last standardised extension element and the **xs:any** extension point declaration.

NOTE: Inclusion of the **NamespaceDelimiter** element at these points avoids schema validation errors due to the Unique Particle Attribution constraint in XML Schema version 1.0 and achieves the aim of both backwards and forwards compatibility for instance documents.

A.1.4 Extension attributes

The presence of an `xs:anyAttribute` element in an element type declaration in the multicast session configuration schema specified in clause A.2 indicates that the element may be extended with the addition of any new attribute.

The multicast session configuration schema permits any number of *extension attributes* to be present at designated extension points in a multicast session configuration instance document by including the XML Schema declaration shown in listing A.1.4-1 at these extension points. If present, this shall appear last in any list of attributes in the schema element or schema type declaration.

Listing A.1.4-1: Schema declaration for extension attributes

```
<xs:anyAttribute processContents="skip"/>
```

A.1.5 List of standardised extension points

The full list of elements comprising the standardised extension points is given in table A.1.5-1 below. Extension elements and/or attributes shall not be present on any element specified in clause 10.2 that is not included in this table.

Table A.1.5-1: Standardised extension points

Element name	Clause	Permits extension elements	Permits extension attributes
EndpointAddress	10.2.3.9	✓	✓
ForwardErrorCorrectionParameters	10.2.3.11	✓	✓
InitSegments	10.2.3.14.2	×	✓
MulticastGatewayConfiguration	10.2.1.0	✓	✓
MulticastGatewayConfigurationTransportSession	10.2.5	✓	✓
MulticastGatewaySessionReporting	10.2.1.0	×	✓
MulticastServerConfiguration	10.2.1.0	✓	✓
MulticastSession	10.2.2	✓	✓
MulticastTransportSession	10.2.3	✓	✓
ObjectCarousel	10.2.3.14	✓	✓
PresentationManifests	10.2.3.14.1	×	✓
ReportingLocator	10.2.1.0	×	✓
ResourceLocator	10.2.3.14.3	×	✓
ServiceComponentIdentifier	10.2.4	×	✓
TransportProtocol	10.2.3.8	×	✓
UnicastRepairParameters	10.2.3.12	✓	✓

A future version of the present document defining new schema types using the mechanism specified in clause A.1.3 may designate these new schema types as extension points by using the syntax specified in clause A.1.2. In this case, the new schema type shall be added to the table A.1.5-1 above.

A future version of the present document shall not retrospectively designate an existing schema type not listed in the above table as an extension point. Hence, no schema type defined in an earlier version of the present document that is not already present in the table A.1.5-1 shall be added to the table.

NOTE: Retrospectively designating a schema type as an extension point prevents parsers based on older versions of the schema from processing instance documents that comply with the new schema version when they contain extensions at positions not so designated by the older schema version.

A.1.6 Examples of schema extension elements (informative)

Schema version 1, shown in listing A.1.6-1 declares a root element **`Root`** of schema type **`RootType`** which has a designated element extension point after the (only) standardised child element **`First`**.

Listing A.1.6-1: Example schema version 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:ExtensionElements" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" targetNamespace="urn:dvb:metadata:Example:ExtensionElements"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1">
    <xs:complexType name="RootType">
        <xs:sequence>
            <xs:element name="First" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:attribute ref="ext:schemaVersion" use="required"/>
    </xs:complexType>

    <xs:element name="Root" type="RootType"/>
</xs:schema>
```

Private extension elements are permitted to appear after the element **`First`**, as shown in the example instance document in listing A.1.6-2. No **`NamespaceDelimiter`** element is required by the schema to appear before the designated extension point in schema version 1, but including this element allows the instance document to be forwards-compatible with future versions of the schema.

Listing A.1.6-2: Example instance document compliant with schema version 1

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns="urn:dvb:metadata:Example:ExtensionElements" xmlns:ext="urn:dvb:metadata:Extensibility:2024"
xmlns:private="tag:example.com,2024:metadata:cs:private" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ext:schemaVersion="1">
    <First>Some text</First>
    <First>Some other text</First>
    <ext:NamespaceDelimiter/>   <!– Required for forwards compatibility with later schema versions. -->
    <private:Thing>...</private:Thing>
</Root>
```

Schema version 2, shown in listing A.1.6-3 declares two standardised extension elements **`Second`** and **`Third`**, defined in the extension schema shown in listing A.1.6-4, terminated by a **`NamespaceDelimiter`** element. As required, the two standardised extension elements are declared in the same namespace as each other. Any private extension elements are now required to appear in an instance document after these standardised extension elements, delimited from them by the **`NamespaceDelimiter`** element.

Listing A.1.6-3: Example schema version 2

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:ExtensionElements" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" xmlns:ext1="urn:dvb:metadata:Example:Extensions:1"
targetNamespace="urn:dvb:metadata:Example:ExtensionElements" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="2">
    <xs:complexType name="RootType">
        <xs:sequence>
            <xs:element name="First" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext1:Second" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext1:Third" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext:NamespaceDelimiter" minOccurs="1" maxOccurs="1"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:attribute ref="ext:schemaVersion" use="required"/>
    </xs:complexType>

    <xs:element name="Root" type="RootType"/>
</xs:schema>
```

Listing A.1.6-4: Example extension schema 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:Extensions:1" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:dvb:metadata:Example:Extensions:1" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Second" type="xs:string"/>
    <xs:element name="Third" type="xs:string"/>
</xs:schema>
```

An example instance document compliant with schema version 2 is shown in listing A.1.6-5.

- A parser that understands only schema version 1 treats the elements **`Second`**, **`Third`**, **`PrivateDelimiter`** and **`Thing`** as private extension elements. Hence, a schema version 1 parser is forwards-compatible with instance documents declaring a later version in the `@schemaVersion` attribute.

- A parser that understands schema version 2 is able to process the elements **`Second`** and **`Third`**, and treats elements after the **`NamespaceDelimiter`** element as private extension elements.

Listing A.1.6-5: Example instance document compliant with schema version 2

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns="urn:dvb:metadata:Example:ExtensionElements" xmlns:ext="urn:dvb:metadata:Extensibility:2024"
xmlns:ext1="urn:dvb:metadata:ElemExtExample:Extensions:1"
xmlns:private="tag:example.com,2024:metadata:cs:private" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ext:schemaVersion="2">
    <First>Some text</First>
    <First>Some other text</First>
    <ext1:Second>...</ext1:Second>
    <ext1:Second>...</ext1:Second>
    <ext1:Third>...</ext1:Third>
    <ext:NamespaceDelimiter/>
    <private:Thing>...</private:Thing>
</Root>
```

Schema version 3, shown in listing A.1.6-6 declares a standardised extension element **`Fourth`**, defined in the extension schema shown in listing A.1.6-7, followed by an additional **`NamespaceDelimiter`** element. This new extension element appears after the standardised extension elements declared in schema version 2. Any private extension elements are now required to appear after these standardised extension elements, delimited from them by a **`NamespaceDelimiter`** element.

Listing A.1.6-6: Example schema version 3

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:ExtensionElements" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" xmlns:ext1="urn:dvb:metadata:Example:Extensions:1"
xmlns:ext2="urn:dvb:metadata:Example:Extensions:2" targetNamespace="urn:dvb:metadata:Example:ExtensionElements"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="3">
    <xs:complexType name="RootType">
        <xs:sequence>
            <xs:element name="First" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext1:Second" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext1:Third" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext:NamespaceDelimiter" minOccurs="1" maxOccurs="1"/>
            <xs:element ref="ext2:Fourth" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="ext:NamespaceDelimiter" minOccurs="1" maxOccurs="1"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:attribute ref="ext:schemaVersion" use="required"/>
    </xs:complexType>

    <xs:element name="Root" type="RootType"/>
</xs:schema>
```

Listing A.1.6-7: Example extension schema 2

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:Extensions:2" xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:dvb:metadata:Example:Extensions:2" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="Fourth" type="xs:string"/>
</xs:schema>
```

An example instance document compliant with schema version 3 is shown in listing A.1.6-8.

Listing A.1.6-8: Example instance document compliant with schema version 3

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns="urn:dvb:metadata:ElemExtExample" xmlns:ext="urn:dvb:metadata:Extensibility:2024"
xmlns:ext1="urn:dvb:metadata:ElemExtExample:Extensions:1"
xmlns:ext2="urn:dvb:metadata:ElemExtExample:Extensions:2"
xmlns:private="tag:example.com,2024:metadata:cs:private" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ext:schemaVersion="3">
    <First>Some text</First>
    <First>Some other text</First>
    <ext1:Second>...</ext1:Second>
    <ext1:Second>...</ext1:Second>
    <ext1:Third>...</ext1:Third>
    <ext:NamespaceDelimiter/>    <!– Required for backwards compatibility with older schema versions. -->
    <ext2:Fourth>...</ext2:Fourth>
    <ext:NamespaceDelimiter/>
    <private:Thing>...</private:Thing>
</Root>
```

This instance document is treated as follows:

- A parser that understands only schema version 1 treats the elements **Second**, **Third**, **Fourth**, **NamespaceDelimiter** and **Thing** as private extension elements.
- A parser that understands only schema version 2 treats the elements **Fourth**, **NamespaceDelimiter** and **Thing** as private extension elements.
- A parser that understands schema version 3 is able to process the **Fourth** element and treats elements after the **NamespaceDelimiter** element as private extension elements.

A.1.7 Examples of schema extension attributes (informative)

Schema version 1, shown in listing A.1.7-1 declares a root element **Root** of schema type **RootType** which has two designated attribute extension points: one on the root element itself and the other on the (only) standardised child element **Node**.

Listing A.1.7-1: Example schema version 1

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:AttributeExtension" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" targetNamespace="urn:dvb:metadata:Example:AttributeExtension"
elementFormDefault="qualified" attributeFormDefault="unqualified" version="1">
    <xs:complexType name="RootType">
        <xs:sequence>
            <xs:element name="Node" minOccurs="0" maxOccurs="unbounded">
                <xs:extension base="xs:string">
                    <xs:anyAttribute processContents="skip"/>
                </xs:extension>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="schemaVersion" type="xs:unsignedInt" use="required"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:element name="Root" type="RootType"/>
</xs:schema>
```

Thus, private extension attributes are permitted to appear alongside the `@schemaVersion` attribute and on the **Node** element, as shown in the example instance document in listing A.1.7-2.

Listing A.1.7-2: Example instance document compliant with schema version 1

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns="urn:dvb:metadata:Example:AttributeExtesions" xmlns:ext="urn:dvb:metadata:Extensibility:2024"
xmlns:private="tag:example.net,2024:metadata:cs:private" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ext:schemaVersion="1" private:tag="65796843">
    <Node>Some text</Node>
    <Node private:tag="68436579">Some other text</Node>
</Root>
```

Schema version 2, shown in listing A.1.7-3 declares two standardised extension attributes, `@eTag` in the **`Root`** element and `@foo` in the **`Node`** element.

Listing A.1.7-3: Example schema version 2

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:Example:AttributeExtension" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" xmlns:ext1="urn:dvb:metadata:Example:Extensions:1"
targetNamespace="urn:dvb:metadata:Example:AttributeExtension" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="2">
    <xs:complexType name="RootType">
        <xs:sequence>
            <xs:element name="Node" minOccurs="0" maxOccurs="unbounded">
                <xs:extension base="xs:string">
                    <xs:attribute name="foo" type="ext1:bar" use="required"/>
                    <xs:anyAttribute processContents="skip"/>
                </xs:extension>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="schemaVersion" type="xs:unsignedInt" use="required"/>
        <xs:attribute name="eTag" type="xs:string" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:element name="Root" type="RootType"/>
</xs:schema>
```

An example instance document compliant with schema version 2 is shown in listing A.1.7-4.

Listing A.1.7-4: Example instance document compliant with schema version 1

```
<?xml version="1.0" encoding="UTF-8"?>
<Root xmlns="urn:dvb:metadata:Example:AttributeExtension" xmlns:ext="urn:dvb:metadata:Extensibility:2024"
xmlns:private="tag:example.net,2024:metadata:cs:private" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" ext:schemaVersion="2" eTag="123-a">
    <Node foo="baz">Some text</Node>
    <Node foo="baz" private:priv="beans">Some other text</Node>
</Root>
```

This instance document is treated as follows:

- A parser that understands only schema version 1 treats the **`Root`**`@eTag`, **`Node`**`@foo` and **`Node`**`@priv` attributes as private extension attributes and skips validating them.
- A parser that understands schema version 2 is able to process the **`Root`**`@eTag` and **`Node`**`@foo` attributes, and treats the **`Node`**`@priv` attribute as a private extension attribute and skips validating it.

A.2 Baseline multicast session configuration schema

The schema defined in this clause is assumed to have the filename "multicast-session-configuration_phase-2_2024-05-28.xsd" for the purpose of import by other schemas.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:MulticastSessionConfiguration:2024" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:ext="urn:dvb:metadata:Extensibility:2024"
targetNamespace="urn:dvb:metadata:MulticastSessionConfiguration:2024" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="2">

   <xs:import namespace="urn:mpeg:mpeg7:schema:2001" schemaLocation="mpeg7_subset.xsd"/>
   <xs:import namespace="urn:dvb:metadata:Extensibility:2024" schemaLocation="extensibility_2024-04-24.xsd"/>
   <!-- -->
   <xs:simpleType name="decimalFraction">
      <xs:annotation>
         <xs:documentation>A fraction expressed as a decimal between 0.0 and 1.0</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:decimal">
         <xs:minExclusive value="0.0"/>
         <xs:maxInclusive value="1.0"/>
      </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="stringNoWhitespaceType">
      <xs:restriction base="xs:string">
         <xs:pattern value="[^\r\n\t \p{Z}]*"/>
      </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="IPAddressType">
      <xs:annotation>
         <xs:documentation>TODO: Restrict this with a regular expression that matches IPv4 and IPv6 addresses in
their respective textual notations.</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string"/>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="PortNumberType">
      <xs:restriction base="xs:positiveInteger">
         <xs:maxInclusive value="65535"/>
      </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="contentAcquisitionMethodType">
      <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="push"/>
         <xs:enumeration value="pull"/>
      </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="transmissionModeType">
      <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="resource"/>
         <xs:enumeration value="chunked"/>
      </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="transportSecurityType">
      <xs:restriction base="xs:NMTOKEN">
         <xs:enumeration value="none">
            <xs:annotation>
               <xs:documentation/>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="integrity">
            <xs:annotation>
               <xs:documentation>A digest of the multicast transport object is present in the metadata describing
it, enabling its integrity to be verified by the Multicast gateway.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
         <xs:enumeration value="integrityAndAuthenticity">
```

```
            <xs:annotation>
               <xs:documentation>A digest of the multicast transport object is present in the metadata describing
it, enabling its integrity to be verified by the Multicast gateway. A digital signature is also provided, enabling the
authenticity of (key fields within) the multicast transport object metadata to be verified by the Multicast
gateway.</xs:documentation>
            </xs:annotation>
         </xs:enumeration>
      </xs:restriction>
   </xs:simpleType>
   <!-- -->
   <xs:simpleType name="tagAttrType">
      <xs:list itemType="xs:anyURI"/>
   </xs:simpleType>
   <!-- -->
   <xs:attributeGroup name="validityAttrs">
      <xs:attribute name="validityPeriod" type="xs:duration">
         <xs:annotation>
            <xs:documentation>The period of time after receiving this document that it may no longer be
valid.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="validUntil" type="mpeg7:timePointType">
         <xs:annotation>
            <xs:documentation>The absolute point in time after which this document may no longer be
valid.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:anyAttribute processContents="skip"/>
   </xs:attributeGroup>
   <!-- -->
   <xs:attributeGroup name="transportSessionRefAttrs">
      <xs:attribute name="serviceIdRef" type="xs:anyURI" use="optional">
         <xs:annotation>
            <xs:documentation>A reference to a service identifier in this multicast session
configuration.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
      <xs:attribute name="transportSessionIdRef" type="xs:NMTOKEN" use="optional">
         <xs:annotation>
            <xs:documentation>A reference to a transport session identifier within the scope of the indicated
multicast session.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:attributeGroup>
   <!-- -->
   <xs:attributeGroup name="targetAcquisitionLatencyAttrs">
      <xs:attribute name="targetAcquisitionLatency" type="xs:duration" use="optional">
         <xs:annotation>
            <xs:documentation>The maximum period of time between transmissions of a multicast transport object as
part of a carousel.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:attributeGroup>
   <!-- -->
   <xs:attributeGroup name="revalidationPeriodAttrs">
      <xs:attribute name="revalidationPeriod" type="xs:duration" use="optional">
         <xs:annotation>
            <xs:documentation>The maximum period of time between revalidation of a resource with its origin
server.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:attributeGroup>
   <!-- -->
   <xs:attributeGroup name="compressionAttrs">
      <xs:attribute name="compressionPreferred" type="xs:boolean" use="optional" default="false">
         <xs:annotation>
            <xs:documentation>Expression of preference to the Multicast server that compression is desired by the
Provisioning function.</xs:documentation>
         </xs:annotation>
      </xs:attribute>
   </xs:attributeGroup>
   <!-- -->
   <xs:attributeGroup name="carouselSizeAttrs">
      <xs:attribute name="aggregateTransportSize" type="xs:positiveInteger" use="optional">
```

```
            <xs:annotation>
               <xs:documentation>The combined size of all transport objects described by this carousel, excluding any
associated metadata and transport protocol overhead.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:attribute name="aggregateContentSize" type="xs:positiveInteger" use="optional">
            <xs:annotation>
               <xs:documentation>The combined size of all transport objects described by this carousel once any content
encoding (such as compression) has been removed from the objects, excluding any associated metadata and transport
protocol overhead.</xs:documentation>
            </xs:annotation>
         </xs:attribute>

      </xs:attributeGroup>
      <!-- -->
      <xs:complexType name="ServiceComponentIdentifierType" abstract="true">
         <xs:attribute name="manifestIdRef" type="xs:NMTOKEN" use="required">
            <xs:annotation>
               <xs:documentation>A cross-reference to a PresentationManifestLocator element in the parent
MulticastSession with a manifestId of the same value.</xs:documentation>
            </xs:annotation>
         </xs:attribute>
         <xs:anyAttribute processContents="skip"/>
      </xs:complexType>
      <!-- -->
      <xs:complexType name="DASHComponentIdentifierType">
         <xs:complexContent>
            <xs:extension base="ServiceComponentIdentifierType">
               <xs:attribute name="periodIdentifier" type="xs:string" use="required"/>
               <xs:attribute name="adaptationSetIdentifier" type="xs:unsignedInt" use="required"/>
               <xs:attribute name="representationIdentifier" type="stringNoWhitespaceType" use="required"/>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
      <!-- -->
      <xs:complexType name="HLSComponentIdentifierType">
         <xs:complexContent>
            <xs:extension base="ServiceComponentIdentifierType">
               <xs:attribute name="mediaPlaylistLocator" type="xs:anyURI" use="required"/>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
      <!-- -->
      <xs:complexType name="GenericComponentIdentifierType">
         <xs:complexContent>
            <xs:extension base="ServiceComponentIdentifierType">
               <xs:attribute name="componentIdentifier" type="xs:NMTOKEN" use="required"/>
            </xs:extension>
         </xs:complexContent>
      </xs:complexType>
      <!-- -->
      <xs:complexType name="MulticastTransportProtocolType">
         <xs:attribute name="protocolIdentifier" type="mpeg7:termReferenceType" use="required"/>
         <xs:attribute name="protocolVersion" type="xs:positiveInteger" use="required"/>
         <xs:anyAttribute processContents="skip"/>
      </xs:complexType>
      <!-- -->
      <xs:complexType name="MulticastEndpointAddressType">
         <xs:sequence>
            <xs:element name="NetworkSourceAddress" type="IPAddressType" minOccurs="0"/>
            <xs:element name="NetworkDestinationGroupAddress" type="IPAddressType"/>
            <xs:element name="TransportDestinationPort" type="PortNumberType"/>
            <xs:element name="MediaTransportSessionIdentifier" type="xs:positiveInteger" minOccurs="0">
               <xs:annotation>
                  <xs:documentation>Uniquely identifying the stream of packets in the multicast group that corresponds
to this multicast transport session, e.g. the LCT Channel identifier. (Some multicast media transport protocols do not
require this additional demultiplexing identifier.)</xs:documentation>
               </xs:annotation>
            </xs:element>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
               <xs:annotation>
                  <xs:documentation>Designated extension point.</xs:documentation>
               </xs:annotation>
            </xs:any>
```

```
        </xs:sequence>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="BitRateType">
        <xs:attribute name="average" type="xs:positiveInteger" use="optional"/>
        <xs:attribute name="maximum" type="xs:positiveInteger" use="required"/>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="ForwardErrorCorrectionParametersType">
        <xs:annotation>
            <xs:documentation>A set of parameters describing an Application Level Forward Error Correction
configuration.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="SchemeIdentifier" type="mpeg7:termReferenceType">
                <xs:annotation>
                    <xs:documentation>A term identifier from ForwardErrorCorrectionSchemeCS identifying the AL-FEC
scheme in use.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="OverheadPercentage" type="xs:positiveInteger">
                <xs:annotation>
                    <xs:documentation>The percentage AL-FEC overhead for the repair stream described by this set of
Forward Error Correction parameters.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="EndpointAddress" type="MulticastEndpointAddressType" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>The endpoint address to which AL-FEC repair packets are transmitted. May be
omitted in cases where AL-FEC is transmitted in band to the same endpoint address as the enclosing multicast transport
session.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="WeightedURIType">
        <xs:annotation>
            <xs:documentation>A URI with an associated weighting attribute.</xs:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="relativeWeight" type="xs:nonNegativeInteger" default="1"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- -->
```

```
    <xs:complexType name="UnicastRepairParametersType">
        <xs:annotation>
            <xs:documentation>An element describing the parameters to be used by a Multicast gateway when performing
unicast repair. One or more base paths may be specified here, each with an optional weighting. If the weighting is
omitted, all base paths are assumed to have an equal weighting of 1.0.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="BaseURL" type="WeightedURIType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:attribute name="transportObjectReceptionTimeout" type="xs:unsignedInt" use="required">
            <xs:annotation>
                <xs:documentation>The time (expressed in milliseconds) that a Multicast gateway should wait for a packet
relating to a particular multicast transport object before it can assume that the object transmission is over, and
commence object repair using Forward Error Correction or unicast patching procedures.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="fixedBackOffPeriod" type="xs:unsignedInt" default="0">
            <xs:annotation>
                <xs:documentation>The minimum number of milliseconds that the Multicast gateway shall back off after the
mutlicast transport object timeout before attempting a unicast repair.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="randomBackOffPeriod" type="xs:unsignedInt" default="0">
            <xs:annotation>
                <xs:documentation>An additional period that the Multicast gateway shall wait after the fixed back-off
period before attempting a unicast repair. It shall be a randomly selected number of milliseconds between zero and the
value specified in this attribute. The Multicast gateway shall choose a different random back-off period for each
multicast transport object.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="transportObjectBaseURI" type="xs:anyURI" use="optional">
            <xs:annotation>
                <xs:documentation>The base path of all multicast transport objects conveyed in this multicast transport
session. This prefix string is substituated by the Multicast gateway with a unicast repair base path when performing
unicast repair.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="TypedLocatorType">
        <xs:annotation>
            <xs:documentation>A URL and accompanying MIME content type.</xs:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="contentType" type="mpeg7:mimeType" use="required">
                    <xs:annotation>
                        <xs:documentation>The MIME content type of the resource pointed to by the content of this
element.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- -->
```

```
    <xs:complexType name="ReportingLocatorType">
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attribute name="proportion" type="decimalFraction" default="1.0">
                    <xs:annotation>
                        <xs:documentation>The proportion of Multicast gateways that should send reports to the specified
endpoint. At the start of a multicast transport session, each Multicast gateway shall randomly decide whether or not
to send reports based on this value.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="period" type="xs:duration" use="required">
                    <xs:annotation>
                        <xs:documentation>The period of time for the Multicast gateway to wait between sending reports to
the specified endpoint.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="randomDelay" type="xs:unsignedInt" use="required">
                    <xs:annotation>
                        <xs:documentation>An additional period that the Multicast gateway shall delay when sending
reports to the specified endpoint. It shall be a randomly selected number of milliseconds between zero and the value
specified in this attribute. The Multicast gateway shall choose a different random delay for each report it
sends.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="reportSessionRunningEvents" type="xs:boolean" use="optional" default="false">
                    <xs:annotation>
                        <xs:documentation>Controls whether Playback session running events are included in
reports.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:anyAttribute processContents="skip"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="SessionReportingType">
        <xs:sequence>
            <xs:annotation>
                <xs:documentation>Parameters used by the Multicast gateway to report statistics about the
session.</xs:documentation>
            </xs:annotation>
            <xs:element name="ReportingLocator" type="ReportingLocatorType" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="CarouselMediaPresentationResourceType">
        <xs:annotation>
            <xs:documentation>The parameters for carouselling a resource related to a media
presentation.</xs:documentation>
        </xs:annotation>
        <xs:attributeGroup ref="targetAcquisitionLatencyAttrs"/>
        <xs:attributeGroup ref="compressionAttrs"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="ReferencingCarouselMediaPresentationResourceType">
        <xs:annotation>
            <xs:documentation>The parameters for carouselling a resource related to a media presentation with reference
to an multicast transport session described in a different document.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="CarouselMediaPresentationResourceType">
                <xs:attributeGroup ref="transportSessionRefAttrs"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="CarouselResourceLocatorType">
        <xs:annotation>
            <xs:documentation>A URL and the parameters for carouselling it.</xs:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
```

```
            <xs:attributeGroup ref="targetAcquisitionLatencyAttrs">
               <xs:annotation>
                  <xs:documentation>The target acquisition period from the carousel for this
resource.</xs:documentation>
               </xs:annotation>
            </xs:attributeGroup>
            <xs:attributeGroup ref="revalidationPeriodAttrs">
               <xs:annotation>
                  <xs:documentation>The revalidation period with the origin server for this
resource.</xs:documentation>
               </xs:annotation>
            </xs:attributeGroup>
            <xs:attributeGroup ref="compressionAttrs">
               <xs:annotation>
                  <xs:documentation>Compression of this resource by the Multicast server is
desired.</xs:documentation>
               </xs:annotation>
            </xs:attributeGroup>
            <xs:anyAttribute processContents="skip"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <!-- -->
   <xs:complexType name="ObjectCarouselType">
      <xs:sequence>
         <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="PresentationManifests" type="CarouselMediaPresentationResourceType"/>
            <xs:element name="InitSegments" type="CarouselMediaPresentationResourceType"/>
            <xs:element name="ResourceLocator" type="CarouselResourceLocatorType"/>
         </xs:choice>
         <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Designated extension point.</xs:documentation>
            </xs:annotation>
         </xs:any>
      </xs:sequence>
      <xs:attributeGroup ref="carouselSizeAttrs"/>
      <xs:anyAttribute processContents="skip"/>
   </xs:complexType>
   <!-- -->
   <xs:complexType name="ReferencingObjectCarouselType">
      <xs:sequence>
         <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="PresentationManifests" type="ReferencingCarouselMediaPresentationResourceType"/>
            <xs:element name="InitSegments" type="ReferencingCarouselMediaPresentationResourceType"/>
            <xs:element name="ResourceLocator" type="CarouselResourceLocatorType"/>
         </xs:choice>
         <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
               <xs:documentation>Designated extension point.</xs:documentation>
            </xs:annotation>
         </xs:any>
      </xs:sequence>
      <xs:attributeGroup ref="carouselSizeAttrs"/>
      <xs:anyAttribute processContents="skip"/>
   </xs:complexType>
   <!-- -->
   <xs:complexType name="MacroDefinitionType">
      <xs:annotation>
         <xs:documentation>Definition of a macro expansion.</xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="key" type="xs:NMTOKEN" use="required">
               <xs:annotation>
                  <xs:documentation>The macro key to be expanded.</xs:documentation>
               </xs:annotation>
            </xs:attribute>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
   <!-- -->
```

```
    <xs:complexType name="BaseMulticastTransportSessionType">
        <xs:sequence>
            <xs:element name="TransportProtocol" type="MulticastTransportProtocolType"/>
            <xs:element name="EndpointAddress" type="MulticastEndpointAddressType" maxOccurs="unbounded"/>
            <xs:element name="BitRate" type="BitRateType"/>
            <xs:element name="ForwardErrorCorrectionParameters" type="ForwardErrorCorrectionParametersType"
minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="UnicastRepairParameters" type="UnicastRepairParametersType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="serviceClass" type="xs:anyURI" use="optional">
            <xs:annotation>
                <xs:documentation>A controlled term URI indicating the class of information conveyed by a multicast
transport session.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attribute name="transportSecurity" type="transportSecurityType" default="none">
            <xs:annotation>
                <xs:documentation>Controls whether the Multicast server adds integrity and/or authenticity metadata to
multicast transport objects. Informs the Multicast gateway whether this metadata should be present and
verified.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="MulticastGatewayConfigurationTransportSessionType">
        <xs:complexContent>
            <xs:extension base="BaseMulticastTransportSessionType">
                <xs:sequence>
                    <xs:element name="ObjectCarousel" type="ReferencingObjectCarouselType" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Specifying a carousel of ancillary multicast transport objects to be
additionally conveyed by this multicast gateway configuration transport session.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="MulticastGatewayConfigurationMacro" type="MacroDefinitionType" minOccurs="0"
maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Defining a macro to be expanded by the Multicast server when generating a
multicast gateway configuration for inclusion in this multicast gateway configuration transport
session.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Designated extension point.</xs:documentation>
                        </xs:annotation>
                    </xs:any>
                </xs:sequence>
                <xs:attribute name="tags" type="tagAttrType" use="optional">
                    <xs:annotation>
                        <xs:documentation>A list of globally unique identifiers that refers to either this multicast
gateway configuration transport session or the content carried by it.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:anyAttribute processContents="skip"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="MulticastTransportSessionType">
        <xs:complexContent>
            <xs:extension base="BaseMulticastTransportSessionType">
                <xs:sequence>
                    <xs:element name="ServiceComponentIdentifier" type="ServiceComponentIdentifierType"
maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>A means of referencing a single component of the linear
service.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                    <xs:element name="ObjectCarousel" type="ObjectCarouselType" minOccurs="0">
                        <xs:annotation>
                            <xs:documentation>Specifying a carousel of ancillary multicast transport objects to be
additionally conveyed by this multicast transport session.</xs:documentation>
```

```
                    </xs:annotation>
                </xs:element>
                <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Designated extension point.</xs:documentation>
                    </xs:annotation>
                </xs:any>
            </xs:sequence>
            <xs:attribute name="id" type="xs:NMTOKEN" use="required"/>
            <xs:attribute name="start" type="mpeg7:timePointType" use="optional"/>
            <xs:attribute name="duration" type="xs:duration" use="optional"/>
            <xs:attribute name="contentIngestMethod" type="contentAcquisitionMethodType" default="pull">
                <xs:annotation>
                    <xs:documentation>Used by the Multicast server to determine whether to pull content for this
service component, or expect it to be pushed.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="transmissionMode" type="transmissionModeType" default="resource">
                <xs:annotation>
                    <xs:documentation>Used by the Multicast server to determine the multicast transmission mode. In
"resource" mode, the Multicast server waits until it has acquired a complete resource before attempting to transmit it
as a multicast transport object. In "chunked" mode, the Multicast server maps a single acquired chunk to a different
multicast transport object.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:attribute name="sessionIdleTimeout" type="xs:positiveInteger" use="required">
                <xs:annotation>
                    <xs:documentation>The time (expressed in milliseconds) that a Multicast gateway should wait for
any packet on this multicast transport session before it can assume that the session is over and unsubscribe from the
corresponding multicast group. If this attribute is omitted, reception of the multicast session never times out due to
a lack of packets, but is still bounded by the transport session duration.</xs:documentation>
                </xs:annotation>
            </xs:attribute>
            <xs:anyAttribute processContents="skip"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:complexType name="MulticastSessionType">
    <xs:annotation>
        <xs:documentation>All the multicast transport sessions required to deliver a single linear service
according to operational needs.</xs:documentation>
    </xs:annotation>
    <xs:sequence>
        <xs:element name="PresentationManifestLocator" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>The URL of a presentation manifest hosted on an origin server accessible to the
system receiving this multicast session configuration.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="TypedLocatorType">
                        <xs:attribute name="manifestId" type="xs:NMTOKEN" use="required">
                            <xs:annotation>
                                <xs:documentation>An opaque identifier, unique within the lexical scope of this
instance document, that identifies this XML element and allows it to be cross-referenced from elsewhere in the same
instance document.</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="transportObjectURI" type="xs:string" use="optional">
                            <xs:annotation>
                                <xs:documentation>The multicast transport object URI of this presentation manifest
when it is carouselled by the Multicast server.</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
                        <xs:attribute name="contentPlaybackPathPattern" type="xs:string">
                            <xs:annotation>
                                <xs:documentation>A pattern string used by the Multicast gateway to match the path
part of presentation manifest request URLs from a Content playback function with this multicast session. May contain
any number of wildcard characters at any position.</xs:documentation>
                            </xs:annotation>
                        </xs:attribute>
```

```
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
        <xs:element name="MulticastGatewaySessionReporting" type="SessionReportingType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>The reporting destination(s) used by the Multicast gateway for all Multicast
transport sessions within the scope of this Multicast session.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="MulticastTransportSession" type="MulticastTransportSessionType" maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>Designated extension point.</xs:documentation>
            </xs:annotation>
        </xs:any>
    </xs:sequence>
    <xs:attribute name="serviceIdentifier" type="xs:anyURI" use="required">
        <xs:annotation>
            <xs:documentation>A URI that uniquely identifies a multicast session within the deployed
system.</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:attribute name="contentPlaybackAvailabilityOffset" type="xs:duration" default="PT0S">
        <xs:annotation>
            <xs:documentation>The period for which the availability start time of media objects delivered at
reference point L should be delayed by the Multicast gateway</xs:documentation>
        </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute processContents="skip"/>
</xs:complexType>
<!-- -->
<xs:complexType name="MulticastServerConfigurationType">
    <xs:sequence>
        <xs:element name="MulticastGatewayConfigurationTransportSession"
type="MulticastGatewayConfigurationTransportSessionType" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>The Multicast gateway configuration can optionally be transmitted via an in-band
multicast transport session.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="MulticastSession" type="MulticastSessionType" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>A set of multicast sessions, each one for a different linear
service.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="MulticastServerConfigurationMacro" type="MacroDefinitionType" minOccurs="0"
maxOccurs="unbounded">
            <xs:annotation>
                <xs:documentation>Defining a macro to be expanded by the Multicast server when processing the
multicast server configuration.</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="MulticastGatewaySessionReporting" type="SessionReportingType" minOccurs="0">
            <xs:annotation>
                <xs:documentation>The reporting destination(s) used by the Multicast gateway for all Multicast
transport sessions declared within the scope of this multicast server configuration.</xs:documentation>
            </xs:annotation>
        </xs:element>
```

```
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:attribute ref="ext:schemaVersion" use="required">
            <xs:annotation>
                <xs:documentation>The version attribute is intended to be increased by 1 in every future version where
new element(s) or attribute(s) are added.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attributeGroup ref="validityAttrs">
            <xs:annotation>
                <xs:documentation>Attributes declaring the validity of this Multicast server configuration document and
its contents.</xs:documentation>
            </xs:annotation>
        </xs:attributeGroup>
    </xs:complexType>
    <!-- -->
    <xs:complexType name="MulticastGatewayConfigurationType">
        <xs:sequence>
            <xs:element name="MulticastGatewayConfigurationTransportSession"
type="MulticastGatewayConfigurationTransportSessionType" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>The Multicast gateway configuration can optionally be transmitted via an in-band
multicast transport session.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="MulticastSession" type="MulticastSessionType" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>A set of multicast sessions, each one for a different linear
service.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="MulticastGatewaySessionReporting" type="SessionReportingType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>The reporting destination(s) used by the Multicast gateway for all Multicast
transport sessions declared within the scope of this multicast gateway configuration.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Designated extension point.</xs:documentation>
                </xs:annotation>
            </xs:any>
        </xs:sequence>
        <xs:attribute ref="ext:schemaVersion" use="required">
            <xs:annotation>
                <xs:documentation>The version attribute is intended to be increased by 1 in every future version where
new element(s) or attribute(s) are added.</xs:documentation>
            </xs:annotation>
        </xs:attribute>
        <xs:attributeGroup ref="validityAttrs">
            <xs:annotation>
                <xs:documentation>Attributes declaring the validity of this Multicast gateway configuration document and
its contents.</xs:documentation>
            </xs:annotation>
        </xs:attributeGroup>
    </xs:complexType>
    <!-- -->
    <xs:element name="MulticastServerConfiguration" type="MulticastServerConfigurationType">
        <xs:annotation>
            <xs:documentation>A document describing the currently configured Multicast sessions of a Multicast
server.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- -->
    <xs:element name="MulticastGatewayConfiguration" type="MulticastGatewayConfigurationType">
        <xs:annotation>
            <xs:documentation>A document describing the currently configured Multicast sessions of a Multicast
gateway.</xs:documentation>
        </xs:annotation>
    </xs:element>
```

```
</xs:schema>
```

Annex B (normative): Classification schemes

B.1 MulticastTransportProtocolCS

```
<?xml version="1.0" encoding="UTF-8"?>
<ClassificationScheme uri="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019">
   <Term termID="FLUTE">
      <Name xml:lang="en">File Delivery over Unidirectional Transport</Name>
      <Definition xml:lang="en">Version 1 (IETF RFC 3926), as profiled by ETSI TS 126 346 Release 16 and ETSI
TS 103 769.</Definition>
   </Term>
   <Term termID="ROUTE">
      <Name xml:lang="en">Real-time Object delivery over Unidirectional Transport</Name>
      <Definition xml:lang="en">Per ATSC A/331, as profiled by ETSI TS 103 769.</Definition>
   </Term>
</ClassificationScheme>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ClassificationScheme uri="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2022">
   <Term termID="NORM">
      <Name xml:lang="en">NACK-Oriented Reliable Multicast</Name>
      <Definition xml:lang="en">Per IETF RFC 5740, as profiled by ETSI TS 103 769.</Definition>
   </Term>
   <Term termID="MSync">
      <Name xml:lang="en">MSync (Multicast Synchronization)</Name>
      <Definition xml:lang="en">Per IETF Internet Draft draft-bichot-msync-06</Definition>
      <Term termID="RTP">
         <Name xml:lang="en">MSync (Multicast Synchronization) conveyed over RTP</Name>
         <Definition xml:lang="en">Per IETF Internet Draft draft-bichot-msync-06</Definition>
      </Term>
   </Term>
</ClassificationScheme>
```

B.2 ForwardErrorCorrectionSchemeCS

The *ForwardErrorCorrectionSchemeCS* controlled vocabulary is a subset of the IANA registry of FEC Encoding IDs for Reliable Multicast Transport.

```
<?xml version="1.0" encoding="UTF-8"?>
<ClassificationScheme uri="urn:ietf:rmt:fec:encoding">
   <Term termID="0">
      <Name xml:lang="en">Compact No-Code FEC Scheme</Name>
      <Definition xml:lang="en">As specified in IETF RFC 5445 Section 3.</Definition>
   </Term>
   <Term termID="1">
      <Name xml:lang="en">Raptor Forward Error Correction Scheme for Object Delivery</Name>
      <Definition xml:lang="en">As specified in IETF RFC 5053.</Definition>
   </Term>
   <Term termID="2">
      <Name xml:lang="en"> Reed-Solomon Forward Error Correction Scheme for Object Delivery - Reed-Solomon Codes
over GF(2^^m)</Name>
      <Definition xml:lang="en">As specified in IETF RFC 5510.</Definition>
   </Term>
   <Term termID="5">
      <Name xml:lang="en"> Reed-Solomon Forward Error Correction Scheme for Object Delivery - Reed-Solomon Codes
over GF(2^^8)</Name>
      <Definition xml:lang="en">As specified in IETF RFC 5510.</Definition>
   </Term>
   <Term termID="6">
      <Name xml:lang="en">RaptorQ Forward Error Correction Scheme for Object Delivery</Name>
      <Definition xml:lang="en">As specified in IETF RFC 6330.</Definition>
   </Term>
</ClassificationScheme>
```

B.3 MulticastTransportObjectTypeCS

```
<?xml version="1.0" encoding="UTF-8"?>
<ClassificationScheme uri="urn:dvb:metadata:cs:MulticastTransportObjectTypeCS:2021">
   <Term termID="gateway-configuration">
      <Name xml:lang="en">Gateway configuration</Name>
      <Definition xml:lang="en">Multicast gateway configuration instance document.</Definition>
   </Term>
</ClassificationScheme>
```

Annex C (informative): Multicast session configuration examples

C.0 General

NOTE: The **`xsi:schemaLocation`** attribute is intentionally omitted from the root element of the examples presented in this annex. The system-dependent location of schema files should not be included in XML instance documents transmitted between different systems. As a convenience, this attribute is included in the example files distributed with the present document for the sole purpose of validation on a particular system.

C.1 Multicast server configuration instance document

The example multicast server configuration instance document below is valid for one day after receipt. It describes:

1) A multicast gateway configuration transport session using the FLUTE multicast media transport protocol that is simulcast to an IPv4 multicast group and to an IPv6 multicast group. Both multicast groups are protected by in-band Raptor AL-FEC. (The AL-FEC endpoint address does not need to be specified because it is the same as that of the enclosing transport session.)

   An object carousel provides the presentation manifests, initialisation segments and a channel ident logo for the "BBC One Scotland" and "BBC Two Scotland" services with a 1 second rotation period. In addition, an implementation-specific private extension has been added to this **`ObjectCarousel`** element instance using the mechanism described in clause A.1.

2) A multicast gateway configuration transport session using the ROUTE multicast media transport protocol that is simulcast to an IPv4 multicast group and to an IPv6 multicast group. Both multicast groups are protected by RaptorQ Repair Flows simulcast to different IPv4 and IPv6 multicast groups.

   An object carousel provides the presentation manifests, initialisation segments and a channel ident logo for the "BBC One Scotland" and "BBC Two Scotland" services with a 1 second rotation period. In addition, an implementation-specific private extension has been added to this **`ObjectCarousel`** element instance using the mechanism described in clause A.1.

3) A multicast session for the service "BBC One Scotland" associated with an MPEG-DASH MPD and an HLS Master Playlist. Explicit multicast transport object URI values (see clause 10.2.2.2) are signalled for use in the transmission of both presentation manifests. Content playback path patterns (*ibid.*) are provided for both presentation manifests.

   There are multicast transport sessions carrying two alternative vision service components, both transmitted in the FLUTE multicast media transport protocol. These are sent to different IPv4 multicast groups. There is also a single multicast transport session carrying the sound service component that is transmitted to a third IPv4 multicast group, again using the FLUTE multicast media transport protocol. All three service components are protected by in-band Raptor FEC. All three service components use the push content ingest method and chunked transmission mode.

   An in-band object carousel (see clause 10.2.3.14) is defined for each of the three multicast transport sessions to provide the presentation manifests, initialisation segments and a channel ident logo with a 1 second rotation period. The *Multicast server* is requested to compress the presentation manifests, if possible.

4) A multicast session for the service "BBC Two Scotland" associated with an MPEG-DASH MPD and an HLS Master Playlist. Explicit multicast transport object URI values (see clause 10.2.2.2) are signalled for use in the transmission of both presentation manifests. Content playback path patterns (*ibid.*) are provided for both presentation manifests.

   There is a single multicast transport session carrying the vision service component and another multicast transport session carrying the sound component, both transmitted in the ROUTE multicast media transport protocol, but to the same IPv4 multicast group. Although multiplexed together on the same multicast group,

the ROUTE Source Flows are addressed to different destination UDP ports and different LCT Channel numbers. The two service components are jointly protected by an out-of-band RaptorQ Repair Flow because the destination group address, destination port and LCT Channel number are identical for both sets of Forward Error Correction parameters. Both service components use the pull content ingest method and resource transmission mode.

An in-band object carousel (see clause 10.2.3.14) is defined for each of the two multicast transport sessions to provide the presentation manifests, initialisation segments and a channel ident logo with a 1 second rotation period. The *Multicast server* is requested to compress the presentation manifests, if possible.

```
<?xml version="1.0" encoding="UTF-8"?>
<MulticastServerConfiguration xmlns="urn:dvb:metadata:MulticastSessionConfiguration:2024"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" xmlns:private="tag:example.com,2024:metadata:cs:carousel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemaVersion="2" validityPeriod="P1D">

   <!-- Special Multicast transport session used to transmit configuration to the population of FLUTE-compatible
Multicast gateways -->
   <MulticastGatewayConfigurationTransportSession transportSecurity="integrityAndAuthenticity">
      <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
      <EndpointAddress>
         <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>232.99.1.1</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
      </EndpointAddress>
      <EndpointAddress>
         <NetworkSourceAddress>2001:DB8::4456:424D</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>FF3E::4950:4801</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
      </EndpointAddress>
      <BitRate average="200" maximum="200"></BitRate>
      <ForwardErrorCorrectionParameters>
         <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
         <OverheadPercentage>20</OverheadPercentage>
         <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent transport
session -->
      </ForwardErrorCorrectionParameters>
      <ObjectCarousel>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-two-scotland.jpg</ResourceLocator>
         <private:Digest algorithm="SHA256" salt="readysalted">
            ba9eb72e806835bd2de13f8386e382a627a1ea14e4edf30cee89b4c0e30f4286
         </private:Digest>
      </ObjectCarousel>
      <MulticastGatewayConfigurationMacro
key="PrimaryCDN">http://edge.cdn1.co.uk</MulticastGatewayConfigurationMacro>
      <MulticastGatewayConfigurationMacro
key="SecondaryCDN">http://edge.cdn2.co.uk</MulticastGatewayConfigurationMacro>
   </MulticastGatewayConfigurationTransportSession>

   <!-- Special Multicast transport session used to transmit configuration to the population of ROUTE-compatible
Multicast gateways -->
   <MulticastGatewayConfigurationTransportSession transportSecurity="integrityAndAuthenticity">
      <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE"
protocolVersion="1"/>
      <EndpointAddress>
         <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>232.98.1.1</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
```

```
      </EndpointAddress>
      <EndpointAddress>
         <NetworkSourceAddress>2001:DB8::4456:424D</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>FF3E::4950:4701</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
      </EndpointAddress>
      <BitRate average="200" maximum="200"/>
      <ForwardErrorCorrectionParameters>
         <SchemeIdentifier>urn:ietf:rmt:fec:encoding:6</SchemeIdentifier>  <!-- RaptorQ -->
         <OverheadPercentage>20</OverheadPercentage>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.98.1.2</NetworkDestinationGroupAddress>
            <TransportDestinationPort>8888</TransportDestinationPort>
            <MediaTransportSessionIdentifier>2</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <EndpointAddress>
            <NetworkSourceAddress>2001:DB8::4456:424D</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>FF3E::4950:4702</NetworkDestinationGroupAddress>
            <TransportDestinationPort>8888</TransportDestinationPort>
            <MediaTransportSessionIdentifier>2</MediaTransportSessionIdentifier>
         </EndpointAddress>
      </ForwardErrorCorrectionParameters>
      <ObjectCarousel>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-two-scotland.jpg</ResourceLocator>
         <private:Digest algorithm="SHA256" salt="readysalted">
            ba9eb72e806835bd2de13f8386e382a627a1ea14e4edf30cee89b4c0e30f4286
         </private:Digest>
      </ObjectCarousel>
      <MulticastGatewayConfigurationMacro
key="PrimaryCDN">http://edge.cdn1.fr</MulticastGatewayConfigurationMacro>
      <MulticastGatewayConfigurationMacro
key="SecondaryCDN">http://edge.cdn2.fr</MulticastGatewayConfigurationMacro>
   </MulticastGatewayConfigurationTransportSession>

   <!-- BBC One Scotland -->
   <MulticastSession serviceIdentifier="tag:bbc.co.uk;2019#bbc-one/scotland"
contentPlaybackAvailabilityOffset="PT3.84S">

      <!-- Origin locations of MPEG-DASH and HLS presentation manifests for this service -->
      <PresentationManifestLocator manifestId="bbc-one-scotland_mpd" contentType="application/dash+xml"
transportObjectURI="tag:bbc.co.uk;2019/manifests/dash#bbc-one/scotland" contentPlaybackPathPattern="*/bbc-
one/scotland/manifest.mpd">$PrimaryCDN$/simulcast/bbc-one/scotland/manifest.mpd</PresentationManifestLocator>
      <PresentationManifestLocator manifestId="bbc-one-scotland_hls"
contentType="application/vnd.apple.mpegURL" transportObjectURI="tag:bbc.co.uk;2019/manifests/hls#bbc-
one/scotland" contentPlaybackPathPattern="*/bbc-one/scotland/master.m3u8">$PrimaryCDN$/simulcast/bbc-
one/scotland/master.m3u8</PresentationManifestLocator>

      <!-- Reporting destination(s) used by the Multicast gateway for this Multicast session -->
      <MulticastGatewaySessionReporting>
         <ReportingLocator proportion="0.3" period="P5M" randomDelay="50"
reportSessionRunningEvents="true">https://reporting.isp.net/endpoint</ReportingLocator>
         <ReportingLocator proportion="0.1" period="PT1H" randomDelay="1000"
reportSessionRunningEvents="true">https://reporting.bbc.co.uk/dvb_multicast_gateway.cgi</ReportingLocator>
      </MulticastGatewaySessionReporting>

      <!-- First video component -->
      <MulticastTransportSession id="vision-low" start="2024-01-01T00:00:00" transmissionMode="chunked"
contentIngestMethod="push" sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
         <EndpointAddress>
```

```
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.100.1.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3922</TransportDestinationPort>
            <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="150000" maximum="150000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
            <OverheadPercentage>20</OverheadPercentage>
            <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent
transport session -->
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">$PrimaryCDN$/bbc-one_scotland/vision-low</BaseURL>
            <BaseURL relativeWeight="3">$SecondaryCDN$/bbc-one_scotland/vision-low</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-one-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="1" representationIdentifier="V1"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-one-scotland_hls"
mediaPlaylistLocator="$PrimaryCDN$/simulcast/bbc-one/scotland/V1.m3u8"/>
         <ObjectCarousel>
            <PresentationManifests targetAcquisitionLatency="PT1S" compressionPreferred="true"/>
            <InitSegments targetAcquisitionLatency="PT1S"/>
            <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         </ObjectCarousel>
      </MulticastTransportSession>

      <!-- Second video component -->
      <MulticastTransportSession id="vision-medium" start="2024-01-01T20:00:00" duration="PT2H"
transmissionMode="chunked" contentIngestMethod="push" sessionIdleTimeout="3840"
transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.100.1.2</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3923</TransportDestinationPort>
            <MediaTransportSessionIdentifier>2</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="250000" maximum="250000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
            <OverheadPercentage>20</OverheadPercentage>
            <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent
transport session -->
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">$PrimaryCDN$/bbc-one_scotland/vision-medium</BaseURL>
            <BaseURL relativeWeight="3">$SecondaryCDN$/bbc-one_scotland/vision-medium</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-one-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="1" representationIdentifier="V2"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-one-scotland_hls"
mediaPlaylistLocator="$PrimaryCDN$/simulcast/bbc-one/scotland/V2.m3u8"/>
         <ObjectCarousel>
            <PresentationManifests targetAcquisitionLatency="PT1S" compressionPreferred="true"/>
            <InitSegments targetAcquisitionLatency="PT1S"/>
            <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         </ObjectCarousel>
      </MulticastTransportSession>

      <!-- Single audio component -->
      <MulticastTransportSession id="sound" start="2024-01-01T00:00:00" transmissionMode="chunked"
contentIngestMethod="push" sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.100.2.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3924</TransportDestinationPort>
```

```
            <MediaTransportSessionIdentifier>5</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="150000" maximum="150000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
            <OverheadPercentage>20</OverheadPercentage>
            <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent
transport session -->
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">$PrimaryCDN$/bbc-one_scotland/sound</BaseURL>
            <BaseURL relativeWeight="3">$SecondaryCDN$/bbc-one_scotland/sound</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-one-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="2" representationIdentifier="A1"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-one-scotland_hls"
mediaPlaylistLocator="$PrimaryCDN$/simulcast/bbc-one/scotland/A1.m3u8"/>
         <ObjectCarousel>
            <PresentationManifests targetAcquisitionLatency="PT1S" compressionPreferred="true"/>
            <InitSegments targetAcquisitionLatency="PT1S"/>
            <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         </ObjectCarousel>
      </MulticastTransportSession>

   </MulticastSession>

   <!-- BBC Two Scotland -->
   <MulticastSession serviceIdentifier="tag:bbc.co.uk;2019#bbc-two/scotland"
contentPlaybackAvailabilityOffset="PT3.84S">

      <!-- Origin locations of MPEG-DASH and HLS presentation manifests for this service -->
      <PresentationManifestLocator manifestId="bbc-two-scotland_mpd" contentType="application/dash+xml"
transportObjectURI="tag:bbc.co.uk;2019/manifests/dash#bbc-two/scotland" contentPlaybackPathPattern="*/bbc-
two/scotland/manifest.mpd">$PrimaryCDN$/simulcast/bbc-two/scotland/manifest.mpd</PresentationManifestLocator>
      <PresentationManifestLocator manifestId="bbc-two-scotland_hls"
contentType="application/vnd.apple.mpegURL" transportObjectURI="tag:bbc.co.uk;2019/manifests/hls#bbc-
two/scotland" contentPlaybackPathPattern="*/bbc-two/scotland/master.m3u8">$PrimaryCDN$/simulcast/bbc-
two/scotland/master.m3u8</PresentationManifestLocator>

      <!-- No reporting destination(s) for this Multicast session -->

      <!-- Single video component -->
      <MulticastTransportSession id="vision" start="2024-01-01T00:00:00" transmissionMode="resource"
contentIngestMethod="pull" sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.200.1.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3922</TransportDestinationPort>
            <MediaTransportSessionIdentifier>11</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="150000" maximum="150000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:6</SchemeIdentifier>  <!-- RaptorQ -->
            <OverheadPercentage>20</OverheadPercentage>
            <EndpointAddress>
               <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
               <NetworkDestinationGroupAddress>232.200.2.1</NetworkDestinationGroupAddress>
               <TransportDestinationPort>3922</TransportDestinationPort>
               <MediaTransportSessionIdentifier>26</MediaTransportSessionIdentifier>
            </EndpointAddress>
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">$PrimaryCDN$/bbc-two_scotland/vision</BaseURL>
            <BaseURL relativeWeight="3">$SecondaryCDN$/bbc-two_scotland/vision</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-two-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="1" representationIdentifier="V1"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-two-scotland_hls"
mediaPlaylistLocator="$PrimaryCDN$/simulcast/bbc-two/scotland/V1.m3u8"/>
```

```
          <ObjectCarousel>
             <PresentationManifests targetAcquisitionLatency="PT1S" compressionPreferred="true"/>
             <InitSegments targetAcquisitionLatency="PT1S"/>
             <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-two/bbc-two-scotland.jpg</ResourceLocator>
          </ObjectCarousel>
       </MulticastTransportSession>

       <!-- Single audio component -->
       <MulticastTransportSession id="sound" start="2024-01-01T00:00:00" transmissionMode="resource"
contentIngestMethod="pull" sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
          <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE"
protocolVersion="1"/>
          <EndpointAddress>
             <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
             <NetworkDestinationGroupAddress>232.200.1.1</NetworkDestinationGroupAddress>
             <TransportDestinationPort>3923</TransportDestinationPort>
             <MediaTransportSessionIdentifier>15</MediaTransportSessionIdentifier>
          </EndpointAddress>
          <BitRate average="150000" maximum="150000"/>
          <ForwardErrorCorrectionParameters>
             <SchemeIdentifier>urn:ietf:rmt:fec:encoding:6</SchemeIdentifier>  <!-- RaptorQ -->
             <OverheadPercentage>20</OverheadPercentage>
             <EndpointAddress>
                <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
                <NetworkDestinationGroupAddress>232.200.2.1</NetworkDestinationGroupAddress>
                <TransportDestinationPort>3922</TransportDestinationPort>
                <MediaTransportSessionIdentifier>26</MediaTransportSessionIdentifier>
             </EndpointAddress>
          </ForwardErrorCorrectionParameters>
          <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
             <BaseURL relativeWeight="7">$PrimaryCDN$/bbc-one_scotland/sound</BaseURL>
             <BaseURL relativeWeight="3">$SecondaryCDN$/bbc-one_scotland/sound</BaseURL>
          </UnicastRepairParameters>
          <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-two-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="2" representationIdentifier="A1"/>
          <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-two-scotland_hls"
mediaPlaylistLocator="$PrimaryCDN$/simulcast/bbc-two/scotland/A1.m3u8"/>
          <ObjectCarousel>
             <PresentationManifests targetAcquisitionLatency="PT1S" compressionPreferred="true"/>
             <InitSegments targetAcquisitionLatency="PT1S"/>
             <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-two/bbc-two-scotland.jpg</ResourceLocator>
          </ObjectCarousel>
       </MulticastTransportSession>

    </MulticastSession>

    <MulticastServerConfigurationMacro key="PrimaryCDN">http://mcast-
srv.provider.net</MulticastServerConfigurationMacro>

    <!-- Reporting destination(s) used by the Multicast gateway for consolidated reports about all multicast
sessions -->
    <MulticastGatewaySessionReporting>
       <ReportingLocator proportion="0.3" period="P5M" randomDelay="50"
reportSessionRunningEvents="false">https://reporting.isp.net/endpoint</ReportingLocator>
       <ReportingLocator proportion="0.1" period="PT1H" randomDelay="1000"
reportSessionRunningEvents="false">https://reporting.bbc.co.uk/dvb_multicast_gateway.cgi</ReportingLocator>
    </MulticastGatewaySessionReporting>

</MulticastServerConfiguration>
```

C.2 Multicast gateway configuration bootstrap instance document

The parameters of the multicast gateway configuration transport session can be provided to a *Multicast gateway* at reference point **C_{MR}** in a separate multicast session configuration instance document. This bootstrap configuration then enables the *Multicast gateway* to acquire the remainder of its multicast gateway configuration (see clause C.3 below) from a multicast gateway configuration transport session transmitted at reference point **M**.

The example multicast gateway configuration bootstrap instance document below is valid for one day after receipt. It corresponds to the two multicast gateway configuration transport sessions described in the multicast server configuration example in clause C.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<MulticastGatewayConfiguration xmlns="urn:dvb:metadata:MulticastSessionConfiguration:2024"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" xmlns:private="tag:example.com,2024:metadata:cs:carousel"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" schemaVersion="2" validityPeriod="P1D">

   <!-- Special Multicast transport session used to transmit configuration to the population of Multicast
gateways -->
   <MulticastGatewayConfigurationTransportSession transportSecurity="integrityAndAuthenticity">
      <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
      <EndpointAddress>
         <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>232.99.1.1</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
      </EndpointAddress>
      <EndpointAddress>
         <NetworkSourceAddress>2001:DB8::4456:424D</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>FF3E::4950:4801</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
      </EndpointAddress>
      <BitRate average="200" maximum="200"/>
      <ForwardErrorCorrectionParameters>
         <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
         <OverheadPercentage>20</OverheadPercentage>
         <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent transport
session -->
      </ForwardErrorCorrectionParameters>
      <ObjectCarousel>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-two-scotland.jpg</ResourceLocator>
         <private:Digest algorithm="SHA256" salt="readysalted">
            ba9eb72e806835bd2de13f8386e382a627a1ea14e4edf30cee89b4c0e30f4286
         </private:Digest>
      </ObjectCarousel>
   </MulticastGatewayConfigurationTransportSession>

   <!-- Special Multicast transport session used to transmit configuration to the population of Multicast
gateways -->
   <MulticastGatewayConfigurationTransportSession transportSecurity="integrityAndAuthenticity">
      <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE"
protocolVersion="1"/>
      <EndpointAddress>
         <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>232.98.1.1</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
```

```
      </EndpointAddress>
      <EndpointAddress>
         <NetworkSourceAddress>2001:DB8::4456:424D</NetworkSourceAddress>
         <NetworkDestinationGroupAddress>FF3E::4950:4701</NetworkDestinationGroupAddress>
         <TransportDestinationPort>9999</TransportDestinationPort>
         <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
      </EndpointAddress>
      <BitRate average="200" maximum="200"/>
      <ForwardErrorCorrectionParameters>
         <SchemeIdentifier>urn:ietf:rmt:fec:encoding:6</SchemeIdentifier>  <!-- RaptorQ -->
         <OverheadPercentage>20</OverheadPercentage>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.99.1.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>8888</TransportDestinationPort>
            <MediaTransportSessionIdentifier>2</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <EndpointAddress>
            <NetworkSourceAddress>2001:DB8::4456:424D</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>FF3E::4950:4702</NetworkDestinationGroupAddress>
            <TransportDestinationPort>8888</TransportDestinationPort>
            <MediaTransportSessionIdentifier>2</MediaTransportSessionIdentifier>
         </EndpointAddress>
      </ForwardErrorCorrectionParameters>
      <ObjectCarousel aggregateTransportSize="1945320" aggregateContentSize="2040900">
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-one/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         <PresentationManifests serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland"
transportSessionIdRef="vision-low" targetAcquisitionLatency="PT1S"/>
         <InitSegments serviceIdRef="tag:bbc.co.uk;2019#bbc-two/scotland" transportSessionIdRef="vision-low"
targetAcquisitionLatency="PT1S"/>
         <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-two-scotland.jpg</ResourceLocator>
         <private:Digest algorithm="SHA256" salt="readysalted">
            ba9eb72e806835bd2de13f8386e382a627a1ea14e4edf30cee89b4c0e30f4286
         </private:Digest>
      </ObjectCarousel>
   </MulticastGatewayConfigurationTransportSession>

</MulticastGatewayConfiguration>
```

C.3 Multicast gateway configuration instance document

The example multicast gateway configuration instance document below is valid until 1st January 2024. It corresponds to the multicast server configuration in clause C.1 and may be transmitted on the multicast gateway configuration transport session described in clause C.2.

```
<?xml version="1.0" encoding="UTF-8"?>
<MulticastGatewayConfiguration xmlns="urn:dvb:metadata:MulticastSessionConfiguration:2024"
xmlns:ext="urn:dvb:metadata:Extensibility:2024" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
schemaVersion="2" validUntil="2024-01-01T22:26:37">

   <!-- BBC One Scotland -->
   <MulticastSession serviceIdentifier="tag:bbc.co.uk;2019#bbc-one/scotland"
contentPlaybackAvailabilityOffset="PT3.84S">

      <!-- Origin locations of MPEG-DASH and HLS presentation manifests for this service -->
      <PresentationManifestLocator manifestId="bbc-one-scotland_mpd" contentType="application/dash+xml"
transportObjectURI="tag:bbc.co.uk;2019/manifests/dash#bbc-one/scotland" contentPlaybackPathPattern="*/bbc-
one/scotland/manifest.mpd">http://edge.cdn1.co.uk/simulcast/bbc-
one/scotland/manifest.mpd</PresentationManifestLocator>
      <PresentationManifestLocator manifestId="bbc-one-scotland_hls"
contentType="application/vnd.apple.mpegURL" transportObjectURI="tag:bbc.co.uk;2019/manifests/hls#bbc-
one/scotland" contentPlaybackPathPattern="*/bbc-one/scotland/master.m3u8">http://edge.cdn1.co.uk/simulcast/bbc-
one/scotland/master.m3u8</PresentationManifestLocator>
```

```
      <!-- Reporting destination(s) used by the Multicast gateway for this Multicast session -->
      <MulticastGatewaySessionReporting>
         <ReportingLocator proportion="0.3" period="P5M" randomDelay="50"
reportSessionRunningEvents="true">https://reporting.isp.net/endpoint</ReportingLocator>
         <ReportingLocator proportion="0.1" period="PT1H" randomDelay="1000"
reportSessionRunningEvents="true">https://reporting.bbc.co.uk/dvb_multicast_gateway.cgi</ReportingLocator>
      </MulticastGatewaySessionReporting>

      <!-- First video component -->
      <MulticastTransportSession id="vision-low" start="2024-01-01T00:00:00" transmissionMode="chunked"
sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.100.1.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3922</TransportDestinationPort>
            <MediaTransportSessionIdentifier>1</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="150000" maximum="150000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
            <OverheadPercentage>20</OverheadPercentage>
            <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent
transport session -->
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">http://edge.cdn1.co.uk/bbc-one_scotland/vision-low</BaseURL>
            <BaseURL relativeWeight="3">http://edge.cdn2.co.uk/bbc-one_scotland/vision-low</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-one-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="1" representationIdentifier="V1"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-one-scotland_hls"
mediaPlaylistLocator="http://edge.cdn1.co.uk/simulcast/bbc-one/scotland/V1.m3u8"/>
         <ObjectCarousel aggregateTransportSize="1398430" aggregateContentSize="1566815">
            <PresentationManifests targetAcquisitionLatency="PT1S"/>
            <InitSegments targetAcquisitionLatency="PT1S"/>
            <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         </ObjectCarousel>
      </MulticastTransportSession>

      <!-- Second video component -->
      <MulticastTransportSession id="vision-medium" start="2024-01-01T20:00:00" duration="PT2H"
transmissionMode="chunked" sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.100.1.2</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3923</TransportDestinationPort>
            <MediaTransportSessionIdentifier>2</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="250000" maximum="250000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
            <OverheadPercentage>20</OverheadPercentage>
            <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent
transport session -->
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">http://edge.cdn1.co.uk/bbc-one_scotland/vision-medium</BaseURL>
            <BaseURL relativeWeight="3">http://edge.cdn2.co.uk/bbc-one_scotland/vision-medium</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-one-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="1" representationIdentifier="V2"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-one-scotland_hls"
mediaPlaylistLocator="http://edge.cdn1.co.uk/simulcast/bbc-one/scotland/V2.m3u8"/>
         <ObjectCarousel aggregateTransportSize="1495526" aggregateContentSize="1600056">
            <PresentationManifests targetAcquisitionLatency="PT1S"/>
            <InitSegments targetAcquisitionLatency="PT1S"/>
```

```
            <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         </ObjectCarousel>
      </MulticastTransportSession>

      <!-- Single audio component -->
      <MulticastTransportSession id="sound" start="2024-01-01T00:00:00" transmissionMode="chunked"
sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.100.2.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3924</TransportDestinationPort>
            <MediaTransportSessionIdentifier>5</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="150000" maximum="150000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:1</SchemeIdentifier>  <!-- Raptor -->
            <OverheadPercentage>20</OverheadPercentage>
            <!-- Endpoint address omitted for FLUTE Sessions since this is always the same as the parent
transport session -->
         </ForwardErrorCorrectionParameters>
         <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
            <BaseURL relativeWeight="7">http://edge.cdn1.co.uk/bbc-one_scotland/sound</BaseURL>
            <BaseURL relativeWeight="3">http://edge.cdn2.co.uk/bbc-one_scotland/sound</BaseURL>
         </UnicastRepairParameters>
         <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-one-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="2" representationIdentifier="A1"/>
         <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-one-scotland_hls"
mediaPlaylistLocator="http://edge.cdn1.co.uk/simulcast/bbc-one/scotland/A1.m3u8"/>
         <ObjectCarousel aggregateTransportSize="1236871" aggregateContentSize="1394220">
            <PresentationManifests targetAcquisitionLatency="PT1S"/>
            <InitSegments targetAcquisitionLatency="PT1S"/>
            <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-one/bbc-one-scotland.jpg</ResourceLocator>
         </ObjectCarousel>
      </MulticastTransportSession>

   </MulticastSession>

   <!-- BBC Two Scotland -->
   <MulticastSession serviceIdentifier="tag:bbc.co.uk;2019#bbc-two/scotland"
contentPlaybackAvailabilityOffset="PT3.84S">

      <!-- Origin locations of MPEG-DASH and HLS presentation manifests for this service -->
      <PresentationManifestLocator manifestId="bbc-two-scotland_mpd" contentType="application/dash+xml"
transportObjectURI="tag:bbc.co.uk;2019/manifests/dash#bbc-two/scotland" contentPlaybackPathPattern="*/bbc-
two/scotland/manifest.mpd">http://edge.cdn1.co.uk/simulcast/bbc-
two/scotland/manifest.mpd</PresentationManifestLocator>
      <PresentationManifestLocator manifestId="bbc-two-scotland_hls"
contentType="application/vnd.apple.mpegURL" transportObjectURI="tag:bbc.co.uk;2019/manifests/hls#bbc-
two/scotland" contentPlaybackPathPattern="*/bbc-two/scotland/master.m3u8">http://edge.cdn1.co.uk/simulcast/bbc-
two/scotland/master.m3u8</PresentationManifestLocator>

      <!-- No reporting destination(s) for this Multicast session -->

      <!-- Single video component -->
      <MulticastTransportSession id="vision" start="2024-01-01T00:00:00" transmissionMode="resource"
sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
         <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE"
protocolVersion="1"/>
         <EndpointAddress>
            <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
            <NetworkDestinationGroupAddress>232.200.1.1</NetworkDestinationGroupAddress>
            <TransportDestinationPort>3922</TransportDestinationPort>
            <MediaTransportSessionIdentifier>11</MediaTransportSessionIdentifier>
         </EndpointAddress>
         <BitRate average="150000" maximum="150000"/>
         <ForwardErrorCorrectionParameters>
            <SchemeIdentifier>urn:ietf:rmt:fec:encoding:6</SchemeIdentifier>  <!-- RaptorQ -->
            <OverheadPercentage>20</OverheadPercentage>
            <EndpointAddress>
```

```
                <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
                <NetworkDestinationGroupAddress>232.200.2.1</NetworkDestinationGroupAddress>
                <TransportDestinationPort>3922</TransportDestinationPort>
                <MediaTransportSessionIdentifier>26</MediaTransportSessionIdentifier>
             </EndpointAddress>
          </ForwardErrorCorrectionParameters>
          <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
             <BaseURL relativeWeight="7">http://edge.cdn1.co.uk/bbc-two_scotland/vision</BaseURL>
             <BaseURL relativeWeight="3">http://edge.cdn2.co.uk/bbc-two_scotland/vision</BaseURL>
          </UnicastRepairParameters>
          <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-two-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="1" representationIdentifier="V1"/>
          <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-two-scotland_hls"
mediaPlaylistLocator="http://edge.cdn1.co.uk/simulcast/bbc-two/scotland/V1.m3u8"/>
          <ObjectCarousel aggregateTransportSize="1896351" aggregateContentSize="1998400">
             <PresentationManifests targetAcquisitionLatency="PT1S"/>
             <InitSegments targetAcquisitionLatency="PT1S"/>
             <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-two/bbc-two-scotland.jpg</ResourceLocator>
          </ObjectCarousel>
       </MulticastTransportSession>

       <!-- Single audio component -->
       <MulticastTransportSession id="sound" start="2024-01-01T00:00:00" transmissionMode="resource"
sessionIdleTimeout="3840" transportSecurity="integrityAndAuthenticity">
          <TransportProtocol protocolIdentifier="urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE"
protocolVersion="1"/>
          <EndpointAddress>
             <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
             <NetworkDestinationGroupAddress>232.200.1.1</NetworkDestinationGroupAddress>
             <TransportDestinationPort>3923</TransportDestinationPort>
             <MediaTransportSessionIdentifier>15</MediaTransportSessionIdentifier>
          </EndpointAddress>
          <BitRate average="150000" maximum="150000"/>
          <ForwardErrorCorrectionParameters>
             <SchemeIdentifier>urn:ietf:rmt:fec:encoding:6</SchemeIdentifier>  <!-- RaptorQ -->
             <OverheadPercentage>20</OverheadPercentage>
             <EndpointAddress>
                <NetworkSourceAddress>10.1.100.1</NetworkSourceAddress>
                <NetworkDestinationGroupAddress>232.200.2.1</NetworkDestinationGroupAddress>
                <TransportDestinationPort>3922</TransportDestinationPort>
                <MediaTransportSessionIdentifier>26</MediaTransportSessionIdentifier>
             </EndpointAddress>
          </ForwardErrorCorrectionParameters>
          <UnicastRepairParameters transportObjectReceptionTimeout="500" fixedBackOffPeriod="10"
randomBackOffPeriod="20">
             <BaseURL relativeWeight="7">http://edge.cdn1.co.uk/bbc-one_scotland/sound</BaseURL>
             <BaseURL relativeWeight="3">http://edge.cdn2.co.uk/bbc-one_scotland/sound</BaseURL>
          </UnicastRepairParameters>
          <ServiceComponentIdentifier xsi:type="DASHComponentIdentifierType" manifestIdRef="bbc-two-
scotland_mpd" periodIdentifier="period1" adaptationSetIdentifier="2" representationIdentifier="A1"/>
          <ServiceComponentIdentifier xsi:type="HLSComponentIdentifierType" manifestIdRef="bbc-two-scotland_hls"
mediaPlaylistLocator="http://edge.cdn1.co.uk/simulcast/bbc-two/scotland/A1.m3u8"/>
          <ObjectCarousel aggregateTransportSize="1478924" aggregateContentSize="1554980">
             <PresentationManifests targetAcquisitionLatency="PT1S"/>
             <InitSegments targetAcquisitionLatency="PT1S"/>
             <ResourceLocator targetAcquisitionLatency="PT15S"
revalidationPeriod="PT1M">http://media.bbc.co.uk/idents/bbc-two/bbc-two-scotland.jpg</ResourceLocator>
          </ObjectCarousel>
       </MulticastTransportSession>

    </MulticastSession>

    <!-- Reporting destination(s) used by the Multicast gateway for consolidated reports about all multicast
sessions -->
    <MulticastGatewaySessionReporting>
       <ReportingLocator proportion="0.3" period="P5M" randomDelay="50"
reportSessionRunningEvents="false">https://reporting.isp.net/endpoint</ReportingLocator>
       <ReportingLocator proportion="0.1" period="PT1H" randomDelay="1000"
reportSessionRunningEvents="false">https://reporting.bbc.co.uk/dvb_multicast_gateway.cgi</ReportingLocator>
    </MulticastGatewaySessionReporting>

</MulticastGatewayConfiguration>
```

Annex D (informative): Media object mapping

Table D-1 below summarizes, for several common ingest object types, the end-to-end mapping between media objects ingested by the *Multicast server* and those exposed to the *Content playback* function by the *Multicast gateway* at reference point **L**.

Table D-1: Summary of object ingest and delivery for regular and low latency provisioned content.

Ingest object (P_{in}' or O_{in})	**Ingest method**	**Multicast server mapping**	**Multicast transport object (M)**	**Multicast gateway conversion**	**Playback delivery object**	**Content playback retrieval (L)**
DASH Segment, identified by URL.	Regular HTTP response. HTTP/1.1 chunked transfer coding optional.	Each DASH Segment maps to a different multicast transport object. Mapping from content ingest object URL to multicast transport object URI may be specified by multicast media transport protocol. When HTTP/1.1 chunked transfer coding is used, one or more HTTP chunks are combined into one multicast transport object.	DASH Segment	Conversion between multicast transport object URI and playback delivery object URL may be specified by multicast media transport protocol.	DASH Segment	HTTP Request for a playback delivery object (DASH Segment)
CMAF chunks, each composed of one or more HTTP chunks, but only the parent DASH Segment has a URL.	HTTP/1.1 chunked transfer coding.	Combine one or multiple HTTP chunks into one multicast transport object with its own URI. Mapping from transport object URLs to DASH Segment URL.	Part of a DASH Segment (minimum size: one HTTP chunk, maximum size: one CMAF chunk).	Assemble received multicast transport objects into a DASH Segment. Conversion may be needed between multicast transport object URI and playback delivery object URL.	Depending on Content playback requests: *Low Latency:* CMAF Chunks delivered at **L** using HTTP chunked transfer coding and minimal buffering. *Conservative player:* Regular DASH Segment (progressive reception).	
DASH Segment composed of multiple CMAF chunks.		Each DASH Segment is mapped to a multicast transport object with its own a URI. CMAF chunks, parts of a DASH segment, are streamed through. Mapping from DASH Segments and CMAF Chunks to multicast transport Objects.	Parts of a transport object which correspond to streamed through CMAF chunks.	Received CMAF chunks. Mapping from multicast transport objects to DASH Segments and CMAF Chunks.		

Annex E (informative): End-to-end worked example

E.1 Mapping of content ingest URL to multicast transport object URI by Multicast server

See clause 8.3.3.

Content ingest URL at reference point $\mathbf{O_{in}}$	http://account9876.cdn.com/service4567/as1/rep1/**segment1234.m4s**
Content ingest URL at reference point $\mathbf{P_{in}'}$	http://multicastserver.isp.net/service4567/as1/rep1/**segment1234.m4s**
Multicast transport object URI at reference point **M**	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s**

E.2 Mapping of multicast transport object URI to unicast repair URL by Multicast gateway

See clause 9.2.2.

Multicast transport object URI at reference point **M**	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s**
Unicast repair URL at reference point **A**	http://account9876.cdn.com/service4567/as1/rep1/**segment1234.m4s**

Where the URL prefix account9876.cdn.com/service4567/as1/rep1/ is derived from the **`UnicastRepairParameters/BaseURL`** element of the multicast transport session.

E.3 Example HTTP-based unicast repair messages

See clause 9.2.

Assuming that the *Multicast gateway* partially receives a multicast transport object with the unicast repair URL http://www.example.com/service1/000018.m4s and the entity tag *3a9aa6-58f5b718495de*, the *Multicast gateway* sends a repair request to the *Content hosting* function to fetch the missing bytes using single and multiple byte range as shown in table E.3-1.

Table E.3-1: Example unicast repair request and response HTTP messages

	HTTP request message	HTTP response message
Single byte range repair	**GET** /service1/000018.m4s **HTTP/1.1** **If-Range:** "3a9aa6-58f5b718495de" **Range:** bytes=500-999 **Host:** www.example.com	**HTTP/1.1 206** Partial Content **Date:** Mon, 05 Aug 2019 09:37:55 GMT **Last-Modified:** Mon, 05 Aug 2019 09:36:32 GMT **Accept-Ranges:** bytes **Content-Range:** bytes 500-999/3840678 …
Multiple byte range repair	**GET** /service1/000018.m4s **HTTP/1.1** **If-Range:** "3a9aa6-58f5b718495de" **Range:** bytes=500-999, 1500-1999 **Host:** www.example.com	**HTTP/1.1 206** Partial Content **Date:** Mon, 05 Aug 2019 09:38:41 GMT **Last-Modified:** Mon, 05 Aug 2019 09:36:32 GMT **Accept-Ranges:** bytes **Content-Length:** 1150 **Content-Type:** multipart/byteranges; boundary=THIS_STRING_SEPARATES **--** THIS_STRING_SEPARATES **Content-Range:** bytes 500-999/3840678 … -- THIS_STRING_SEPARATES **Content-Range:** bytes 1500-1999/3840678 … -- THIS_STRING_SEPARATES--

E.4 Mapping of multicast transport object URI to playback delivery object URL

The format of the playback delivery object URL is at the discretion of the *Multicast gateway* (see clause 8.4.4). The following example illustrates two different possible formats for the playback delivery object URL exposed at reference point **L**: one with a local host name that can be resolved by a DNS resolver in the local network and one with an unresolved IP address. The latter addressing format may be appropriate in the case where the *Multicast gateway* is deployed in a home gateway device (see clause 6.2) lacking local DNS resolution capability. In both examples, the *Multicast gateway* has inserted a session identifier into the URL path that enables it to track usage independently for each playback session.

Multicast transport object URI at reference point **M**	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s**
Playback delivery object URL at reference point **L** *(local host name)*	http://gateway.local:8080/session5678/as1/rep1/**segment1234.m4s**
Playback delivery object URL at reference point **L** *(unresolved local IP address)*	http://192.0.2.1:8080/session5678/as1/rep1/**segment1234.m4s**

Annex F (normative): FLUTE-based multicast media transport protocol

F.0 Introduction

The 3GPP FLUTE profile specified in this annex is based on the 3GPP MBMS Download Profile as defined in clause L.4 of ETSI TS 126 346 [17] and on MBMS DASH Streaming as defined in clause 5.6 of [17]. MBMS DASH Streaming allows the sending of both non-real-time content and DASH-formatted content via MBMS.

This profile differs from the 3GPP FLUTE profile in the following ways:

- The MBMS User Service Discovery/Announcement mechanism specified in clause 5.2 of [17], including the use of SDP, is replaced by the multicast session configuration instance document specified in clause 10.2.
- The use of RTSP for session setup and control, specified in clause L.4.6 of [17], is not required.
- The use of Byte-Range based File Repair, as specified in clause 9.3.6.2 of [17] is permitted, as specified in clause F.3.2 below. However, the operation of the unicast repair procedure in the *Multicast gateway* is governed by the contents of the multicast gateway configuration, and the **`Alternate-Content-Location-1`** and **`Alternate-Content-Location-2`** elements in the FDT Instance shall be ignored. In addition, the usage of the Associated Delivery Procedure Fragment, which contains some additional parameters for the File Repair Procedure configuration, is excluded from this profile.
- Symbol-based file repair over reference points **U** or **A** (as defined in [17]) is not supported by this profile, because this profile does not support the Associated Delivery Procedure Description Fragment.
- A means of subdividing a media object into smaller multicast transport objects for low-latency operations is specified in clause F.2.2.
- In deployments with low rates of packet loss the Close Object (B) LCT header flag may be used by the *Multicast server.* In this case, the flag shall be set only in the last FLUTE packet comprising a multicast transport object.
- The File Delivery Table (FDT) is extended to include a more robust content integrity protection mechanism specified in clause F.2.3, which is utilised by the message signature mechanism specified in clause F.2.4 to provide authenticity protection to multicast transport objects.

F.1 Signalling in the multicast session configuration

The use of the multicast media transport protocol specified in this annex shall be signalled in the multicast session configuration as follows (see clause 10.2.3.8):

TransportProtocol/**@protocolIdentifier**	TransportProtocol/**@protocolVersion**
urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:FLUTE	*1*

The multicast transmission mode, as specified in clause 10.2.3.5, shall indicate whether multicast transport objects are formatted using resource transmission mode (per clause F.2.1 below) or chunked transmission mode (per clause F.2.2).

If every multicast transport object in a multicast transport session is protected by the integrity checks specified in clause F.2.3, the transport security mode "Integrity" (see clause 10.2.3.6) shall be indicated in the multicast session configuration.

If every multicast transport object in a multicast transport session is protected by the integrity and authenticity protection mechanism specified in clause F.2.4, the transport security mode *integrityAndAuthenticity* shall be indicated in the multicast session configuration.

If neither integrity nor authenticity is protected, the transport security mode "None" shall be indicated in the multicast session configuration.

The IP source address (in the case of source-specific multicast), IP multicast group destination address and destination UDP port number of the FLUTE Session shall be signalled in the multicast transport endpoint address, as specified in clause 10.2.3.9. The LCT Transport Session Identifier shall be conveyed in the multicast media transport protocol session identifier, as also specified in clause 10.2.3.9.

Application Level Forward Error Correction is always carried in band (i.e. in the same FLUTE Session as the multicast transport objects protected). Accordingly, the **`ForwardErrorCorrectionParameters/EndpointAddress`** element (see clause 10.2.3.11) should be omitted.

Omitting the **`ForwardErrorCorrectionParameters`** element altogether shall imply that the "Compact No-Code" AL-FEC scheme is in use for the multicast transport session in question.

F.2 Mapping of multicast transport objects to 3GPP FLUTE Transport Objects

F.2.0 General

Each multicast transport session shall be realized as a FLUTE Session comprising one FLUTE Channel (LCT channel), per the MBMS Download Profile in clause L.4 of [17]. Each multicast session may comprise one or more of these FLUTE Sessions, e.g. one FLUTE Session for each service component.

A multicast transport object is realized in this profile by a 3GPP FLUTE Transport Object. A multicast transport object may be a media object (for resource transmission mode, per clause F.2.1) or a byte range of a media object (for chunked transmission mode, per clause F.2.2).

The multicast transport object URI shall be signalled as the **`File/`**`@Content-Location` attribute of exactly one file description entry in a FLUTE File Delivery Table (FDT) Instance. The size of the multicast transport object is signalled as the **`File/`**`@Content-Length` attribute.

NOTE: The FDT Instance is conveyed as TOI=0 in the same FLUTE Session as the FLUTE Transport Object it describes.

F.2.1 Resource transmission mode

When resource transmission mode is in use (see clause 10.2.3.5), the FLUTE FDT Instance describes the size (`@Content-Length` attribute) and multicast transport object URI (`@Content-Location` attribute) of exactly one multicast transport object.

In this transmission mode the *Multicast gateway* may start serving a playback delivery object at reference point **L** once at least the FLUTE FDT Instance for the corresponding FLUTE Transport Object is received.

F.2.2 Chunked transmission mode

If chunked transmission mode is signalled (implicitly or explicitly) for a particular multicast transport session (see clause 10.2.3.5), this indicates that a single ingest object is transmitted as a sequence of one or more related multicast transport objects. This allows for low-latency operation. In this transmission mode additional components (specified below) are present in the multicast transport object URI.

The multicast transport object is preferably a complete CMAF chunk [i.11]. Alternatively, the *Multicast server* may simply transmit ingested HTTP chunks as multicast transport objects. Accordingly, the `@Content-Length` attribute within the FLUTE FDT Instance shall reflect the size of the multicast transport object being conveyed.

The multicast transport object URI carried by the `@Content-Location` attribute shall be suffixed with a fragment identifier component (i.e. *fragment* as specified in section 3.5 of IETF RFC 3986 [15]) indicating the offset of the first byte of this multicast transport object within the overall media object: the **chunk offset**. If some chunks are not correctly received by the *Multicast gateway*, the chunk offset may be used by the *Multicast gateway* to identify and repair the missing byte range of the affected media object according to clauses F.3.1 and/or F.3.2 below.

All multicast transport objects belonging to the same media object shall be signalled with the same URI prefix (i.e. *hier-part* in IETF RFC 3986 [15]). When the *Multicast gateway* receives a multicast transport object with byte offset 0, the *Multicast gateway* shall start assembling a new playback delivery object and may respond to requests from the *Content playback* function for that playback delivery object. The playback delivery object URL exposed by the *Multicast gateway* at reference point **L** shall not include any fragment identifier component (i.e. the chunk offset) nor any query component. Requests for playback delivery objects at reference point **L** shall elicit an HTTP chunked transfer coding response until such time as the multicast transport object representing the last chunk in the playback delivery object has been received.

The *Multicast server* shall signal the end of a media object explicitly at reference point **M**. If a *Multicast server* has determined that the last HTTP chunk of an ingest media object has been received at reference point $\mathbf{P_{in}'}$ or $\mathbf{O_{in}}$, it may add an isLast query component (i.e. *query* as specified in section 3.5 of IETF RFC 3986 [15]) to the multicast transport object URI to signal completion. Otherwise, if a *Multicast server* does not have this information at the point when it starts transmitting the last multicast transport object, it should instead transmit a zero-length multicast transport object.

A *Multicast gateway* detects the completion of a media object either by receiving a zero-length multicast transport object, or by receiving a multicast transport object carrying the isLast URI query component. When the multicast transport objects used to convey a media object contain authenticity assertions as specified in clause F.2.4, the final multicast transport object shall include a representation digest of the whole media object, as specified in clause F.2.3.1 below.

In chunked transmission mode, not all multicast transport objects contain a Random Access Point marker. The *Multicast server* may add a hasRandomAccessPoint URI query component to indicate that this multicast transport object is appropriate for commencing playback.

NOTE 1: Signalling of this information is in particular beneficial for unidirectional transmissions, i.e. when unicast start-up or unicast repair is not supported.

NOTE 2: Details of end-to-end operations for fast start-up are expected in a later specification phase.

For example:

Multicast transport object URI (**`File/`**`@Content-Location`) for *first* multicast transport object	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s?hasRandomAccessPoint#0**
Multicast transport object URI (**`File/`**`@Content-Location`) for *second* multicast transport object	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s#5321**
etc.	
Multicast transport object URI (**`File/`**`@Content-Location`) for *last* multicast transport object	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s?isLast&hasRandomAccessPoint#154360**
Resulting playback delivery object URL at reference point **L**	http://gateway.local:8080/session5678/as1/rep1/**segment1234.m4s**

In order to achieve low-latency operation, the *Multicast gateway* should start serving the content of a multicast transport object as a playback delivery object at reference point **L** as soon as the FLUTE FDT Instance for the first multicast transport object comprising that playback delivery object is received.

F.2.3 Multicast transport object integrity protection

F.2.3.0 General

The integrity of a multicast transport object may be protected in 3GPP FLUTE by an MD5 digest conveyed in the optional **`File/`**`@Content-MD5` attribute of the FDT Instance as specified in clause L.4.4 of ETSI TS 126 346 [17]. However, the MD5 hashing algorithm is known to be vulnerable to collision attacks [i.19]. It is therefore not sufficiently secure for use with the cryptographic authenticity check specified in clause F.2.4 on an insecure network because a bad actor may use a crafted MD5 collision attack to spoof the authenticity check.

To address this potential vulnerability, the integrity of a multicast transport object may alternatively or additionally be protected by a more robust content digest conveyed in the optional **`File/`**`@Content-Digest` extension attribute specified in clause F.2.3.1.

F.2.3.1 Content-Digest extension attribute

The FLUTE FDT schema is here extended to introduce new content digest (**File/**`@Content-Digest`) and representation digest (**File/**`@Repr-Digest`) attributes modelled respectively on the HTTP `Content-Digest` and `Repr-Digest` headers specified in RFC 9530 [20]. The XML Schema of the **File**`@Content-Digest` and **File/**`@Repr-Digest` attributes is specified in clause F.2.5.1. An example of a FLUTE FDT instance containing a **File**`@Content-Digest` and **File/**`@Repr-Digest` attributes is provided in clause F.2.5.2.

- The **File**`@Content-Digest` attribute shall be used to convey a digest hash of the payload of the FLUTE multicast transport object. The value of the **File**`@Content-Digest` attribute shall contain a string syntactically identical to that specified in section 2 of [20], where the hashing algorithm used is conveyed, followed by an equals character "=" and a colon character ":", and then the output of the digest calculation is carried encoded in Base64 according to RFC 4648 [9], followed by another colon character ":" to terminate the string.

- Where the multicast transport object conveys only a part of a media object, as used by the chunked transmission mode specified in clause F.2.2, the **File/**`@Repr-Digest` attribute shall be used to convey a digest hash (the representation digest) of the whole media object in the FDT that describes the final part of the media object. The value shall take the same form as the **File/**`@Content-Digest` attribute above.

F.2.4 Multicast transport object authenticity protection

F.2.4.0 General

3GPP FLUTE does not specify a mechanism to protect the authenticity of multicast transport objects conveyed in a FLUTE Session. Clause F.2.4.1 specifies an extension to the 3GPP FLUTE FDT schema that includes a new **File/Signature** extension element which allows a receiver to verify the authenticity of fields in the metadata that accompanies the multicast transport object, including the content digest (see clause F.2.3) to verify both the integrity and authenticity of the multicast transport object itself.

The cryptographic signature carried in this element may protect any attribute or child element of the **File** instance, and shall at minimum protect the following attributes of the FDT instance:

- The **Signature/**`@scope` attribute as specified in clause F2.4.1 (in place of the HTTP `Signature-Input` field).

- The **File/**`@Content-Digest` attribute, as specified in clause F.2.3.1, carrying a cryptographically secure hash of the multicast transport object.

- If present, the **File/**`@Repr-Digest` attribute, as specified in clause F.2.3.1, carrying a cryptographically secure hash of the media object.

The following attributes of the File instance should additionally be protected by the signature:

- The **File/**`@TOI` attribute.

- The **File/**`@Content-Location` attribute.

F.2.4.1 Signature extension element

The syntax of the **File/Signature** child element is specified in table F.2.4.1-1 below and is loosely based on the format of the HTTP `Signature` header specified in RFC 9421 [38]. The XML Schema of the **File/Signature** element is specified in clause F.2.5.1. An example of a FLUTE FDT instance containing a **File/Signature** element is provided in clause F.2.5.2.

The HTTP `Signature-Input` header specified in RFC 9421 [38] is replaced by attributes carried in the **File/Signature** child element. In addition, the implementation of the **File/Signature** element is significantly simpler than that of the HTTP `Signature` header as it does not support the use case where multiple signatures are included in a single object metadata instance, and as such there is no mapping of the `tag` signature parameter.

Table F.2.4.1-1: Signature element syntax

Element or attribute name	Use	Data type	Description
File/Signature	0..n	String	Container for a cryptographically secure signature, used to verify the authenticity of the protected multicast transport object, encoded in Base64 per RFC 4648 [9].
@scope	1	String	An ordered list of metadata components protected by the signature expressed as an ordered inner list of string values as described in section 2.3 of RFC 9421 [38], separated with a comma and optional whitespace. Each string value identifies a derived component (see clause 12.2.1.1), expressed as an XPath relative to the parent **File** element and prefixed with an "@" character, as specified in section 2.2 of [38].
@created	0..1	Integer	Creation time of the signature as a UNIX timestamp value.
@expires	0..1	Integer	Expiration time of the signature as a UNIX timestamp value.
@algorithm	1	String	The message signature algorithm from the HTTP Message Signature Algorithm Registry.
@keyUri	1	URI String	The subject key identifier of an X.509 certificate identifying the public key that shall be used to verify the signature.

The @scope attribute is equivalent to the ordered list of component identifiers for the covered components conveyed in the Signature-Input header as described in section 2.3 of [38].

Metadata components listed in the @scope attribute may be:

- Any attribute of the **File** element.
- The contents of child elements of the **File** element or explicitly named attributes of those child elements.

For example:

EXAMPLE 1: The **File**@TOI attribute appears in the @scope attribute as "*@@TOI*".

EXAMPLE 2: The **File/Cache-Control** element appears in the @scope attribute as "*@Cache-Control*".

EXAMPLE 3: A **File/Example/Element**@attribute appears in the @scope attribute as "*@Example/Element/@attribute*".

When the **Signature** element is used to verify the authenticity of a multicast transport object, the **File**@Content-Digest attribute specified in clause F.2.3.1 shall be present, and it shall be protected by the signature by listing the value "*@@Content-Digest*" in the @scope attribute.

The signature base is as specified in section 2.5 of [38] and shall be carried as the value of the **Signature** element. For each message component, the component identifier is the XPath of the attribute or element, and the parameter is the content contained within the XML element or the value of the XML attribute. The order of these message components is order-sensitive, so the signature base shall be created in the same order as the list of component identifiers carried in the @scope attribute.

Because each instance of the **Signature** element contains only a single signature, there is no need for the structured fields dictionary syntax specified in [38]. Instead, the parameters of a given signature are provided as attributes on the **Signature** element itself. The @created, @expires, @alg and @keyid parameters are identical to the parameters with the same names specified in section 2.3 of [38]. There is no mapping of the key name or tag parameter for each signature in the **Signature** element.

A given FDT **File** instance may include multiple **Signature** child elements to allow for multiple signature algorithms to be provided for a given multicast transport object. In this case, each such child element shall carry a unique value of the @alg attribute.

F.2.5 Extended FLUTE File Delivery Table

F.2.5.1 Extended FLUTE FDT schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns="urn:dvb:metadata:ExtendedFileDeliveryTable:2022"
    xmlns:mbms2022="urn:3GPP:metadata:2022:FLUTE:FDT"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:dvb:metadata:ExtendedFileDeliveryTable:2022"
    elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="urn:3GPP:metadata:2022:FLUTE:FDT" schemaLocation="FLUTE-FDT-3GPP-Main-
2022.xsd"/>
    <xs:complexType name="SignatureType">
        <xs:annotation>
            <xs:documentation>Definition of the File/Signature type for message
authenticity.</xs:documentation>
        </xs:annotation>
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute name="scope" type="xs:string" use="required">
                    <xs:annotation>
                        <xs:documentation>An ordered list of metadata components protected by the
signature, expressed as an ordered list of XPath values.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="created" type="xs:positiveInteger" use="optional">
                    <xs:annotation>
                        <xs:documentation>Creation time of the signature as a UNIX timestamp
value.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="expires" type="xs:positiveInteger" use="optional">
                    <xs:annotation>
                        <xs:documentation>Expiration time of the signature as a UNIX timestamp
value.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="algorithm" type="xs:NMTOKEN" use="required">
                    <xs:annotation>
                        <xs:documentation>Message signature algorithm from the HTTP Message
Signature Algorithm Registry expressed as a fully-qualified URI.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="keyId" type="xs:hexBinary" use="required">
                    <xs:annotation>
                        <xs:documentation>The subject key identifier of an X.509 certificate
identifying the public key that shall be used to verify the signature.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

    <xs:simpleType name="MessageDigestType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[a-z][-_.*a-z0-9]*=:[a-zA-Z0-9+/=]*:"/>
        </xs:restriction>
    </xs:simpleType>

    <xs:complexType name="FileType">
        <xs:complexContent>
            <xs:extension base="mbms2022:FileType">
                <xs:sequence>
                    <xs:element name="Signature" type="SignatureType" minOccurs="0"
maxOccurs="unbounded"/>
                    <xs:element name="delimiter" type="mbms2022:DelimiterType"/>
                    <xs:any namespace="##other" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
                </xs:sequence>
                <xs:attribute name="Content-Digest" type="MessageDigestType" use="optional">
                    <xs:annotation>
                        <xs:documentation>A message digest of the FLUTE payload
contents.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:attribute name="Repr-Digest" type="MessageDigestType" use="optional">
```

```
                    <xs:annotation>
                        <xs:documentation>A message digest of the whole object that the FLUTE
payload contents are a whole or part of.</xs:documentation>
                    </xs:annotation>
                </xs:attribute>
                <xs:anyAttribute processContents="skip"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

F.2.5.2 Extended FLUTE FDT instance document example (informative)

The example XML instance document below illustrates the use of the extended schema in clause F.2.5.1 for the purposes of signalling the integrity and asserting the authenticity of a multicast transport object using metadata in a FLUTE File Delivery Table instance.

```
<?xml version="1.0" encoding="UTF-8"?>
<FDT-Instance
    xmlns="urn:IETF:metadata:2022:FLUTE:FDT"    xmlns="urn:3GPP:metadata:2022:FLUTE:FDT"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:mabr2022="urn:dvb:metadata:ExtendedFileDeliveryTable:2022"
    xsi:schemaLocation="urn:dvb:metadata:ExtendedFileDeliveryTable:2022 extended-file-delivery-
table_2022.xsd"
    Complete="true" Content-Encoding="gzip" Expires="331129600">

    <File xsi:type="mabr2022:FileType" Content-Type="video/mp4" Content-Length="16157"
TOI="12345"
        Content-Location="tag:mcast-srv.isp.net,2022-12-
12:mts101,3922:/prd1/adp1/1080p/12345.m4s?isLast#1348792"
        mabr2022:Content-Digest="sha-256=:EinLl9N+lsIVghfGitvsfyWtNu3ADIumP9S7mO3BWIs=:"
        mabr2022:Repr-Digest="sha-256=:JOA0j36rphFxXS7oCvZi1DZEJGvbEbzmcHHTGUsA90I=:">
        <delimiter>1</delimiter>
        <mabr2022:Signature created="1670855500" algorithm="rsa-pss-sha512"
            keyId="888ABA7B0ADF37D1F97492877A68D5CE527B9FD2"
            scope="@@Content-Digest, @@TOI, @@Content-Location">
            llhL/2zn+wcTE8QRyMO8YVWW+loDhCH1xQwVZ86r4kA=
        </mabr2022:Signature>
        <mabr2022:delimiter>1</mabr2022:delimiter>
    </File>
    <schemaVersion>1</schemaVersion>
    <delimiter>1</delimiter>
</FDT-Instance>
```

F.3 Multicast gateway operation

F.3.1 Forward Error Correction

Multicast gateway implementations support the reception of all encoding symbols as required by clause L.4.7 of ETSI TS 126 346 [17]. However, a *Multicast gateway* may ignore the repair symbols.

If Forward Error Correction is supported by the *Multicast gateway* and provided as part of a multicast transport session, then the *Multicast gateway* should consider already received repair symbols as defined in clause 9.3.3 of [17] as part of the unicast repair procedure.

F.3.2 Unicast repair

The *Multicast gateway* shall follow the unicast repair procedure specified in clause 9. It shall ignore the **`Alternate-Content-Location-1`** and **`Alternate-Content-Location-2`** elements in the FDT Instance.

The start time of the unicast repair procedure is at latest the expiry time of the FDT Instance as defined in clause 9.3.2 of ETSI TS 126 346 [17] or the reception of a FLUTE packet with the Close Object (B) flag set.

In the case of resource transmission mode, the range of bytes to be requested is calculated as specified in clause 9.3.6.2 of ETSI TS 126 346 [17].

In the case of chunked transmission mode, the *Multicast gateway* shall add the chunk offset value (as specified in clause F.3.1 above) to the identified byte ranges. The byte range is determined according to clause 9.3.6.2 of ETSI TS 126 346 [17]. For example, when a packet from a chunk with chunk offset 5321 is missing, the start of the byte range is 5321 + ESI × Symbol Size of the missing packet.

Annex G (informative): Implementation guidelines for FLUTE-based multicast media transport protocol

G.1 Multicast system implementation guidelines

G.1.0 Overview

This clause describes implementation guidelines for mapping DVB-DASH content onto a FLUTE-based reference point **M** realization. Other presentation and segment formats are possible but are beyond the scope of this clause.

The *Multicast server* supports push and pull content ingest methods and this clause describes the mapping of ingest objects to multicast transport objects using FLUTE at reference point **M**, the reception procedure by the *Multicast gateway*, and the mapping of multicast transport objects into playback delivery objects for delivery at reference point **L**.

G.1.1 Regular latency operation

When serving DASH content with regular latency, resource transmission mode (see clauses 8.3.4.1 and F.2.1) can be used and media objects can be ingested into the *Multicast server* using either the pull or push content ingest methods. Figure G.1.1-1 depicts pull-based ingest of segments and figure G.1.1-2 depicts push-based ingest. As can be seen, operation downstream of the *Multicast server* is identical in both cases.

In the case of pull-based ingest, either the content is not formatted for low latency (i.e. **`SegmentTemplate/`**`@availabilityTimeOffset` is absent from the DASH MPD presentation manifest, segment is not subdivided into smaller chunks, etc.) or the *Content ingest* subfunction of the *Multicast server* ignores the `@availabilityTimeOffset` value. Regardless of the reason, in both cases the *Content ingest* subfunction of the *Multicast server* fetches each DASH segment only once it is made fully available by the *Content hosting* function.

In the case of push-based ingest, either the content of a segment is not made available as CMAF chunks before the full segment is created, or the content is not provided to the *Multicast server* using HTTP chunked transfer coding. the *Content preparation* function pushes the segment to the *Content ingest* subfunction only once it is fully available. (HTTP chunked transfer coding can still be used by the *Content preparation* function in these cases, but the HTTP chunks are generated only after the full segment is available.) Alternatively, the *Multicast server* may simply wait until a full segment has been ingested before processing it further.

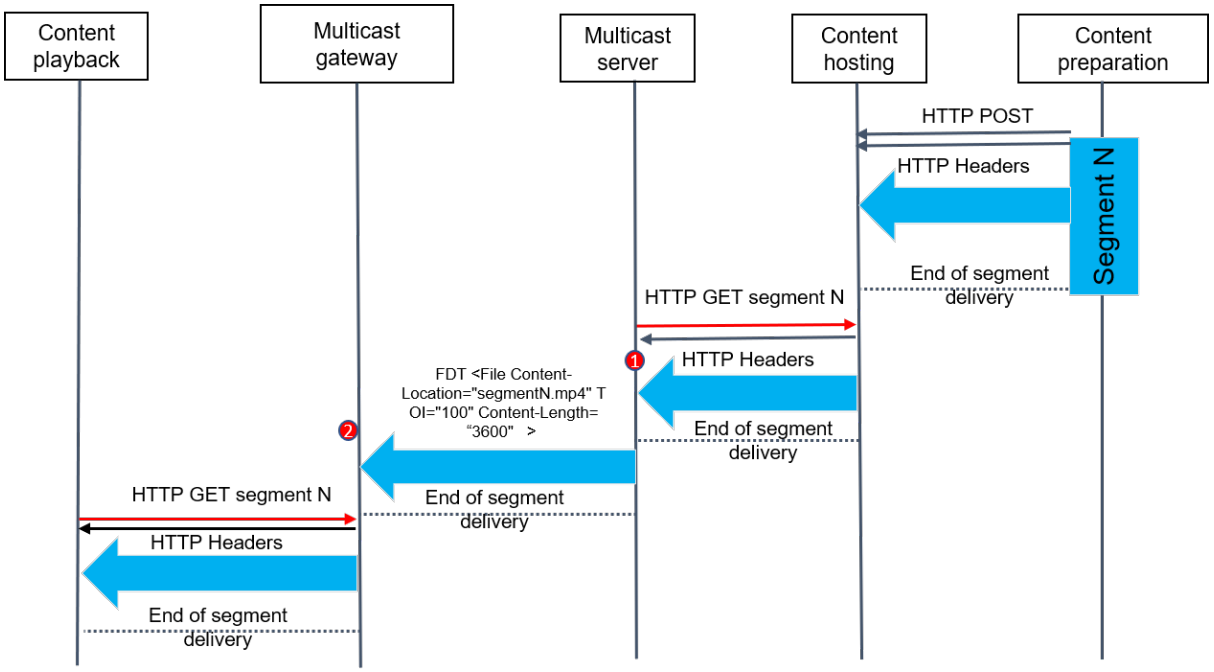


Figure G.1.1-1: Resource transmission mode with pull-based ingest

Figure G.1.1-1 depicts pull-based ingest operations. The *Content preparation* function uploads the segment to the *Content hosting* function. The *Multicast server* requests a DASH segment according to its Segment Availability Start Time (SAST), e.g. as described by clause 5.3.9.1 of MPEG-DASH [i.2]. At mark #1, the *Multicast server* issues the HTTP `GET` request and starts downloading the segment. The segment size is provided at the beginning of the download by the *Content hosting* function as the `Content-Length` HTTP response header and so the *Multicast server* can potentially commence multicast transmission (i.e. construct the FLUTE FDT and start transmitting the multicast transport object at reference point **M**) immediately after the HTTP response headers are received.

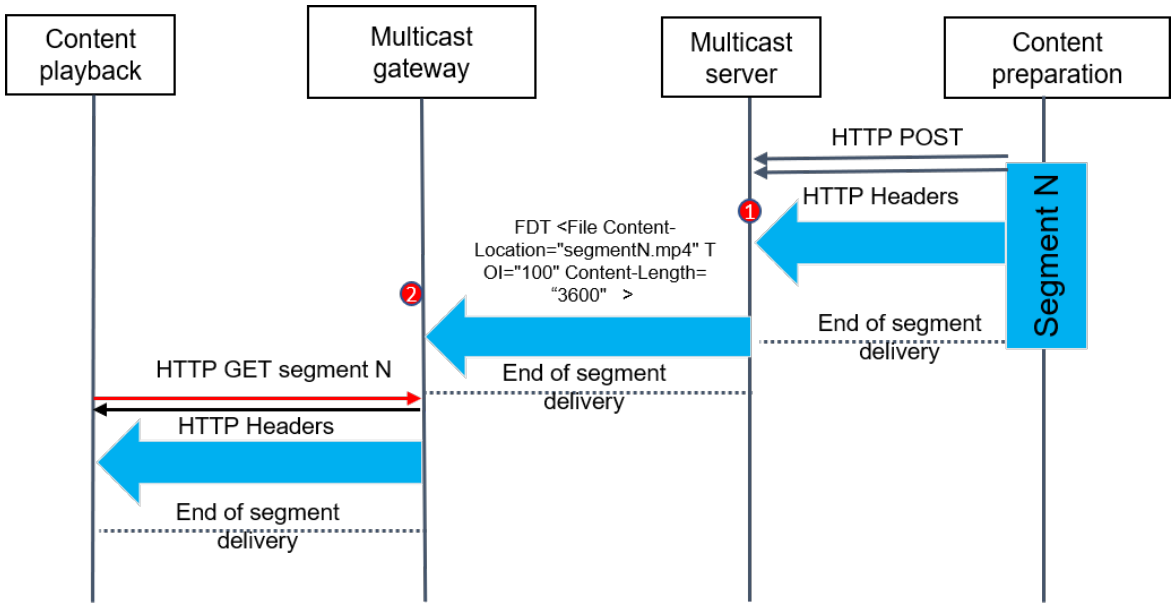


Figure G.1.1-2: Resource transmission mode with push-based ingest

Figure G.1.1-2 depicts push-based ingest operations. The *Content preparation* function pushes ingest objects according to the Segment Availability Start time (SAST) of the DASH segment. The segment is contained in the body of the HTTP request message. (HTTP `POST` is depicted in figure G.1.2-2 as an example.) The segment size is provided at the beginning of the upload by the *Content preparation* function as the `Content-Length` HTTP request header and so the *Multicast server* can potentially commence multicast transmission immediately after the HTTP headers are received.

For both push- and pull-based ingest, the *Multicast server* can use the same forwarding procedure for handling the segment in the HTTP message body (response body in case of pull and request body in case of push).

In both pull- and push-based ingest, the available network bandwidth between the *Content preparation* function and the *Multicast server* determines the transfer latency of ingest objects. When it is guaranteed that the ingest bit rate is always higher or equal to the multicast transport session bit rate (i.e. QoS-provisioned ingest), the *Multicast server* can start the multicast transmission process immediately after the HTTP headers are received. Otherwise it is recommended that the *Multicast server* buffers the ingest objects for a configurable duration.

For starting the multicast transmission process, the *Multicast server* creates the FLUTE FDT Instance for the segment, for example at time of reception of the HTTP headers (see mark #1 in figure G.1.1-1 and figure G.1.1-2). The HTTP headers include the `Content-Length` and the `Content-Type` of the ingest object, which are essential parameters in the FLUTE FDT Instance. The multicast transport object URI carried by the **`File/`**`@Content-Location` attribute is derived from the request URL.

NOTE: The *Multicast server* may rewrite the content ingest URL, as specified in clause 8.3.3.

It is recommended that the FLUTE FDT Instance, containing the metadata for the new multicast transport object, is sent on the FLUTE Session immediately before the multicast transport object it describes. The FLUTE FDT Instance can be repeated during the transmission of the multicast transport object.

Once the *Multicast gateway* receives an FDT Instance describing a new DASH segment (mark #2 in figure G.1.1-1 and figure G.1.1-2), the *Multicast gateway* can start offering the corresponding playback delivery object at reference point **L**. The *Multicast gateway* may delay offering the playback delivery object at reference point **L** for various reasons, for example to allow extra time for unicast repair operations. The *Multicast gateway* may delay offering the playback delivery object at reference point **L** until the multicast transport object has been fully received and (if necessary) repaired using AL-FEC and/or the unicast repair procedure.

G.1.2 Low-latency operation

When serving low-latency DASH content chunked transmission mode is used (see clauses 8.3.4.1 and F.2.2). The content is offered as low-latency content according to [10] (i.e. **`SegmentTemplate/`**`@availabilityTimeOffset` is present in the DASH MPD and segments are formatted according to a low-latency segment profile). In the following, the *Multicast server* is assumed to be requesting new DASH segments of a representation according to the DASH-IF Live Media Ingest Protocol [i.4].

The *Multicast server* receives the content as a sequence of HTTP chunks. Each HTTP chunk starts with a ***chunk-size*** field.

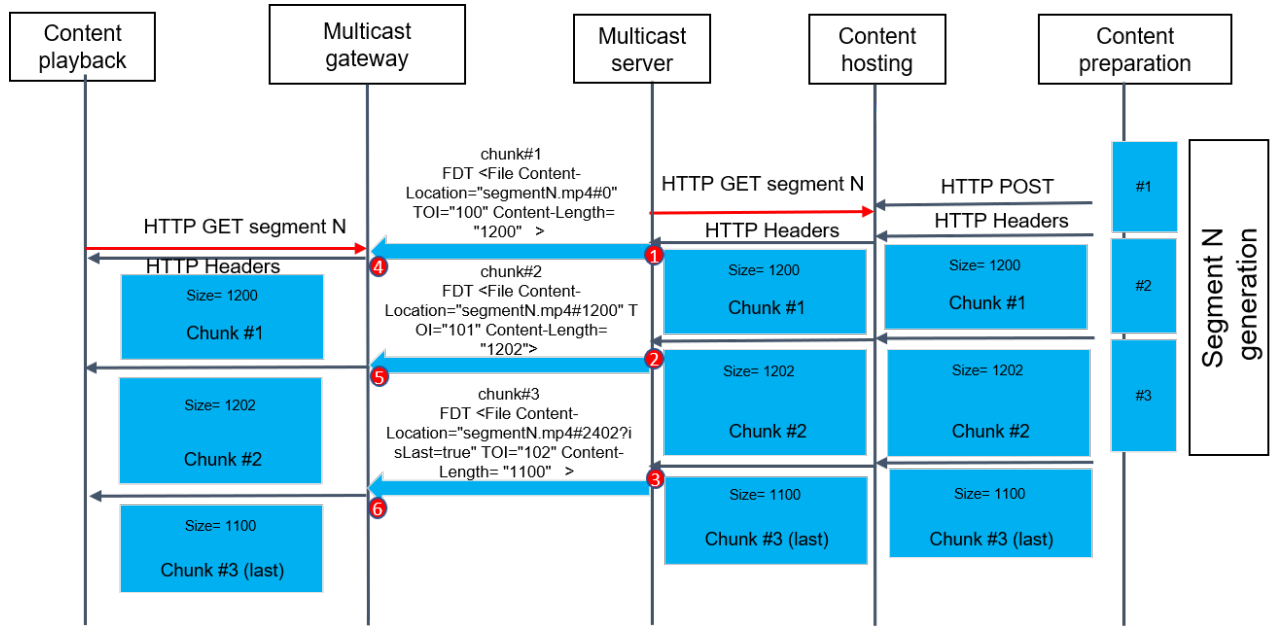


Figure G.1.2-1: Chunked transmission mode with pull ingest

Figure G.1.2-1 depicts pull-based ingest operations. The *Content preparation* function uploads the DASH segment to the *Content hosting* function as sequence of HTTP chunks. The *Multicast server* requests a segment according to its Segment Availability Start Time (SAST). At mark #1, the *Multicast server* issues the HTTP `GET` request and starts downloading the segment. In case of HTTP chunked transfer coding, the HTTP headers do not include a `Content-Length` header. The chunk size is provided at the beginning of each chunk. To maintain low-latency operation the *Multicast server* should commence multicast transmission (i.e. construct the FLUTE FDT Instance and start transmitting the multicast transport object at reference point **M**) immediately after receiving the *chunk-size* field (mark #1 in figure G.1.2-1).

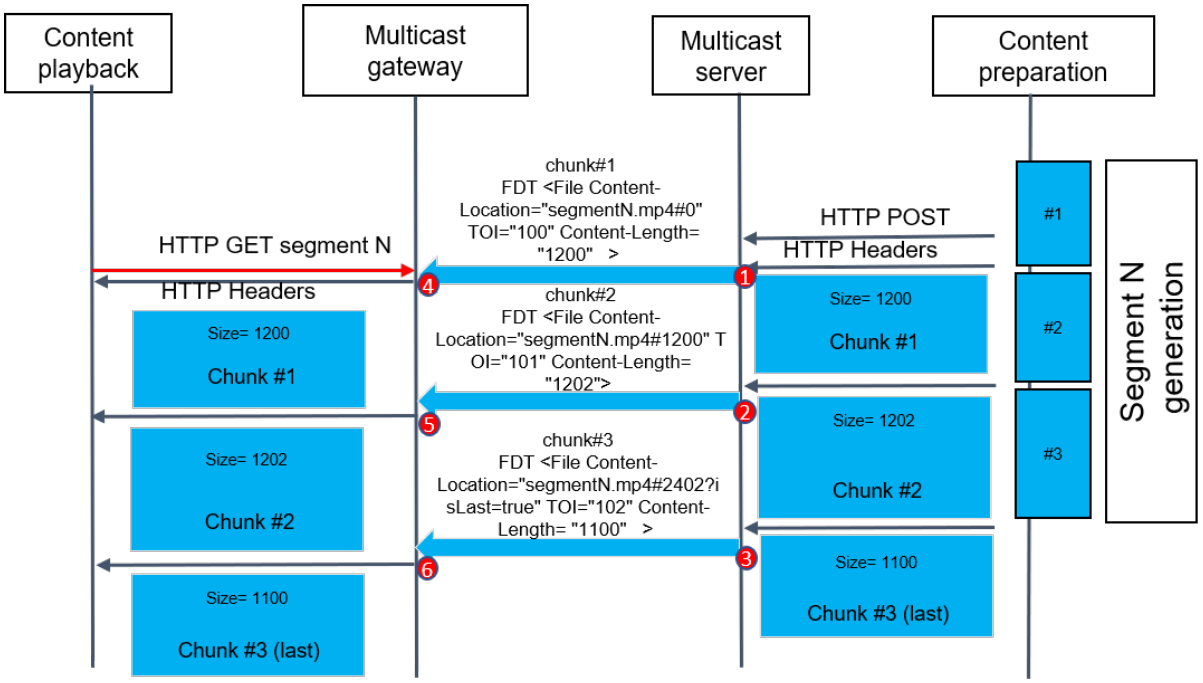


Figure G.1.2-2: Chunked Transmission Mode with push ingest

Figure G.1.2-2 depicts push-based ingest operations. The *Content preparation* function pushes ingest objects according to the Segment Availability Start time (SAST) of the corresponding DASH segment. The segment is supplied as sequence of HTTP chunks in the body of the HTTP request message. (HTTP `POST` is depicted in figure G.1.2-2 as an example.) In case of HTTP chunked transfer coding, the HTTP headers do not include a `Content-Length` header. The chunk size is provided at the beginning of each chunk. To maintain low-latency operation the *Multicast server* should commence multicast transmission immediately after receiving the *chunk-size* field (mark #1 in figure G.1.2-2).

For both push- and pull-based ingest, the *Multicast server* can use the same forwarding procedure for handling the segment in the HTTP message body (response body in case of pull and request body in case of push). In order to sustain low-latency operation the ingest bit rate should always be equal to or higher than the multicast transport session bit rate.

The *Multicast server* creates the FLUTE FDT Instance based on the following information:

- **`File/`**`@Content-Location` is either the ingest object URL or a rewritten version of it. The *Multicast server* appends the HTTP chunk offset as the fragment identifier. The chunk offset is the sum of all previously ingested chunk sizes of the ingest object in question. If the ingested DASH segment has a Random Access Point marker, the *Multicast server* may append hasRandomAccessPoint URI query component, as specified in clause F.2.2. If the ingested HTTP chunk is the last of a given ingest object, the *Multicast server* may append the isLast URI query component, as specified in clause F.2.2 (mark #3 in figure G.1.2-1 and figure G.1.2-2).

- **`File/`**`@Content-Type` is taken from the ingested HTTP response header information. The *Multicast server* uses the same MIME content type for all HTTP chunks of the same HTTP resource.

- **`File/`**`@Content-Length` is the HTTP chunk size taken from the *chunk-size* field.

As soon as the *Multicast gateway* receives the FDT Instance describing the first chunk of a given media object (mark #4 in figure G.1.2-1 and figure G.1.2-2) and the first bytes of the first chunk, it makes the media segment available (even

partially) for download at reference point L while continuing to receive the remaining chunks of that media object over reference point **M**.

G.1.3 In-band carriage of presentation manifest

If a presentation manifest is delivered over reference point **M** it can be sent either in the same FDT Instance with another multicast transport object or in a separate FDT Instance.

Figure G.1.3-1 below shows the case where the presentation manifest is sent in the same FDT Instance with another multicast transport object. In this case, the FDT Instance describes two TOIs: one for the presentation manifest and the other for a DASH segment.

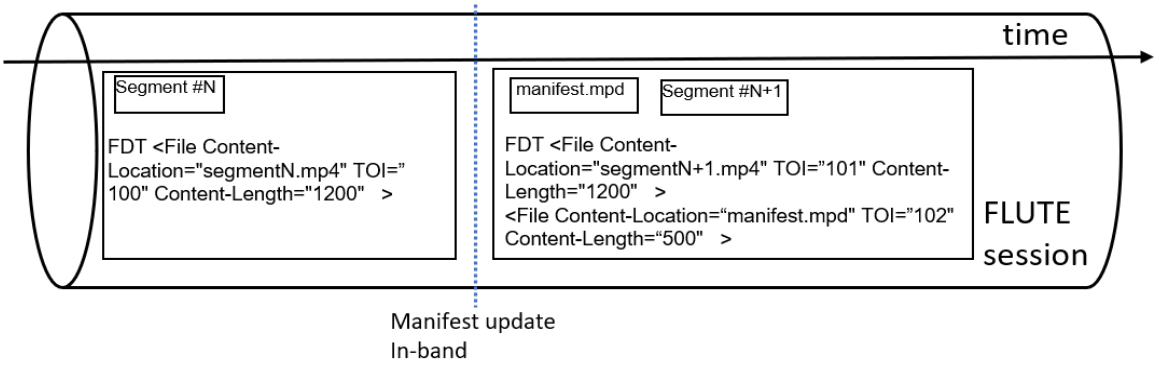


Figure G.1.3-1: Manifest update in the same FDT Instance with media segment

Figure G.1.3-2 below shows the case where the presentation manifest is sent in a different FDT Instance.

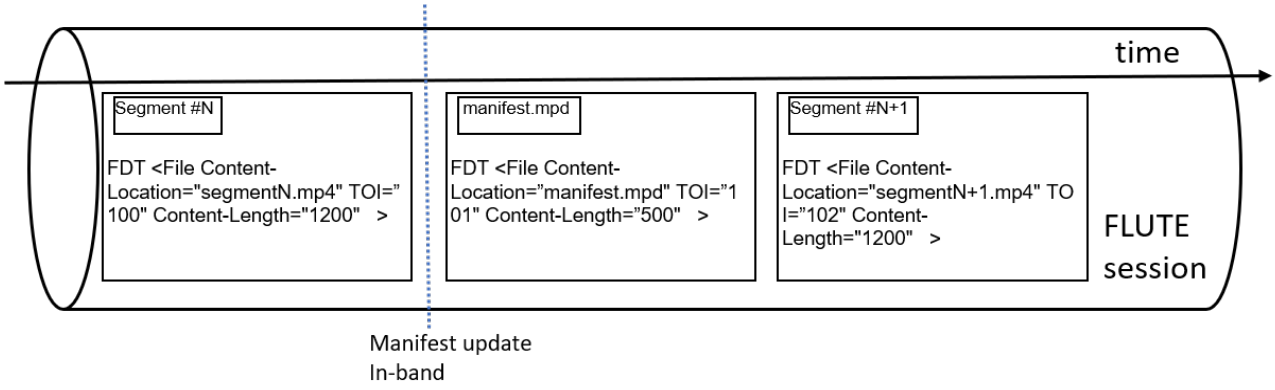


Figure G.1.3-2: Manifest update in a separate FDT Instance

G.2 Example FLUTE Session configurations

G.2.0 Introduction

This clause provides example FLUTE Session configurations that illustrate the mapping between multicast session configuration and the FLUTE Sessions. The *Multicast server* is configured to either map each service component to an individual multicast transport session (FLUTE Session), or to multiplex a set of service components (e.g. DASH representations from different adaptation sets) onto the same multicast transport session. The latter is beneficial to reduce the number of concurrent IP multicast streams in the system.

G.2.1 Multicast configuration channel over FLUTE Session

Figure G.2.1-1 depicts the in-band configuration method whereby the *Multicast server* carousels the current multicast gateway configuration instance document via a configured multicast gateway configuration transport session at reference point **M** (see clause 8.3.5) based on the multicast session configuration instance data model specified in clause 10.2. The multicast gateway configuration transport session is realized using FLUTE.

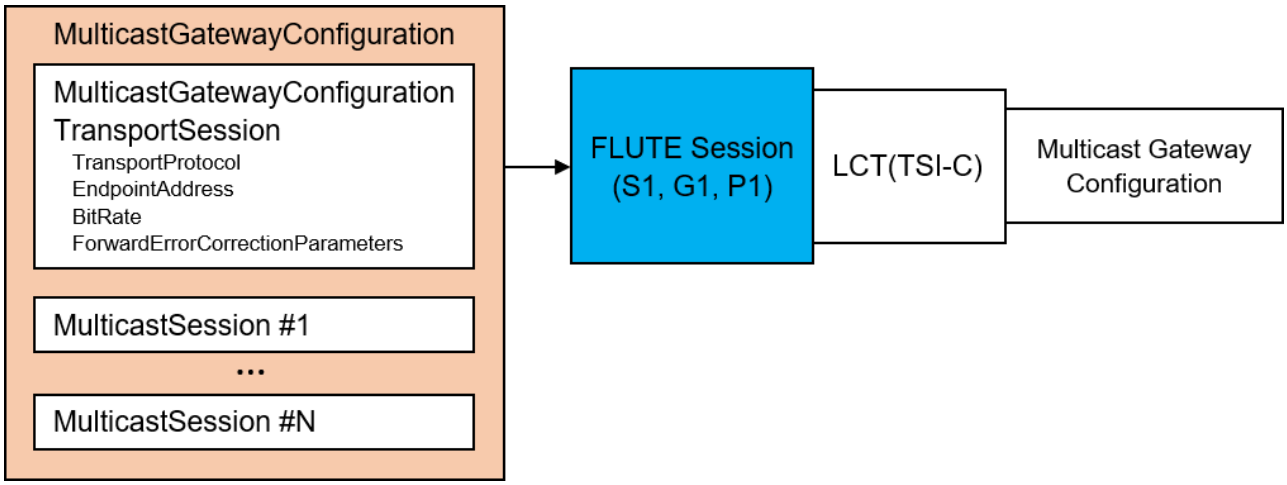


Figure G.2.1-1: Multicast configuration channel over FLUTE

G.2.2 Audio and video service components multiplexed into a single FLUTE Session

Figure G.2.2-1 depicts an example where audio and video service components are delivered over a single multicast transport session, which is realized as a single FLUTE Session. In this case, the URLs of the segments are used to separate the different service components (e.g. audio, video).

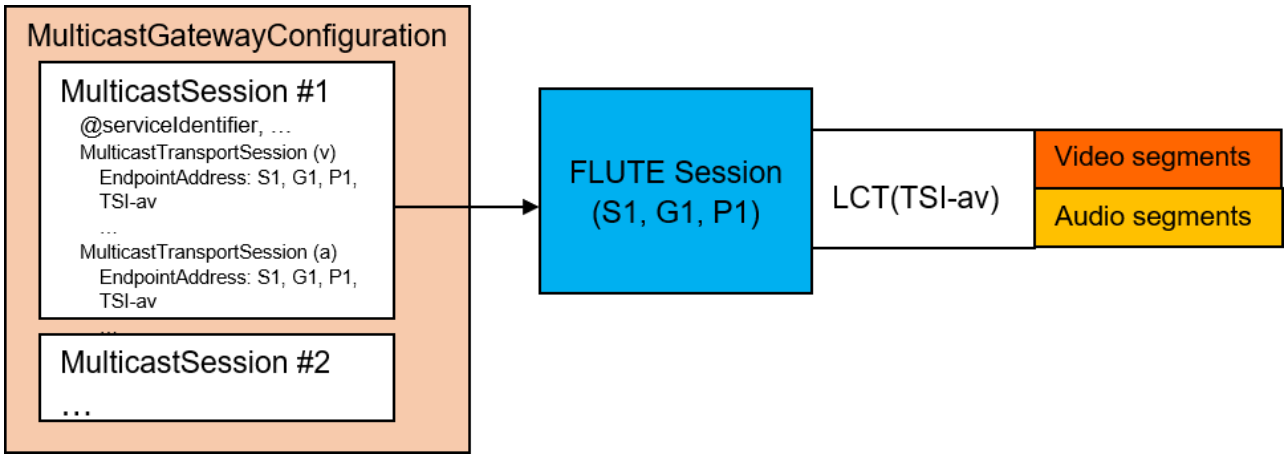


Figure G.2.2-1: Audio and video service components multiplexed into a single FLUTE Session

G.2.3 Audio and video service components carried in separate FLUTE Sessions using the same multicast group

Figure G.2.3-1 depicts an example where audio and video service components are delivered in separate multicast transport sessions, each realized as a single FLUTE Session. Here, the two multicast transport sessions are carried over the same IP multicast group G1. The two multicast transport sessions are separated based on the UDP destination port (P1 and P2).

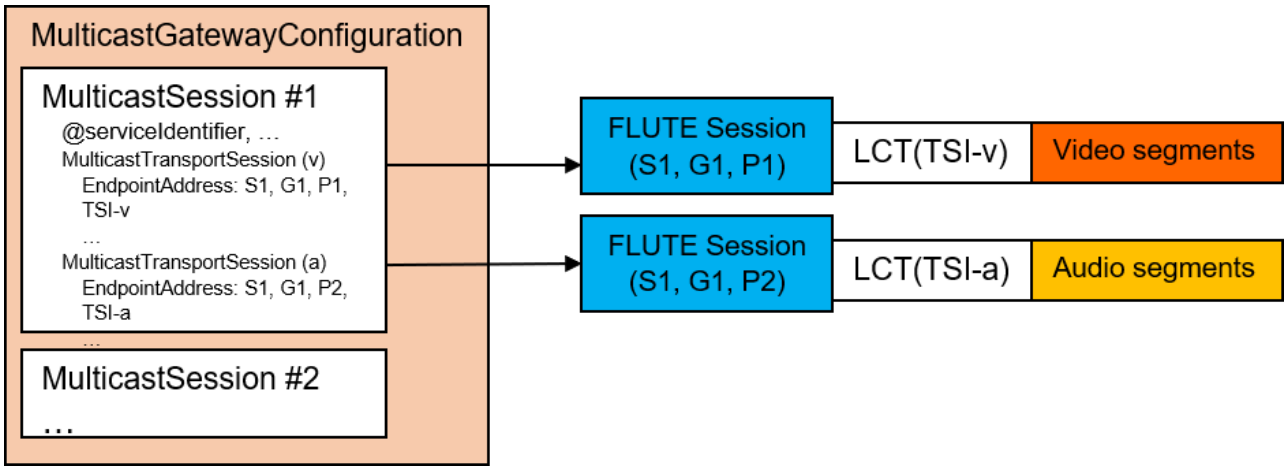


Figure G.2.3-1: Audio and video service components carried in separate FLUTE Sessions

An alternative is to use different IP multicast groups for the different multicast transport sessions, e.g. G1 and G2, so that the service components are separated even on transport level.

The manifest update can be delivered in FLUTE Session (S1, G1, P1) or FLUTE Session (S1, G1, P2) or both.

G.2.4 Audio and multiple video service components carried in separate FLUTE Sessions using independent multicast group

Figure G.2.4-1 depicts an example where audio and video service components are delivered in separate FLUTE Sessions and each video component is delivered in different FLUTE Sessions. Different IP multicast groups are used for each service component.

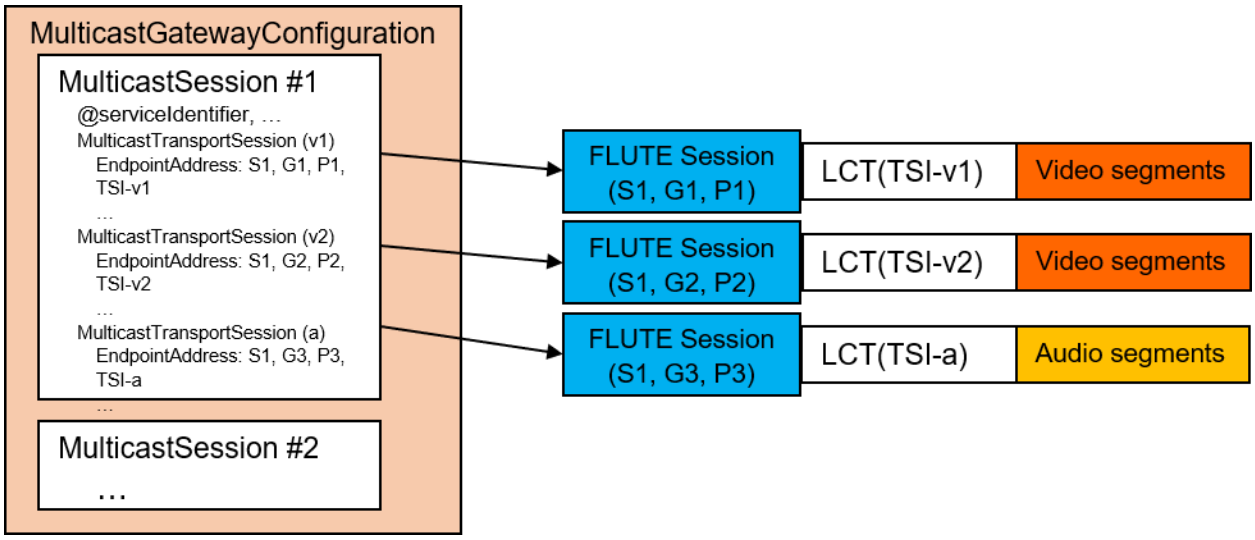


Figure G.2.4-1: Audio and multiple video service components carried in separate FLUTE Sessions

An alternative realization can use the same IP multicast groups for the multicast transport sessions of the service components.

The benefit of using different IP multicast groups for the multicast transport sessions is that only the needed multicast traffic is reaching the *Multicast gateway*, i.e. the *Multicast gateway* would join only one of the two multicast transport sessions, carrying the video service component. A potential drawback is the increase number of simultaneous multicast groups in the system which can create some routing overhead.

G.2.5 FEC data carried in FLUTE Sessions

Figure G.2.5-1 depicts an example where FEC is used for a given linear service. FEC packets separately protect audio and video service components in different FLUTE Sessions.

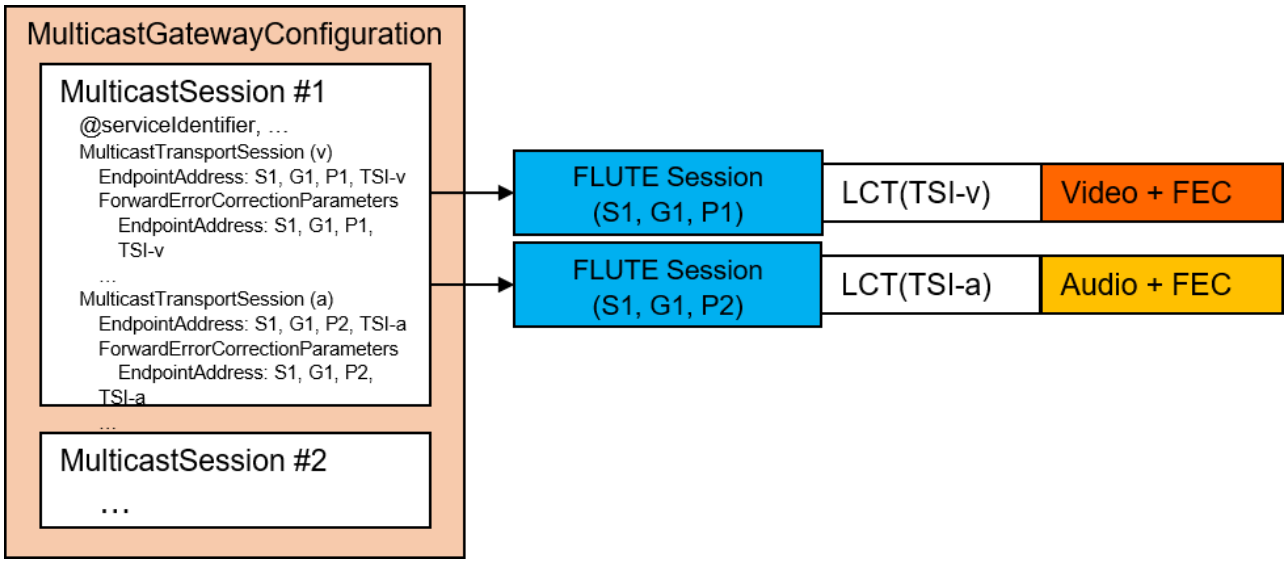


Figure G.2.5-1: Usage of FEC data to protect audio and video service components separately

Annex H (normative): ROUTE-based multicast media transport protocol

H.0 Introduction

This annex specifies a multicast media transport protocol based on ROUTE as defined in annex A of ATSC A/331 [18] with focus on linear live media streaming applications. In particular, the profile of ROUTE specified here is most beneficial in combination with live DASH and especially with low-latency live DASH as it provides similar functionalities as HTTP chunked transfer coding [1] when conveying CMAF chunks [i.10].

The ROUTE profile specified in this annex differs from ATSC A/331 [18] in the following aspects:

- Only a subset of Codepoints specified in table A.3.6 of [18] is used, as specified in clause H.4 below.
- The use of MPD-less start-up mode is prohibited by this profile (see clause H.5.1).
- The **`Alternate-Content-Location-1`** and **`Alternate-Content-Location-2`** Extended FDT elements specified in clause A.3.3.2 of [18] are ignored by the *Multicast gateway* (see clause H.6.2).
- An exception to the basic delivery object recovery operation is specified in clause H.8.0.

H.1 Signalling in the multicast session configuration

The use of the multicast media transport protocol specified in this annex shall be signalled in the multicast session configuration as follows (see clause 10.2.3.8):

TransportProtocol/**@protocolIdentifier**	TransportProtocol/**@protocolVersion**
urn:dvb:metadata:cs:MulticastTransportProtocolCS:2019:ROUTE	*1*

In Entity Mode only the integrity of a multicast transport object may be protected as specified in clause H.3.2. If every multicast transport object in a multicast transport session is so protected, the transport security mode "Integrity" (see clause 10.2.3.6) shall be indicated in the multicast session configuration.

In Entity Mode only the authenticity of a multicast transport object may be asserted by conveying a message signature as specified in clause H.3.2 in addition to an object digest. If every multicast transport object in a multicast transport session is so protected, the transport security mode *integrityAndAuthenticity* (see clause 10.2.3.6) shall be indicated in the multicast session configuration.

If neither integrity nor authenticity is protected, the transport security mode "None" shall be indicated in the multicast session configuration.

The Transport Session Identifier of the underlying LCT channel shall be conveyed in the **`EndpointAddress/ MediaTransportSessionIdentifier`** element (see clause 10.2.3.9).

Omitting the **`ForwardErrorCorrectionParameters`** element shall imply that no ROUTE Repair Flows are protecting the multicast transport session in question.

H.2 ROUTE packet format profile

H.2.0 General

When using the multicast media transport protocol specified in this annex, multicast transport objects shall be realised as ROUTE delivery objects conveyed in ROUTE Source Flows and optionally protected by ROUTE Repair Flows. The packet format used by ROUTE Source Flows and Repair Flows (specified respectively in clauses A.3 and A.4 of [18]) follows the ALC packet format specified in IETF RFC 5775 [21], with further details specified in clause A.3.5 of ATSC A/331 [18], i.e. the UDP header followed by the default LCT header and the source FEC Payload ID followed by the packet payload. The overall ROUTE packet format is as depicted in figure H.2.0-1 below.

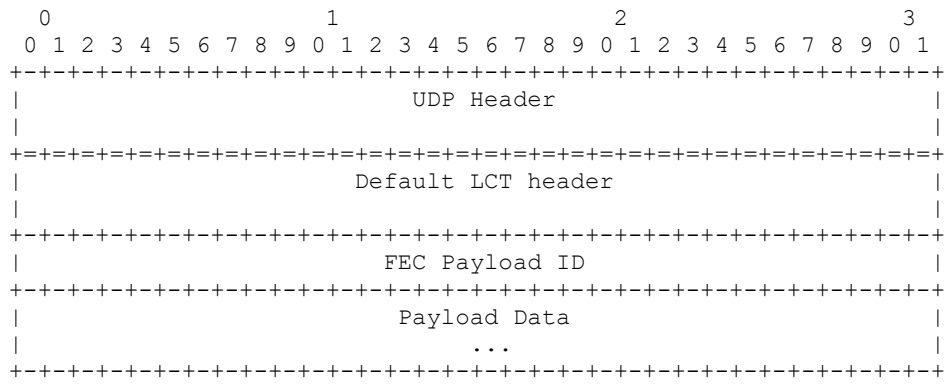


Figure H.2.0-1: Overall ROUTE packet format, from clause A.3.5 of ATSC A/331 [18]

The Default LCT header field is as defined in IETF RFC 5651 [19]. The usage of the LCT building block for ROUTE is specified in clause A.3.6 of ATSC A/331 [18].

In order to support the transmission of CMAF chunked content [i.10] the default LCT header is further constrained here as follows:

Table H.2.0-1: Usage of LCT header fields for ROUTE Source and Repair Flows

LCT header field	ROUTE Source Flows		ROUTE Repair Flows
	DASH Segments	DASH Segments comprising CMAF chunks	
Close Object (B) flag in Default LCT header	Set according to clause 7.1.7 of ATSC A/331 [18].	Set if and only if the ROUTE packet contains the last byte of the multicast transport object for both Source Flows and Repair Flows.	Set according to IETF RFC 5651 [19].
Congestion Control Information (CCI) field in Default LCT header	Should convey the earliest presentation time contained in the ROUTE packet. The length of the field may be 32 bits (C=0) or 64 bits (C=1). For more details refer to clause I.1.4.		Not applicable.
FEC Payload ID field	Conveys the *start_offset* of the LCT Payload Data field in the delivery object, as specified in clause A.3.5 (Packet Format) of ATSC A/331 [18].	Conveys the *start_offset* of the LCT Payload Data field in the delivery object, as specified in clause A.3.5 (Packet Format) of ATSC A/331 [18], with further guideline in clause I.1.4.	Not applicable.

H.2.1 LCT header extensions

The following LCT header extensions are used in this profile. All other header extensions shall be silently ignored by the *Multicast gateway*.

The length in bytes of the multicast transport object shall be signalled using *EXT_TOL* as specified in clause A.4.2.6.1 of ATSC A/331 [18] with 24 bits or, if required, 48 bits of Transfer Length. The frequency of using the *EXT_TOL* header extension is determined by channel conditions that may cause the loss of the packet carrying Close Object (B) flag, as specified in clause H.2.0 above.

NOTE: The transport object length can also be determined without the use of *EXT_TOL* by examining the LCT packet with the Close Object (B) flag. However, if this packet is lost, then the *EXT_TOL* information can be used by the receiver to determine the transport object length.

The Sender Current Time shall be signalled using *EXT_TIME* as specified in clause 8.1.1 of ATSC A/331 [18].

H.3 Delivery object mode

H.3.0 General

The delivery object mode (as indicated by the value of Delivery Object Format ID signalled in **Payload/**`@formatId` field in Session metadata for Source Flows per clause H.5.1), shall either be set to 1 or 2, where 1 refers to the File Mode and 2 refers to the Entity Mode.

H.3.1 File Mode

If the delivery object mode is set to 1 (File Mode), the following applies:

- File/object metadata carried in the Extended FDT shall be embedded within the session and delivery object signalling metadata, as specified in clause H.5.1 below.
- Certain parameters of the Extended FDT which may vary from one delivery object to another, such as the **File/**`@Content-Location` and **File/**`@Content-Length` attributes, are not permitted to appear in the session and delivery object signalling metadata. Instead, the **File/**`@Content-Location` is derived locally at the ROUTE receiver using a file template (**FDT-Instance/**`@fileTemplate` attribute of the session and object signalling) in the form of the *TOI* substitution parameter, and the length of the delivery object is instead conveyed using the LCT header extension *EXT_TOL*, as specified in clause H.2.1 above.

H.3.2 Entity Mode

H.3.2.0 General

If the delivery object mode is set to 2 (Entity Mode), the following applies:

- Delivery object metadata shall be expressed in the form of fields as defined in HTTP/1.1 [1], corresponding to one or more of the representation data and metadata header fields as defined in section 8 of IETF RFC 9110 [2].
- The entity headers sent along with the delivery object provide all information about that multicast transport object.

Sending a media object in Entity Mode is achieved using one of the available transmission modes as specified in the following clauses.

H.3.2.1 Resource transmission mode

If the length of the ingest object is known to the *Multicast server* when it starts to send the multicast transport object at reference point **M**, the ROUTE Entity Mode shall be used in resource transmission mode as specified in clause 10.2.3.5, and the resulting ROUTE delivery object shall be sent without using HTTP/1.1 chunked transfer coding, i.e. the object starts with an HTTP header containing the `Content-Length` field, followed by the concatenation of CMAF chunks:

```
|HTTP Header+Length||---chunk ----||---chunk ----||---chunk ----||---chunk ----|
```

The integrity of the ROUTE delivery object may be protected by including a `Content-Digest` or `Repr-Digest` HTTP field and message digest value in the ROUTE delivery object metadata, as specified in clause 12.1. The MD5 digest algorithm shall not be used for this purpose. A digest header may be sent as a trailing field according to section 6.5 of RFC 9110 [2].

The authenticity of the ROUTE delivery object may be asserted by including a `Signature` HTTP field and message signature in the ROUTE delivery object metadata, as specified in clause 12.2. If present, this message signature shall sign a selection of header fields in the ROUTE delivery object metadata – including a secure content digest header –to verify both the integrity and authenticity of the delivery object. The message signature may protect any header field in the ROUTE delivery object metadata and shall at minimum protect a `Content-Digest` or `Repr-Digest` header carrying a cryptographically secure hash of the delivery object. The message signature should additionally protect the multicast transport object URI. A message signature field may be sent as a trailing field according to section 6.5 of RFC 9110 [2].

H.3.2.2 Chunked transmission mode

H.3.2.2.0 General

If the length of the ingest object is unknown to the *Multicast server* when starting to send the multicast transport object at reference point **M**, ROUTE Entity mode may be used in chunked transmission mode as specified in clause 10.2.3.5, in which case the HTTP/1.1 chunked transfer coding format specified in section 7.1 of IETF RFC 9112 [1] shall be used to format the corresponding ROUTE delivery object as an HTTP message with a chunked body:

```
|HTTP Header||Separator+Length[+Chunk Digest][+Chunk Signature]||---chunk ----
            ||Separator+Length[+Chunk Digest][+Chunk Signature]||---chunk ----
            ||Separator+Length[+Chunk Digest][+Chunk Signature]||---chunk ----
            ||Separator+Length[+Chunk Digest][+Chunk Signature]||---chunk ----
            ||Separator+Length=0|[|HTTP Trailers|]
```

NOTE: It is not required to send a CMAF chunk in exactly one HTTP chunk.

H.3.2.2.1 Multicast transport object integrity protection

The integrity of the entire ROUTE delivery object may be protected by including a `Repr-Digest` HTTP field carrying a digest of the media object as a trailing field, as specified in clause 12.1 and according to the syntax specified in section 6.5 of RFC 9110 [2].

In order to facilitate low-latency delivery at reference point **L**, the integrity of each chunk of the delivery object may be protected by including a `chunk-content-digest` HTTP chunk extension, as specified in clause 12.1.2. The MD5 digest algorithm shall not be used for this purpose. Where individual chunk digests are provided, implementations should also include a `Repr-Digest` HTTP field in the HTTP trailer section, as specified in clause 12.1, to ensure the reassembled media object is complete and correct when reception is completed.

H.3.2.2.2 Multicast transport object authenticity assertion

The authenticity of the entire ROUTE delivery object may be asserted by including a message signature as specified in clause 12.2 as a trailer field according to section 6.5 of RFC 9110 [2] in the ROUTE delivery object metadata. If present, this message signature shall sign a selection of header fields in the ROUTE delivery object metadata, including a secure representation digest header to verify both the integrity and authenticity of the delivery object. The message signature may protect any header field in the ROUTE delivery object metadata and shall at minimum cover the `Repr-Digest` field carrying a cryptographically secure hash of the delivery object. The message signature should additionally protect the multicast transport object URI.

In order to facilitate low-latency delivery at reference point **L**, the authenticity of each chunk of the delivery object may be protected by including a `chunk-signature` HTTP chunk extension, as specified in clause 12.2.2. Where individual chunk signatures are provided, implementations may additionally include an HTTP message signature (`Signature` field) in the HTTP trailer section, as specified in clause 12.2, in combination with a `Repr-Digest` HTTP trailing field, as specified in clause H.3.2.2.1.

H.4 Codepoint signalling

Irrespective of the delivery object mode in use, a Codepoint value exclusively from 5 through 9 shall be signalled in the Codepoint field of the LCT Header. The semantics of the values of this field are specified in table A.3.6 of [18]. To summarize:

- DASH initialization segments are signalled using Codepoint values 5, 6, or 7.
- DASH media segments are signalled using either Codepoint value 8 (indicating ROUTE File Mode) or Codepoint value 9 (indicating ROUTE Entity Mode).

H.5 In-band multicast session metadata signalling

H.5.0 General

The technical metadata describing each multicast transport session may be signalled following the principles of service level signalling specified in clause 7.1 of [18] in addition to the multicast session configuration specified in clause 10. Specifically and exclusively, Service-based Transport Session Instance Description (S-TSID) and MPD fragments may be signalled. If present, these fragments shall be conveyed to the *Multicast gateway* within the same multicast transport session they describe, in a dedicated LCT transport channel with TSI=0.

The S-TSID metadata fragment shall include the description for each Source Flow and Repair Flow within the ROUTE Session (see clause A.2.2 of [18]) described by this metadata, represented by the **SrcFlow** and **RepairFlow** elements, respectively, in the S-TSID.

The **RS** (ROUTE Session) element of the S-TSID metadata fragment shall replicate information from the **MulticastTransportSession** element (clause 10.2.3), as specified in table H.5.0-1 below.

Table H.5.0-1: Mapping of multicast transport session parameters to S-TSID

Multicast transport session parameter	S-TSID element or attribute
MulticastTransportSession/EndpointAddress/NetworkSourceAddress	**RS/**@sIpAddr
MulticastTransportSession/EndpointAddress/NetworkDestinationGroupAddress	**RS/**@dIpAddr
MulticastTransportSession/EndpointAddress/TransportDestinationPort	**RS/**@dPort
MulticastTransportSession/EndpointAddress/MediaTransportSessionIdentifier	**RS/LS/**@tsi
MulticastTransportSession/@start	**RS/LS/**@startTime
MulticastTransportSession/@start + **MulticastTransportSession/**@duration	**RS/LS/**@endTime
MulticastTransportSession/BitRate/@maximum	**RS/LS/**@bw
NOTE: It is possible for a single ROUTE Session to comprise multiple multicast transport sessions, each with the same destination group IP address and port number, but distinguished by different LCT Transport Session Identifiers (**LS/**@tsi), and hence multiple **MulticastTransportSession** elements (clause 10.2.3) may be used to signal a single ROUTE Session.	

In the case where a Repair Flow is carried in the same ROUTE Session as the Source Flow it protects, with the two flows differing only in their respective LCT Transport Session Identifiers, the multicast transport session parameters specified in table H.5.0-2 below shall additionally be mapped into a different **LS** child element of the same **RS** that describes the Source Flow protected.

Table H.5.0-2: Mapping of multicast transport session forward error correction parameters to S-TSID for Repair Flow in the same ROUTE Session

Multicast transport session parameter	S-TSID element or attribute
MulticastTransportSession/ForwardErrorCorrectionParameters/ EndpointAddress/MediaTransportSessionIdentifier	**RS/LS/**@tsi
MulticastTransportSession/ForwardErrorCorrectionParameters/ OverheadPercentage	**RS/LS/RepairFlow/ FECParameters/**@overhead

In the case where one or more Repair Flows are carried in different ROUTE Sessions from the Source Flow they protect (for example, see clause I.3.4), an additional independent **RS** element shall be present in the S-TSID metadata fragment for each such Repair Flow replicating information from the **MulticastTransportSession/ForwardErrorCorrection Parameters** element (clause 10.2.3), as specified in table H.5.0-3 below.

Table H.5.0-3: Mapping of multicast transport session forward error correction parameters to S-TSID for Repair Flow in different ROUTE Session

Multicast transport session parameter	S-TSID element/attribute
MulticastTransportSession/ForwardErrorCorrectionParameters/ EndpointAddress/NetworkSourceAddress	**RS/**@sIpAddr
MulticastTransportSession/ForwardErrorCorrectionParameters/ EndpointAddress/NetworkDestinationGroupAddress	**RS/**@dIpAddr
MulticastTransportSession/ForwardErrorCorrectionParameters/ EndpointAddress/TransportDestinationPort	**RS/**@dPort
MulticastTransportSession/ForwardErrorCorrectionParameters/ EndpointAddress/MediaTransportSessionIdentifier	**RS/LS/**@tsi
MulticastTransportSession/ForwardErrorCorrectionParameters/ OverheadPercentage	**RS/LS/RepairFlow/ FECParameters/**@overhead

H.5.1 Session metadata for Source Flows

The session metadata for each Source Flow in a ROUTE Session is conveyed by a **SrcFlow** element in the S-TSID fragment, as specified in clause A.3 of [18].

The **MediaInfo/**@startUp attribute shall be set to false.

NOTE: MPD-less start-up mode is prohibited by this profile.

H.5.2 Session metadata for Repair Flows

The session metadata for each Repair Flow in a ROUTE Session is conveyed by a **RepairFlow** element in the S-TSID fragment, as specified in clause A.4.3 of [18].

H.6 Delivery Object signalling in File mode

H.6.1 Carriage of Extended FDT

In ROUTE File Mode, delivery object signalling uses the Extended FDT. The Extended FDT may be conveyed as the **EFDT** child element of the **SrcFlow** element (see clause H.5.1 above) or as a separate delivery object. Both mechanisms are specified in clause A.3.3.2 of [18].

H.6.2 Profile of Extended FDT

The **Alternate-Content-Location-1** and **Alternate-Content-Location-2** elements specified in clause A.3.3.2 of [18] shall be ignored by the *Multicast gateway*.

H.7 Multicast server operation

The *Multicast server* shall perform all of the functions specified in clause 8.3. The *Multicast server* shall use the session signalling specified in clause H.5 and delivery object signalling specified in clause H.6 when constructing multicast transport objects at reference point **M**. The Codepoints for signalling the ROUTE delivery mode shall be used according to the values specified in clause H.4.

Both resource transmission mode and chunked transmission mode are treated the same by this profile: in either mode one ingest object shall be mapped to one multicast transport object. Guidelines for further optimizations for the transport of DASH/CMAF media objects at reference point **M** are outlined in clause I.1.

H.8 Multicast gateway operation

H.8.0 Overview

The *Multicast gateway* shall perform all of the functions specified in clause 8.4. The *Multicast gateway* shall use the in-band multicast session metadata signalling specified in clause H.5 and (in File Mode) the delivery object signalling specified in clause H.6 when receiving multicast transport objects at reference point **M**.

The multicast gateway shall ensure the timely publishing of playback delivery objects at reference point **L**, based on the media timing described in the presentation manifest. Specifically, HTTP/1.1 chunked transfer coding shall be enabled at reference point **L** if **`MPD/`**`@availabilityTimeOffset` is signalled in the DASH presentation manifest, to allow for timely sending of segment data to the *Content playback* function. This is an exception to the basic delivery object recovery operations specified in clause A.3.10.2 of [18], where it is specified that the delivery object is handed to the application only upon recovery of a complete set of its packet payloads.

Further optimized channel acquisition may be achieved by optional signalling as documented in clause I.2.

H.8.1 Forward Error Correction

The file repair procedures using AL-FEC shall follow the specification in clause A.4 of [18]. Specifically, this procedure uses the signalling in the **`S-TSID/RS/LS/RepairFlow`** element.

H.8.2 Unicast repair

The *Multicast gateway* shall follow the unicast repair procedure specified in clause 9. However, the file repair procedure specified in clause 7.1.7.2 of [18] may trigger an earlier unicast repair. This procedure includes the use of the Close Object (B) flag and Close Session (A) flag.

The *start_offset* field as specified in clause A.3.5 of [18] and the size of the ROUTE packet payload determines the byte range of the missing data to be requested via unicast. The missing byte range of data following a received packet *i* and preceding a subsequently received packet *k* after the lost packet(s) is *start_offset(i) + size(i)* to *start_offset(k) - 1* where *size(i)* is the packet payload of ROUTE packet *i*.

Annex I (informative): Implementation guidelines for ROUTE-based multicast media transport protocol

I.1 Multicast server implementation guidelines

I.1.0 Overview

This clause documents how a *Multicast server* should map DASH and CMAF media objects to ROUTE delivery objects based on the ROUTE profile specified in annex H. Specifically, this clause documents the conversion of ingest objects (both presentation manifest and DASH segments) at reference point **P_{in}'** to multicast transport objects at reference point **M**. The use of "Interface 2" of the DASH-IF Specification of Live Media Ingest [i.4] is assumed.

I.1.1 Presentation manifest signalling

The *Content ingest* subfunction of the *Multicast server* can read and possibly modify the presentation manifest to change the timing on the availability of the objects, e.g. in MPEG-DASH by changing the value of the **MPD/**`@availabilityStartTime` when required, since this can differ for the media components delivered over reference point **M**.

I.1.2 Service component mapping to object flows

Assuming the presentation manifest is a DASH MPD containing a Period with possibly multiple Adaptation Sets, and each Adaptation Set contains one or more Representations (each being a service component), then the following mapping is recommended:

- Each Representation is sent in a separate LCT session with its unique TSI.
- If *$Number$* templating is used in the MPD, it is recommended to use the File Mode.
- If *$Time$* templating or any other scheme for Segment addressing is used, then it is recommended to use the Entity Mode.
- Source Flow session metadata is set in the S-TSID fragment as follows:
  - **Srcflow/**`@rt` is set to *TRUE*.
  - **Srcflow/**`@timescale` is set to the *timescale* value of the Representation as present in the movie header.
  - **Srcflow/**`@codePoints` is set to *TRUE*.
  - **Srcflow/**`@type` is set to the media type of the Representation, including a codecs and profile parameter according to IETF RFC 6381 [22].
- Each DASH segment is sent with a new Transport Object Identifier (TOI).
- If File Mode is used, then an **FDT-Instance/**`@fileTemplate` is provided.
- If Entity Mode is used, then the Extended FDT is not present.

I.1.3 Mapping of ingest objects to ROUTE packets

I.1.3.0 General

For sending DVB DASH content [10], a full segment (that may contain CMAF chunks [i.10]) represents an ingest object at reference point **P_{in}'**.

In this description, the following is assumed:

- The *Multicast transmission* subfunction operates in ROUTE File Mode.
- `$Number$` templating is used and the *Multicast transmission* subfunction knows the number of the segment.
- The *Content ingest* subfunction may receive ingest objects from the *Content preparation* function as a sequence of CMAF chunks using HTTP chunked transfer coding.

NOTE: If the segment consists of only a single CMAF chunk, then this describes the regular Segment-based processing.

- If a CMAF chunk is provided, then the *Content preparation* function is aware whether or not this is the last chunk of the DASH segment.
- The *Content preparation* function may be made aware of a CMAF Random Access chunk in addition to the first CMAF chunk in the segment (signalled by the MPD **`RandomAccess/`**`@interval` [i.2] and the presence of the brand *cmfr* in the *styp* box of the chunk).
- The sending of one Representation is described.
- The source FEC Payload ID comprises the *start_offset* field.

The principles of the ROUTE Sender operation as defined in annex A.3.9 of ATSC A/331 [18] apply.

I.1.3.1 ROUTE packetization

The following guidelines should be followed for ROUTE packetization:

1) Each ROUTE packet sent carries contiguous bytes of data from the media object to be sent.
2) The amount of data to be sent in a single ROUTE packet is limited by the maximum transfer unit of the data packets or the size of the remaining data of the CMAF chunk being sent, whichever is smaller.
3) The *start_offset* field in the LCT header of the ROUTE packet indicates the byte offset of the carried data in the media object being sent.
4) If it is the first ROUTE packet carrying a CMAF Random Access chunk, except for the first CMAF chunk in the segment, the least significant bit of the Protocol-Specific Indication (PSI) field in the LCT header may be set to 1. The receiver may use this information for optimization of random access.
5) As soon as the total length of the media object is known, potentially with the reception of the last CMAF chunk of a segment, the *EXT_TOL* extension header may be added to the LCT header.
6) The Close Object (B) flag is set to 1 if this is the last ROUTE packet carrying the data of the media object.

Pseudo-code implementing the above guidelines is described in the following:

- T is the size of the CMAF chunk to be sent, in bytes.
- X is start offset, initially set to zero.
- R is the remaining number of bytes of the chunk to be sent, set initially to R=T.
- Y is the maximum transfer unit of the data packets. The transfer unit is determined either by knowledge of underlying transport block sizes or by other constraints

Packet generation is done as follows:

7) Set *start_offset* field in the ROUTE packet to X.

8) If it is the first packet carrying a CMAF Random Access chunk, except for the first CMAF chunk in the segment, the least significant bit of the Protocol-Specific Indication (PSI) field in the LCT header may be set to 1.

9) As soon as the transport object length of the Segment is known, potentially with the reception of the last CMAF fragment of a segment, *EXT_TOL* extensions headers may be added to the LCT header.

10) If Y < R:

    a) Set the B Flag to 0.

    b) Send byte X + 1 till X + Y of the chunk in the packet.

    c) Update X = X + Y.

    d) Repeat from step 1.

11) Otherwise (Y ≥ R):

    e) Set the B Flag to 1.

    f) Send R last bytes of the CMAF chunk in this packet.

The setting of the CCI field is discussed in clause I.1.4 below.

I.1.4 CMAF/DASH timing mapping

The ROUTE sender knows the earliest presentation time of each CMAF chunk in timescale of the `@timescale` value.

If the least significant bit of the PSI field in the LCT packet header is set to 1 as described in clause I.1.3.1 above, the LCT Congestion Control Information (CCI) field is set to the earliest presentation time of the CMAF chunk included in the ROUTE packet (see table H.2.0-1). A receiver may use this information to optimize the channel acquisition time.

I.2 Multicast gateway implementation guidelines

I.2.0 Overview

This clause documents the *Multicast gateway* operations for conversion of multicast transport objects received at reference point **M** to playback delivery objects provided at reference point **L**. In this clause, delivery of a DVB DASH MPD [10] describing DVB DASH/CMAF content is described.

I.2.1 Presentation manifest

The presentation manifest is made available to the *Multicast gateway* either via reference point **M** or via unicast at reference point **A**.

I.2.2 Fast stream acquisition

When the *Multicast gateway* initially starts reception of ROUTE packets, it is likely that the reception does not start from the very first packet carrying the data of a multicast transport object, and in this case such a partially received object is normally discarded. However, the channel acquisition or "tune-in" times can be improved if the partially received object is usable by the application.

One example realization for this is as follows:

- The *Multicast gateway* checks for the first received packet with the least significant bit of the PSI set to 1, indicating the start of a CMAF Random Access chunk.
- The *Multicast gateway* may make the partially received object (a partial DASH segment starting from the packet above) available at reference point **L**.
- It may recover the earliest presentation time of this CMAF Random Access chunk from the ROUTE packet LCT Congestion Control Information field (see table H.2.0-1) and add a new Period element in the MPD exposed at reference point **L** containing just the partially received DASH segment with period continuity signalling [i.2].

I.3 Example ROUTE Session configurations

I.3.0 Overview

This clause provides a few example ROUTE session configurations to illustrate the relation of signalling between the multicast session configuration (clause 9), the S-TSID metadata fragment (clause H.5), and the resulting ROUTE Sessions.

In the following figures, *SX*, *GX*, *PX*, and *@TSI-X* represent the source IP address **RS/**`@sIpAddr`, destination group IP address **RS/**`@dIpAddr`, destination port **RS/**`@dPort`, and LCT Transport Session Identifier **LS/**`@tsi`, respectively (see clause H.5.0).

NOTE: In all examples, the S-TSID is itself conveyed in the ROUTE Session on *TSI=0*, but this is not depicted for reasons of clarity.

I.3.1 Audio and video service components multiplexed into a single ROUTE Session

The first example in figure I.3.1-1 depicts a basic multicast transport session with a single video and a single audio service component, both carried in the same ROUTE Session.

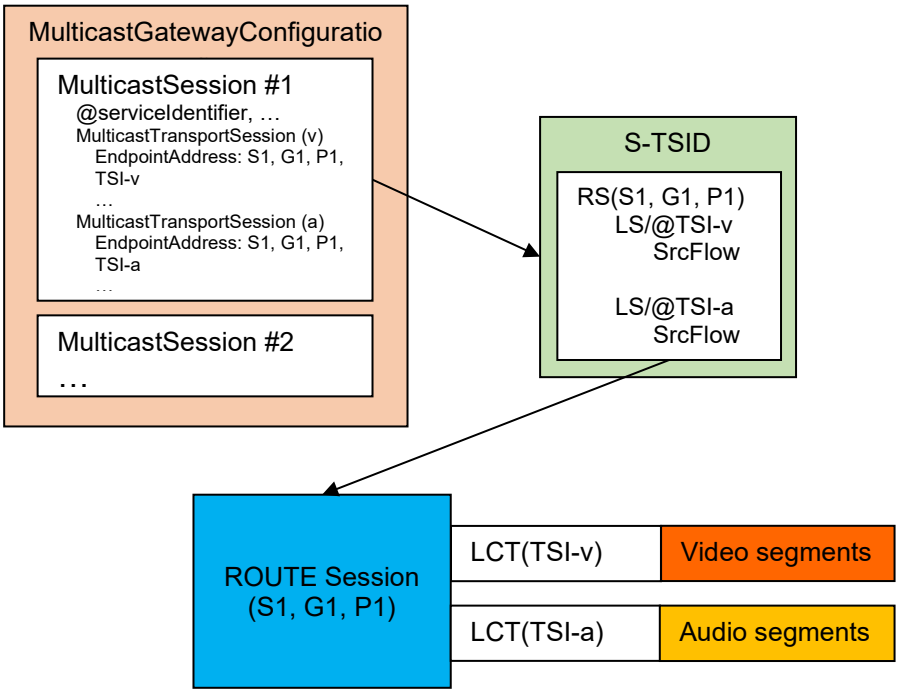


Figure I.3.1-1: Two Source Flows multiplexed on the same <S, G, P> with no Repair Flows

I.3.2 Audio and video service components carried in separate ROUTE Sessions

The second example in figure I.3.2-1 extends the first example, and in this case, the audio and the video service components are carried in their own separate ROUTE Session.

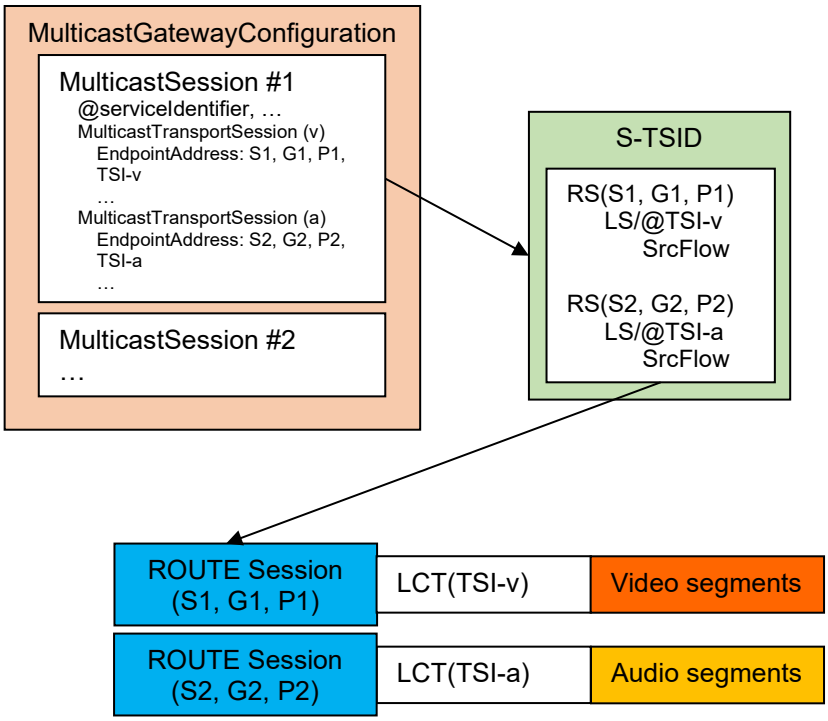


Figure I.3.2-1: Two Source Flows on different <S, G, P> with no Repair Flows

I.3.3 Audio and multiple video service components carried in separate ROUTE Sessions

The third example in figure I.3.3-1 extends the second example by depicting multiple video Source Flows with different video qualities (different video bitrates), potentially enabling adaptive multicast by the underlying network.

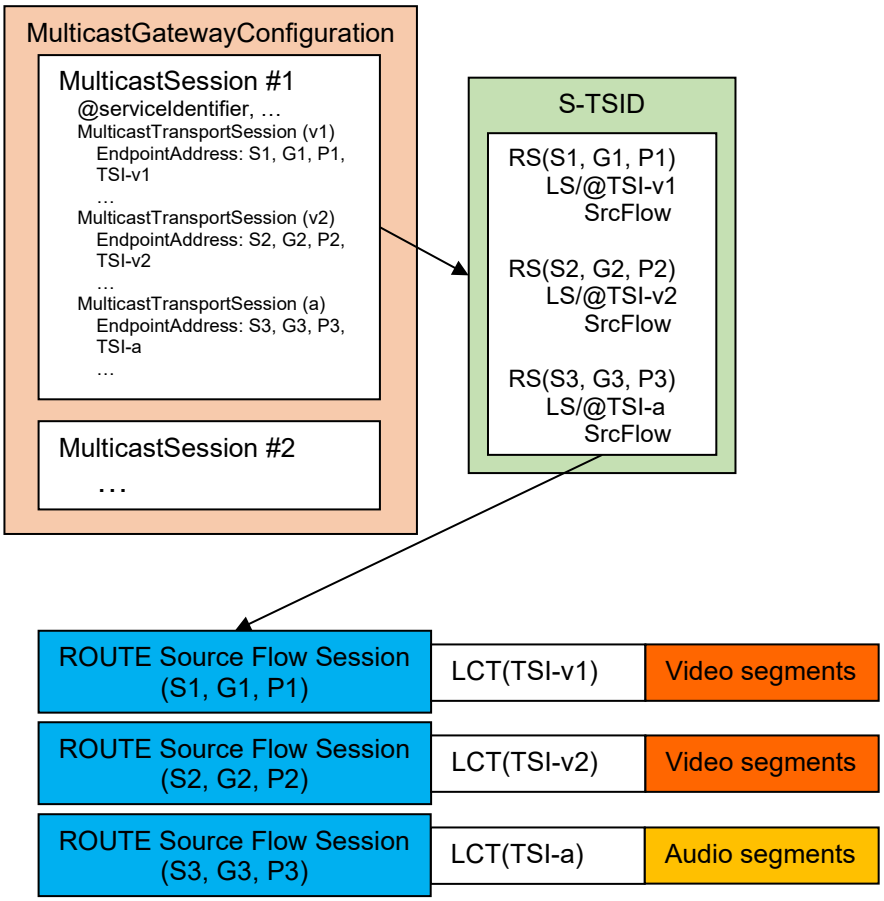


Figure I.3.3-1: Two video Source Flows with an audio on different <S, G, P>

I.3.4 Source Flows and Repair Flows carried in separate ROUTE Sessions

The final example in figure I.3.4-1 depicts Repair Flows for both audio and video service components being sent to protect their respective Source Flows.

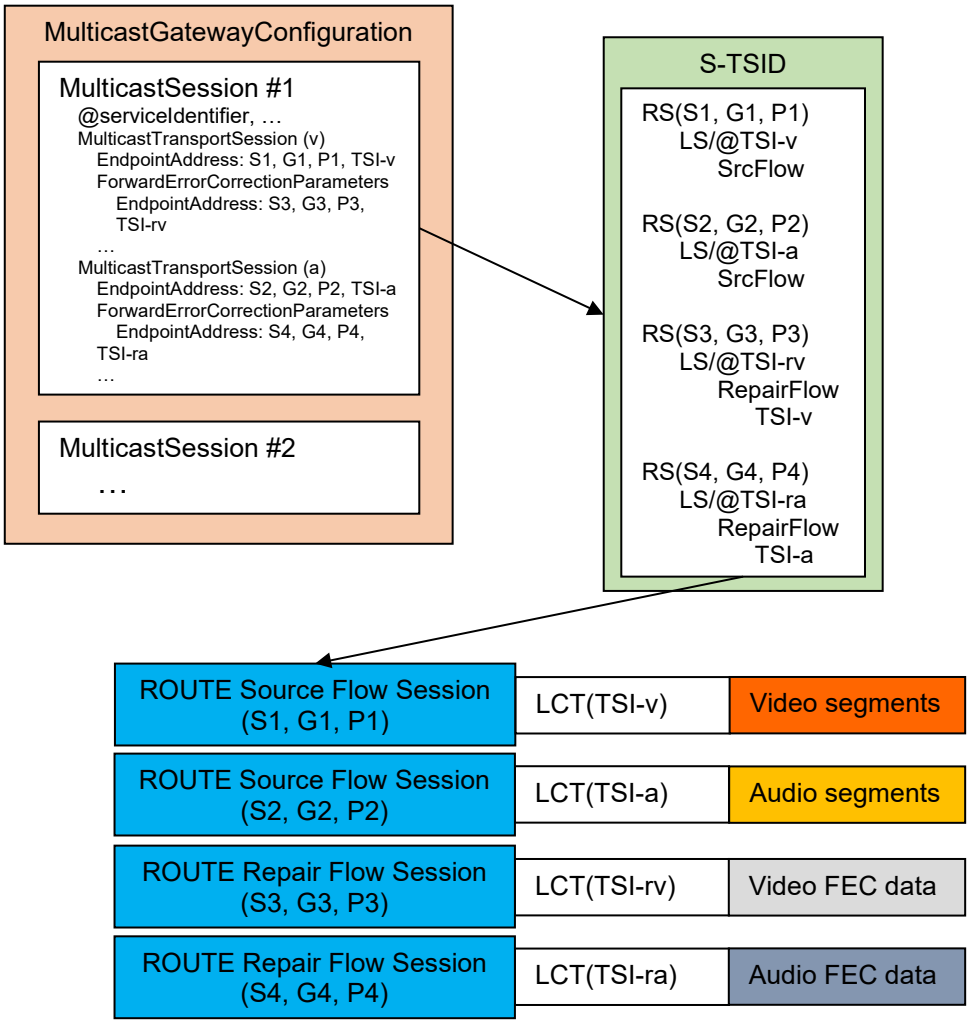


Figure I.3.4-1: Two Source Flows on different <S, G, P>, each with associated Repair Flows, also on different <S, G, P>

Annex J (normative): NORM-based multicast media transport protocol

J.0 Introduction

This annex specifies a multicast media transport protocol based on NORM (NACK-Oriented Reliable Multicast) [29] that optionally provides reliable delivery of objects by using negative acknowledgements (NACKs) to initiate protocol-level retransmission of lost or damaged packets. Section 5 of RFC 5740 [29] describes the high-level protocol operation. This annex provides guidance for using in-band FEC repair symbols for error correction. The use of NACKs as an additional mechanism for error recovery is described in clause J.3.2.3.

The profile specified in this annex differs from RFC 5740 [29] in the following aspects:

- *NORM_DATA* messages of type *NORM_OBJECT_DATA* shall be used by this profile. The *Multicast gateway* may store multicast transport objects received using these messages in volatile or non-volatile memory.
- *NORM_DATA* messages of type *NORM_OBJECT_FILE* and *NORM_OBJECT_STREAM* shall not be used by this profile.
- *NORM_INFO* metadata objects shall be sent on the same multicast transport session as the data they describe. The metadata format to use for this profile is specified in clause J.2.1 below.
- The use of Session Description Protocol or Session Announcement Protocol as specified in section 1.2 of RFC 5740 [29] shall be replaced by the **`MulticastTransportSession`** carried in the multicast session configuration instance document specified in clause 10.2.3 of the present document.
- The unicast-based repair strategies specified in clause 9 of the present document may be used in conjunction with in-band FEC-based packet repair strategies, as specified in clause J.3.2 below.
- Multiple sender mode specified in RFC 5740 [29] shall not be used by this profile.

J.1 Signalling in the multicast session configuration

The use of the multicast media transport protocol specified in this annex shall be signalled in the multicast session configuration as follows (see clause 10.2.3.8):

TransportProtocol/**@protocolIdentifier**	TransportProtocol/**@protocolVersion**
urn:dvb:metadata:cs:MulticastTransportProtocolCS:2022:NORM	*1*

Each multicast transport session shall be realised as a *NormSession* as described in section 1.2 of RFC 5740 [29]. A multicast transport object is realised in this profile by a *NORM_DATA* message transmitted on a *NormSession*.

- Multicast transmission mode may be either "resource" or "chunked" as signalled in the **`MulticastTransportSession`**`@transmissionMode` attribute specified in clause 10.2.3.5.
- In resource transmission mode, a single ingest object is mapped to single multicast transport object. In this case, the *NORM_INFO* message shall contain metadata about one complete ingest object as specified in clause J.2.2.1. *NORM_DATA* messages contain the whole ingest object as their payload.
- In chunked transmission mode, a single ingest object is transmitted as a sequence of multicast transport objects. This is used in low-latency operation where a part/chunk of the ingest object generated is available for transmission. In such case, the *NORM_INFO* message shall contain metadata about the chunk of the ingest object in question, as specified in clause J.2.2.2. *NORM_DATA* messages shall contain the current chunk of the ingest object as their payload.
- If every multicast transport object in a multicast transport session is protected by the integrity checks specified in clause J.2.1.2, the transport security mode *integrity* (see clause 10.2.3.6) shall be indicated in the multicast session configuration.

- If every multicast transport object in a multicast transport session is protected by the authenticity checks specified in clause J.2.1.3, the transport security mode *integrityAndAuthenticity* (see clause 10.2.3.6) shall be indicated in the multicast session configuration.
- If neither integrity nor authenticity is protected, the transport security mode *none* shall be indicated in the multicast session configuration.
- AL-FEC information is carried in band with the multicast transport objects it is protecting and hence **`ForwardErrorCorrectionParameters/EndpointAddress`** specified in clause 10.2.3.11 may be omitted.

J.2 Multicast server operation

J.2.0 General

The *Multicast server* and the *Unicast repair service* together play the role of the NORM sender.

A single *NormSession* may carry multicast transport objects of multiple service components from the same multicast transport session. For instance, one Representation from each Adaptation Set in an MPEG-DASH presentation may be carried in a single *NormSession*. It is also permitted to use a single *NormSession* per Representation.

Media objects referenced using the *$Number$* and/or *$Time$* URL templates in an MPEG-DASH presentation manifest shall be carried in *NORM_DATA* messages of the *NORM_OBJECT_DATA* type.

- Each multicast transport object within a *NormSession* is uniquely identified using the *object_transport_id* field in the NORM header as specified in section 4.2.1 of RFC 5740 [29].
- The length of the multicast transport object is signalled in *object_size* field of *EXT_FTI* header extension as specified in section 4.2.1 of RFC 5740 [29] and is also signalled in the length field of the metadata, as specified in clause J.2.1.1.

The FEC settings of the *NormSession* are obtained from **`ForwardErrorCorrectionParameters`** in the multicast server configuration as specified in clause 10.2.3.1. This information is added as part of the *EXT_FTI* header extension as specified in clause J.3.1.

J.2.1 Multicast transport object descriptor

J.2.1.1 Syntax

The NORM protocol requires the NORM sender to describe NORM transport objects using *NORM_INFO* messages.

- The maximum size of each *NORM_INFO* message is that of a single NORM message (packet), as specified in section 1.2 of RFC 5740 [29].
- The payload of the *NORM_INFO* message shall be a Protocol Buffers (protobuf) message [31] as specified in listing J.2.1.1-1 and table J.2.1.1-1 below.

NOTE: Protobuf [31] is a compact, efficient and cross-platform data format used to serialize structured data.

Listing J.2.1.1-1: Protobuf schema for NORM_INFO message

```
package dvbnorm;

message NormTransportObjectInfo {
    message Pair {
        string name = 1;
        string value = 2;
    }
```

```
    message ChunkInfo {
        enum ChunkStatus {
            SUCCESS =  0;
            ERROR =    1;
            PROGRESS = 2;
        }

        int32 index = 1;
        int64 offset = 2;
        ChunkStatus status = 3;
    }

    enum ObjectType {
        MEDIA_SEGMENT = 0;
        MANIFEST = 1;
        CONFIG = 2;
        OTHER  = 3;
    }

    ObjectType type = 1;
    string uri = 2;
    int64 length = 3;
    repeated Pair headers = 4;
    repeated Pair applicationData = 5;
    optional ChunkInfo chunkInfo = 6;
}
```

Table J.2.1.1-1: Protobuf message specification for use in NORM_INFO message

Field	Use	Description
type	*1*	Type of multicast transport object. - *MEDIA_SEGMENT*: Media segment. - *MANIFEST*: Presentation manifest. - *CONFIG*: Multicast gateway configuration instance document. - *OTHER*: Generic type.
uri	*1*	Multicast transport object URI.
length	*1*	Size (in bytes) of the multicast transport object. - Length of complete ingest object when used in resource transmission mode (as specified in clause J.2.2.1). - Length of the current chunk when used in chunked transmission mode (as specified in clause J.2.2.2).
headers	*0..n*	HTTP header name–value pairs associated with the multicast transport object. - The *Multicast server* may add response headers obtained at reference point **O_{in}** or request headers obtained at reference point **P_{in}'** (as appropriate), excluding connection-specific headers. - The *Multicast gateway* may use this field when generating delivery objects at reference point **L**.
applicationData	*0..n*	Application-specific name–value pairs.
chunkInfo	*0..1*	Metadata relating to chunk. Present only when using chunked transmission mode as specified in clause J.2.2.2.
index	*1*	This chunk's sequence number, starting from zero.
offset	*1*	This chunk's offset (in bytes) from the start of the ingest object.
status	*1*	Current status of the chunked transmission. - *PROGRESS*: This is not the last chunk. - *SUCCESS*: This is the last chunk and chunked transmission completed successfully. - *ERROR*: The chunked transmission was terminated early due to an error encountered by the *Multicast server*.

J.2.1.2 Multicast transport object integrity protection

The integrity of a multicast transport object may be protected by a content digest as specified in clause 12.1. In this case, the `Content-Digest` or `Repr-Digest` HTTP response header shall be conveyed in the *headers* component of the *NORM_INFO* message.

J.2.1.3 Multicast transport object authenticity assertion

The authenticity of a multicast transport object can be verified using a message signature as specified in clause 12.2. In this case, the `Signature` HTTP response header shall be conveyed in the *headers* component of the *NORM_INFO* message in association with the integrity protection specified in clause J.2.1.2. The message signature shall at minimum protect a `Content-Digest` or `Repr-Digest` header, carrying a cryptographically secure hash of the multicast transport object. The message signature should additionally protect the multicast transport object URI as carried in the *uri* field of the *NORM_INFO* message using the *@target-uri* derived component specified in clause 12.2.1.1.

If the *chunkInfo* metadata field specified in clause J.2.1.1 is present in the protobuf message associated with a multicast transport object, the message signature shall protect it using the following derived components in addition to those specified in clause 12.2.1.1 of the present document and section 2.2 of RFC 9421 [38].

1. The *@norm-chunkinfo-index* derived component refers to the chunk sequence number of the multicast transport object in the scope of the media object being transmitted. The value is a string representation of the number carried in the *index* parameter signalled in the *chunkInfo* field of the protobuf message.

   For example, the following *chunkInfo.index* parameter would result in the specified signature base:

Table J.2.1.3-1: Fragment signature base example

chunkInfo.index parameter	**2**
Signature base	"@norm-chunkinfo-index": **2**

2. The *@norm-chunkinfo-offset* derived component refers to the offset of the chunk in bytes. The value is a string representation of the number carried in the *offset* parameter signalled in the *chunkInfo* field of the protobuf message.

   For example, the following *chunkInfo.offset* parameter would result in the specified signature base:

Table J.2.1.3-2: Fragment signature base example

chunkInfo.offset parameter	**133768**
Signature base	"@norm-chunkinfo-offset": **133768**

3. The *@norm-chunkinfo-status* derived component refers to the current status of the chunked transmission. The value is a lowercase representation of the name of the enumerated value signalled in the *chunkInfo.status* field of the protobuf message.

   For example, the following *chunkInfo.status* parameter would result in the specified signature base:

Table J.2.1.3-3: Fragment signature base example

chunkInfo.status parameter	**PROGRESS (2)**
Signature base	"@norm-chunkinfo-status": **progress**

If a protobuf message accompanying a multicast transport object includes any application-specific name–value pairs carried in the *applicationData* field, these may be protected by using the *@norm-applicationdata* derived component. The *@norm-applicationdata* derived component individually addresses each name-value pair.

NOTE: The format of this derived component is based on that of the query parameters derived component specified in section 2.2.8 of RFC 9421 [38].

For example, the following two *applicationData* name–value pairs would result in the specified signature base:

Table J.2.1.3-4: Fragment signature base example

	Pair 1	Pair 2
applicationData name	**foo**	**param**
applicationData value	**bar**	**value**
Signature base	"@norm-applicationdata";name="**foo**": **bar** \ "@norm-applicationdata";name="**param**": **value**	

J.2.2 Transmission modes

J.2.2.1 Resource transmission mode

For MPEG-DASH regular latency operation, resource transmission mode shall be signalled in the multicast session configuration, as specified in clause 10.2.3.5.

In resource transmission mode, each ingest object (regardless of the content ingest method in the multicast server configuration) maps to a single multicast transport object. In this case, the *NORM_INFO* message for the multicast transport object shall contain metadata about the complete ingest object. The format of the metadata is specified in clause J.2.1.1.

The following restrictions apply to the metadata in resource transmission mode:

- *uri* field shall contain the multicast URI of the complete ingest object.

EXAMPLE: URI of media segment for type *MEDIA_SEGMENT*.

- *length* field shall contain the size (in bytes) of the complete ingest object.
- *headers* fields may be present.
- *applicationData* fields may be present.
- *chunkInfo* field shall not be present.

See clause K.3.1 for sample metadata values for resource transmission mode.

J.2.2.2 Chunked transmission mode

For MPEG-DASH low-latency operation, chunked transmission mode shall be signalled in the multicast session configuration, as specified in clause 10.2.3.5.

In the MPEG-DASH low-latency scenario, the *Content preparation* function typically pushes a continuous sequence of CMAF chunks to the *Multicast server* at reference point $\mathbf{P_{in}'}$. Multiple CMAF chunks combine to form a single complete DASH media segment. It is critical to transmit the CMAF chunks with minimum latency to the population of receiving *Multicast gateway* instances.

- In chunked transmission mode, each CMAF chunk shall be mapped to single multicast transport object for transmission at reference point **M**.
- In the general case, any HTTP chunk of defined size may be mapped to one or more multicast transport objects for transmission at reference point **M**. The *Multicast server* may combine several such chunks and map them to a single multicast transport object if the latency is within acceptable bounds.

The *NORM_INFO* message shall contain metadata about the chunk being transmitted. The format of the metadata is specified in clause J.2.1.1.

The following restrictions apply to the metadata in chunked transmission mode:

- *type* field shall be *MEDIA_SEGMENT*.
- *uri* field shall contain the multicast URI of the complete ingest object to which the chunk belongs.
- *length* field shall contain the size (in bytes) of the chunk.
- *headers* fields may be present.
- *applicationData* fields may be present.
- *chunkInfo* field shall be present.
  - *chunkInfo.status* field shall be *PROGRESS* for all chunks except the last one.
  - *chunkInfo.status* shall be set to *SUCCESS* in the last chunk provided, indicating that the *Multicast server* was able to acquire and transmit successfully over reference point **M** all the chunks pertaining to the media object. If the last chunk is not known at the time of its ingest and transmission by the *Multicast server*, a zero-length chunk with *chunkInfo.status* set to *SUCCESS* shall be transmitted after it.
  - *chunkInfo.status* values *ERROR* and *SUCCESS* shall signal abandonment and successful completion respectively of a chunked object transmission.

See clause K.3.2 for sample metadata values for chunked transmission mode.

J.2.3 In-band signalling of presentation manifest

Presentation manifest(s) may be sent in band at reference point **M** as specified in clause 8.3.5. The *NORM_DATA* message shall carry the presentation manifest as its payload.

- *type* field of the *NORM_INFO* metadata message is set to *MANIFEST* as specified in clause J.2.1.1.
- Only resource transmission mode shall be used, hence the restrictions specified in clause J.2.2.1 apply.

J.2.4 In-band signalling of multicast gateway configuration

The multicast gateway configuration instance document may be sent in band at reference point **M** as specified in clause 8.3.5. The *NORM_DATA* message shall carry a multicast gateway configuration instance document as its payload.

- *type* field of the *NORM_INFO* metadata message is set to *CONFIG* as specified in clause J.2.1.1.
- Only resource transmission mode shall be used, hence the restrictions specified in clause J.2.2.1 apply.

J.2.5 Sending application-specific payloads

If the data sent at reference point **M** is not a media segment or a manifest or a configuration, then the generic *NORM_INFO* metadata message type *OTHER* shall be used to signal the metadata. Only resource transmission mode shall be used.

J.3 Multicast gateway operation

J.3.0 General

The *Multicast gateway* plays the role of the NORM receiver.

J.3.1 Forward Error Correction

Any of the Application Layer Forward Error Correction schemes specified in annex B.2 may be used in conjunction with the NORM-based multicast media transport protocol.

- The AL-FEC scheme used is signalled by the *fec_id* field of the *NORM_DATA* and *NORM_INFO* headers as specified in section 4.2.1 of RFC 5740 [29]. The value of this field shall be the same as the terminal path element of **ForwardErrorCorrectionParameters**`@SchemeIdentifier` (see clause 10.2.3.11).

- FEC-specific parameters such as the block size, symbol size and parity size are signalled in the FEC Object Transmission Information (*EXT_FTI*) header extension to the *NORM_DATA* and *NORM_INFO* messages, as specified in section 4.2.1 of RFC 5740 [29].

- The position of the missing symbols (packets) is inferred from the *block id* and *symbol id* fields of the *fec_payload_id* field in received symbols as specified in section 4.2.1 of RFC 5740 [29]. The format of this field is specific to the AL-FEC scheme used. For instance, the *fec_payload_id* field of the *NORM_DATA* header and the *EXT_FTI* header extension for Reed-Solomon systematic FEC code are specified in RFC 5510 [30].

- The *Multicast gateway* shall use the repair symbols received at reference point **M** to recover erased packets and should use unicast repair only if the received repair symbols are inadequate for recovery.

J.3.2 Unicast repair

J.3.2.1 Triggering unicast repair

If the in-band FEC parity symbols are not sufficient to recover the missing packets, the *Multicast gateway* may start the NACK-based packet repair procedure defined in clause J.3.2.2 below, as specified in section 5.3 of RFC 5740 [29].

If the missing symbols cannot be recovered using NACK, the *Multicast gateway* may use the HTTP-based repair protocol at reference point **A** as detailed in clause 9.2 of the present document and in clause J.3.2.3 below.

NACK-based packet repair should only be performed within a multicast transport object. If an entire multicast transport object is lost, the *Multicast gateway* should use the HTTP-based object repair mechanism at reference point **A** as detailed in clause 9.2 of the present document and in clause J.3.2.3 below to retieve the media object instead of requesting a whole multicast transport object using NACK-based packet repair.

In the case of chunked transmission mode, clause J.2.2.2 specifies that each chunk of a media object is transmitted as a unique multicast transport object. If the next expected chunk is not received within the timeout period specified by **UnicastRepairParameters/**`@transportObjectReceptionTimeout`, the *Multicast gateway* shall use the specified HTTP-based object repair mechanism to retrieve the rest of the media object.

In the case where chunked transmission of a media object is abandoned by the *Multicast server* before completion (signalled in *chunkInfo.status* per clause J.2.2.2) the *Multicast gateway* shall use the HTTP-based object repair mechanism at reference point **A** as detailed in clause 9.2 of the present document and in clause J.3.2.3 below to retrieve the rest of the media object.

J.3.2.2 NACK-based packet repair

This clause outlines the procedure for enabling NACK and ACK feedback messages in the reference architecture specified in clause 5.2. and the associated repair mechanism.

The *Multicast gateway* may send feedback messages to the *Unicast repair service* at reference point **U**. The *Unicast repair service* shall forward this traffic to the *Multicast server* through reference point **U'**. (The *Unicast repair service* here is essentially acting as a forward proxy.) NACK messages shall be conveyed at reference point **U** using the same transport protocol (e.g. UDP) as that used at reference point **M**.

J.3.2.3 HTTP-based object repair

The *Multicast gateway* shall follow the unicast repair procedure specified in clause 9.

- The start offset of the missing symbol (packet) and its length shall be derived from the fields of the FEC Object Transmission Information header (*EXT_FTI*). If multiple packets of a single multicast transport object are missing, the multicast gateway shall issue a single HTTP range request with those ranges.

- In case of chunked transmission mode, the *chunkInfo.offset* and *length* fields in the *NORM_INFO* metadata message of the previous chunk (see clause J.2.1.1) shall be used to derive the offset of the missing chunk.

Annex K (informative): Implementation guidelines for NORM-based multicast media transport protocol

K.0 Introduction

NORM [29] is a standards-based multicast transport protocol offering the use of selective negative acknowledgements (NACKs) to request repairs over reference point **U** of missing data to be re-transmitted at reference point **M** ("multicast repair"). NORM provides for the use of packet-level Forward Error Correction (FEC) techniques for efficient multicast repair and optional proactive transmission robustness. FEC-based repair can be used to greatly reduce the quantity of reliable multicast repair requests and repair transmissions in a NACK-oriented protocol.

The additional optional use of a byte-range unicast repair protocol (see clause 9.2) can also be used to compliment NORM/FEC as needed, depending on the underlying network characteristics on which the system is being deployed. This effectively defines three levels of content delivery resiliency: in-band FEC, NACK-based multicast repair and HTTP-based unicast repair.

K.1 Multicast system implementation guidelines

K.1.0 General

This clause describes the implementation guidelines of the multicast system.

K.1.1 Guidance on use of NACK-based packet repair

NACK (Negative ACKnowledgment) is a mechanism in which receivers request the sender to re-transmit the packets lost during the transmission to achieve reliability. This helps in achieving less frequent feedback messaging than reliability protocols based on positive acknowledgement and hence simplifies the problem of feedback implosion as the receiver group size grows larger. NACK-based reliable protocols like NORM make use of FEC erasure coding techniques during NACK operation. Packet-level erasure coding allows missing packets from a given FEC block to be recovered using the parity packets instead of classical, individualized retransmission of original source data content.

In addition to using NACK, NORM also provides an option to proactively transmit some amount of redundancy (in-band FEC) in the form of FEC parity symbols along with the original content to potentially enhance performance.

Following are some important points about NORM's NACK mechanism:

- NACK-based error recovery is efficient especially when multiple receivers are facing errors in similar FEC blocks. Point 4 of clause 5.3.4 in the present document mentions such a case.
- The *Multicast server* prioritizes sending the FEC repair symbols instead of the source symbols in response to the set of NACK messages from different receivers. This helps receivers to recover the symbols irrespective of the position of the missing symbol in the FEC block, thereby reducing the size of the multicast retransmission.
- The *Multicast server* aggregates the repair state when it receives the NACK messages from different *Multicast gateway* instances via the *Unicast repair service* at reference point **U'**. It also relays the current repair state to all the receivers using the *NORM_CMD(REPAIR_ADV)* message at reference point **M**. This has the effect of suppressing the transmission of redundant NACK requests from other NORM receivers.
- NORM uses probabilistic suppression of redundant feedback based on exponentially distributed random backoff timers. It dynamically measures the multicast group's round-trip timing (GRTT) status to set its suppression timer and other protocol timers. This allows NORM to scale well while maintaining reliable data delivery transport with low latency relative to the network topology over which it is operating. Section 5.3 of RFC 5740 [29] details the back-off mechanism.

RFC 5740 [29] defines two types of feedback message that a NORM receiver may send to the NORM sender:

- The *NACK* message (see section 4.3.1 of [29]) is used for error recovery, as specified in clause J.3.2.2.
- The *ACK* message (see section 4.3.2 of [29]) is mainly used for GRTT (Group Round-Trip Time) measurement and congestion control operations.

K.2 Example NORM session configurations

Table K.2-1: Example NORM session configurations

Scenario	MulticastSession Configuration	NORM session configuration	Comments
Single audio and single video service components multiplexed into one multicast transport session	Video – (S1, G1, P1) Audio – (S1, G1, P1)	$NORMSession_{mux}$ (S1, G1, P1)	Audio and video representations are sent in the same NORM session.
Multiple audio and video service components multiplexed into one multicast transport session	Video1 – (S1, G1, P1) Video2 – (S1, G1, P1) Audio1 – (S1, G1, P1) Audio2 – (S1, G1, P1)	$NORMSession_{mux}$ (S1, G1, P1)	All four video and audio representations are sent in the same NORM session.
Single audio and single video service components sent in separate multicast transport sessions	Video – (S1, G1, P1) Audio – (S2, G2, P2)	$NORMSession_{video}$ (S1, G1, P1) $NORMSession_{audio}$ (S2, G2, P2)	Video is sent in $NORMSession_{video}$ and audio is sent in $NORMSession_{audio}$.
Multiple video and multiple audio service components sent in separate multicast transport sessions	Video1 – (S1, G1, P1) Video2 – (S2, G2, P2) Audio1 – (S3, G3, P3) Audio2 – (S4, G4, P4)	$NORMSession_{video1}$ (S1, G1, P1) $NORMSession_{video2}$ (S2, G2, P2) $NORMSession_{audio1}$ (S3, G3, P3) $NORMSession_{audio2}$ (S4, G4, P4)	Each video and audio representation is sent in a separate NORM session.

K.3 Example multicast transport object metadata

K.3.1 Sample metadata values for resource transmission mode

Table K.3.1-1 illustrates the sample metadata values for a media segment segment1234.m4s whose length is 1,048,576 bytes transmitted in resource transmission mode.

Table K.3.1-1: Sample metadata values for resource transmission mode

Field	Value
Type	MEDIA_SEGMENT
Uri	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s**
Length	1048576
Headers	headers[0] => name : "Content-Type", value : "video/mp4" headers[1] => name : "Content-Encoding", value : "gzip"
applicationData	Omitted.
chunkInfo	Omitted.

K.3.2 Sample metadata values for chunked transmission mode.

Table K.3.2-1 illustrates the sample metadata values for a chunk that belongs to media segment segment1234.m4s. This is the third chunk, starting at an offset 262,100 and having a length of 131,072 bytes. This is not the last chunk, and more chunks are expected to follow.

Table K.3.2-1: Sample metadata values for chunked transmission mode

Field	Value
Type	MEDIA_SEGMENT
Uri	tag:multicastserver.isp.net,2019-01-01:mts100,3922:**segment1234.m4s**
Length	131072
Headers	headers[0] => name : "Content-Type", value : "video/mp4" headers[1] => name : "Content-Encoding", value : "gzip"
applicationData	Omitted
chunkInfo.index	2
chunkInfo.offset	262100
chunkInfo.status	PROGRESS

Annex L (normative): MSync-based multicast media transport protocol

L.0 Introduction

This annex specifies an optional multicast media transport protocol based on MSync as specified in [32]. MSync transfers media objects over multicast UDP/IP or multicast RTP/UDP/IP. Although generic, MSync has been designed primarily for transporting HTTP adaptive streaming objects – media segments (e.g., MP4 [i.18], CMAF [i.11]) as well as MPEG-DASH presentation manifests [i.2] and/or HLS playlists [i.5] – between a *Multicast server* and a population of *Multicast gateway* instances.

The MSync transport protocol in its simplest form is carried by UDP over IP multicast. Alternatively, MSync supports carriage by RTP [33] according to section 3.9 of [32]. Operating MSync over RTP permits a system operator to reuse traffic monitoring tools that work with RTP, such as those deployed in legacy IPTV environments.

L.1 Signalling in the multicast session configuration

The use of the multicast media transport protocol specified in this annex shall be signalled in the multicast session configuration as follows (see clause 10.2.3.8):

TransportProtocol/**@protocolIdentifier**	TransportProtocol/**@protocolVersion**
urn:dvb:metadata:cs:MulticastTransportProtocolCS:2022:MSync	*3*
urn:dvb:metadata:cs:MulticastTransportProtocolCS:2022:MSync:RTP	*3*

The multicast transmission mode, as specified in clause 10.2.3.5, shall indicate whether multicast transport objects are formatted using resource transmission mode or chunked transmission mode as specified in clause L.2 below.

If every multicast transport object in a multicast transport session is protected by the integrity checks specified in clause L.2.4, the transport security mode *integrity* (see clause 10.2.3.6) shall be indicated in the multicast session configuration.

If every multicast transport object in a multicast transport session is protected by the authenticity checks specified in clause L.2.5, the transport security mode *integrityAndAuthenticity* (see clause 10.2.3.6) shall be indicated in the multicast session configuration.

If neither integrity nor authenticity is protected, the multicast transport security mode *none* shall be indicated in the multicast session configuration.

The IP source address (in the case of source-specific multicast), IP multicast group destination address and destination UDP port number of the MSync transport multicast session shall be signalled in the multicast transport endpoint address, as specified in clause 10.2.3.9.

MSync does not define a transport-level multiplexing identifier. When MSync packets are transported over RTP [33] as described in section 3.9 of [32], the RTP payload type value shall be signalled in the multicast media transport protocol session identifier (see also clause 10.2.3.9). Otherwise, the multicast media transport protocol session identifier shall be omitted.

The use of Application Layer Forward Error Correction (AL-FEC) with MSync is for further study. Accordingly, the **ForwardErrorCorrectionParameters/EndpointAddress** element (see clause 10.2.3.11) shall be omitted from the multicast session configuration.

L.2 Protocol mapping

L.2.0 General

Each multicast transport session (as defined in clause 3.1) shall be realised as a single MSync transport multicast session.

The multicast transport object shall be realised as an MSync object. MSync permits the delivery of both media objects and ancillary media objects as defined in clause 3.1.

Depending on the multicast transport object type and multicast transmission mode, the MSync object payload is transported in MSync packets of type *object data* or *object data-part*, as specified in clause L.2.1 below.

Each MSync object shall be associated with metadata carried in one *object info* packet and, optionally, with one or more *object http header* packets.

NOTE: Each HTTP header may be carried in one or more MSync *object http header* packets, as specified in section 3.4 of [32].

The multicast transport object identifier defined in the present document is realised by the *object identifier* field in the MSync packet header. The *object info* and *object http header* packets carry the same object identifier as the *object data* or *object data part* packets that they describe.

The multicast transport object URI defined in the present document is realised by the *object URI* field present in the MSync *info* packet. The content of this field is governed by clauses 8.3.3 and 8.4.4, and according to sections 3.2 and 3.8.1.2 of [32].

L.2.1 Signalling of multicast transport object types

L.2.1.0 Summary

The different types of multicast transport objects shall be signalled as specified in table L.2.1.0-1:

Table L.2.1.0-1: Signalling of multicast transport objects in MSync

Multicast transport object type		MSync Packet *packet type* (see section 3.1 of [32])	Object Info Packet *object type* (see section 3.2 of [32])
Ancillary media object		*object data*	0x05 "control"
Media object	Presentation manifest	*object data*	0x01 "media manifest"
	Initialisation segment	*object data*	0x03 "media content (e.g. an MPEG-2 TS or CMAF container format)"
	Media segment	*object data* or *object data-part*	

L.2.1.1 Ancillary media object

MSync packets carrying ancillary media objects (such as the multicast session configuration instance document) shall indicate type *object data* (see section 3.1 of [32]).

The *object info* packet describing such objects (see section 3.2 of [32]) shall have the *object type* field set to 0x05 ("control").

L.2.1.2 Presentation manifest

MSync packets carrying presentation manifests shall indicate type *object data* (see section 3.1 of [32]).

The *object info* packet describing such objects (see section 3.2 of [32]) shall have the *object type* field set to 0x01 ("media manifest").

L.2.1.3 Initialisation segment

MSync packets carrying initialisation segments shall indicate type ***object data*** (see section 3.1 of [32]).

Depending on the media container format (e.g. fragmented MP4 [i.18]), the corresponding media data ***object type*** value is indicated in the ***object info*** packet describing such objects as specified in section 3.2 of [32].

L.2.1.4 Media segment

MSync packets carrying media segments shall indicate type ***object data*** (see section 3.1 of [32]).

- When the multicast transport session is configured in resource transmission mode as specified in clause 10.2.3.5, multicast transport objects shall be transmitted in ***object data*** packets.
- When the multicast transport session is configured in chunked transmission mode as specified in clause 10.2.3.5, multicast transport objects shall be transmitted in ***object data-part*** packets.

NOTE: Further considerations for the use of low-latency delivery with MSync are described in clause L.2.3 below.

Depending on the media container format (e.g., fragmented MP4 [i.18]), the corresponding media data ***object type*** value is indicated in the ***object info*** packet describing such objects as specified in section 3.2 of [32].

L.2.2 Packet transmission order

When sending an MSync object, the *Multicast server* shall transmit packets in the order specified in section 3.7.1 of [32].

L.2.3 Low-latency delivery

Clause 8.3.4.1 describes the chunked transmission mode wherein the *Multicast server* transmits the ingest object as one or more multicast transport objects. Such multicast transport objects are referred to as ***media data-part*** objects in MSync and are transported using one or more ***object data-part*** packets as specified in clause L.2.1.4.

In chunked transmission mode, the size of the ingest object is not known in advance. Instead, the size of each chunk is signalled in the object size field of the ***object info*** packet that describes the ***media data-part*** object.

When the *Multicast server* has sent the final chunk of payload data, the *Multicast server* signals the end of transmission for a given ingest object by sending an ***object info*** packet with the object size set to 0. This is done through sending an ***object info*** packet having the object size set to 0 as detailed in sections 3.2 and 3.8.1.2 of [32].

An example of the workflow between the *Content hosting* function, the *Multicast server* and the *Multicast gateway* for low-latency delivery can be found in clause M.2.1.

L.2.4 Multicast transport object integrity protection

The integrity of a multicast transport object may be protected in MSync by using the (insecure) object CRC header field in the MSync ***object info*** packet as specified in section 3.2 of [32], and/or by a secure content digest (as profiled in clause 12.1) conveyed in an MSync ***object http header*** packet.

L.2.5 Multicast transport object authenticity protection

The authenticity of a multicast transport object may be asserted by conveying a message signature as profiled in clause 12.2 in an ***object http header*** packet. The message signature shall at minimum protect a `Content-Digest` or `Repr-Digest` header, carrying a cryptographically secure hash of the multicast transport object. The message signature should additionally protect the multicast transport object URI as carried in the object URI field of the ***object info*** packet using the ***@target-uri*** derived component as specified in clause 12.2.1.1.

L.3 Error recovery

L.3.1 Forward Error Correction (FEC)

The use of Application Level Forward Error Correction (AL-FEC) with MSync is for further study.

L.3.2 Unicast RTP repair

When MSync is carried over RTP [33], RTP retransmission may be used to facilitate packet loss recovery. When RTP packet loss recovery is used, the *Unicast repair service* shall play the role of the RTP Repair server specified in section 3.9.2 of [32].

L.3.3 Unicast HTTP repair

If it supports HTTP-based unicast repair, the *Multicast gateway* shall follow the unicast repair procedure specified in clause 9 of the present document.

- Missing byte ranges shall be detected by observing discontinuity in the object offset field present in all MSync packets.
- If operating MSync on top of RTP as described in clause L.0, packet loss may additionally be detected by observing discontinuity in the sequence numbers of received RTP packets.

Annex M (informative): Implementation guidelines for MSync-based multicast media transport protocol

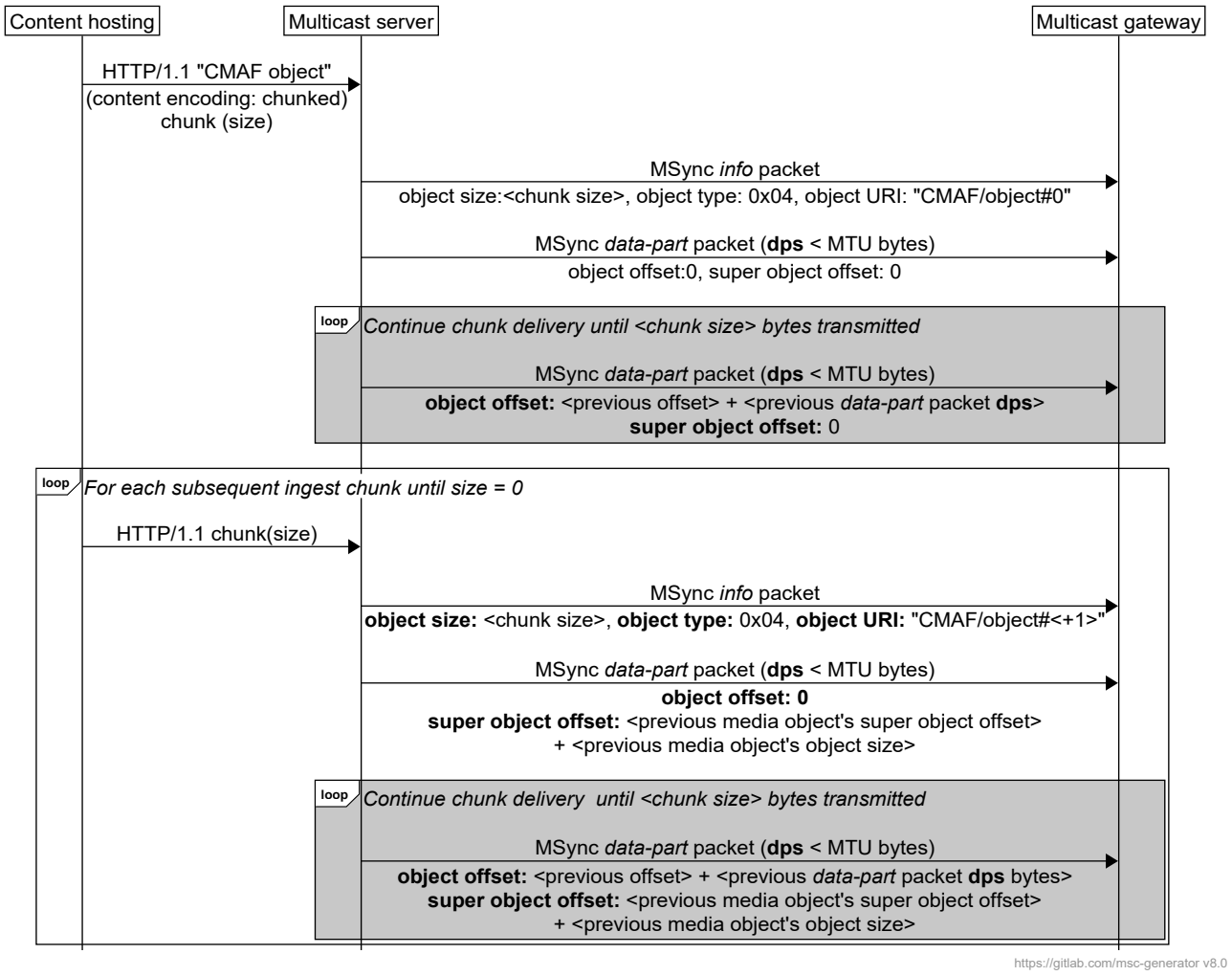
M.1 Multicast system implementation guidelines

This clause is for future study.

M.2 Example workflows

M.2.1 Example low-latency workflow

Figure M.2.1-1 shows an example of the workflow between the *Content hosting* function, the *Multicast server* and the *Multicast gateway* assuming the use of HTTP/1.1 chunked transfer coding transmission.



NOTE: "dps" refers to the MSync *data-part* packet payload size in bytes.

Figure M.2.1-1: Low-latency delivery over MSync

Annex N (normative): Service reporting information document schema

This annex contains a YAML description of the service reporting information document syntax according to OpenAPI version 3.0.1 [37].

```
openapi: 3.0.1
info:
  title: Service reporting information document synatx
  description: 'Service reporting information document syntax as specified in DVB BlueBook A168
Annex N.'
  contact:
    email: 'dvb@dvb.org'
  version: 1.0.0
externalDocs:
    description: Find out more
    url: 'https://dvb.org/?standard=adaptive-media-streaming-over-ip-multicast'
paths:
  /dvb/mabr/reportingInformationInstance:
    post:
      summary: 'To send one reporting information instance document.'
      operationId: dvb_mabr_reporting
      description: 'Allowing one or more events related to one or more streaming sessions managed
by the Multicast gateway to be reported.'
      requestBody:
        content:
          application/json:
            schema:
              type: object
              required:
                - report
              properties:
                report:
                   $ref: '#/components/schemas/MABR_Report'
        required: true
      responses:
        200:
          description: 'Document digested succesfully.'
        400:
          description: 'Bad request.'
        403:
          description: 'Forbidden.'
components:
  schemas:
    MABR_Report:
      type: object
      description: 'One reporting information instance document.'
      required:
        - timestamp
        - gateway-id
        - event
      properties:
        timestamp:
          description: 'Date/time (UTC) when this service reporting information instance document
was generated. The value of this property shall comply with the MPEG-7 TimePoint format.'
          type: string
        gateway-id:
          description: 'Opaque identifier that uniquely identifies the Multicast gateway instance
in the system.'
          type: string
        gateway-description:
          description: 'Description of the Multicast gateway instance.'
          type: string
        events:
          description: 'A list of events to be reported.'
          type: array
          items:
            $ref: '#/components/schemas/MABR_Event'
        playback-sessions:
          description: 'Metrics for a set of playback sessions'
          type: array
          items:
            $ref: '#/components/schemas/MABR_Session'
    MABR_Event:
      type: object
```

```
      description: 'An event to be reported'
      required:
        - timestamp
        - playback-session-id
        - type
      properties:
        timestamp:
          description: 'Date/time (UTC) when this service reporting information instance document
was generated. The value of this property shall comply with the MPEG-7 TimePoint format.'
          type: string
        playback-session-id:
          description: 'Unique identifier for the playback session, possibly communicated by the
Content playback function.'
          type: string
        type:
          description: 'The type of event.'
          type: string
          enum: [session-started, heartbeat, service-component-switch, multicast-join, multicast-
leave, object-delivery, session-ended]
        object-delivery-status:
          description: 'Delivery status of a media object. Present if the event type is object-
delivery.'
          type: string
          enum: [cache-miss-incomplete,cache-miss-filter,cache-miss-nodata-s,cache-miss-nodata-
m,cache-miss-nodata-j,cache-miss-nodata-o,cache-miss-timeshift,cache-hit-m,cache-hit-a]
        service-component-identifier:
          description: 'An object that uniquely identifies a service component in the
presentation manifest attached with that playback session. Present if the event type is service-
component-switch.'
          $ref: '#/components/schemas/MABR_ComponentIdentifier'
        multicast-endpoint-address:
          description: 'Present if the event type is multicast-join or multicast-leave.'
          $ref: '#/components/schemas/MABR_MulticastAddress'
    MABR_Session:
      type: object
      description: 'A playback sessions currently served by the multicast gateway.'
      required:
        - playback-session-id
        - service-id
        - manifest-id
        - manifest-url
      properties:
        playback-session-id:
          description: 'Unique identifier for the playback session, possibly communicated by the
Content playback function.'
          type: string
        service-id:
          description: 'The service identifier of the logical service.'
          type: string
        manifest-id:
          description: 'The unique identifier of the presentation manifest as specified in
10.2.2.2 used by the Multicast gateway to deliver the presentation session.'
          type: string
        manifest-url:
          description: 'The presentation manifest URL requested by the Content playback function
at reference point L.'
          type: string
        user-agent:
          description: 'Unspecified string identifying the user agent.'
          type: string
        in-session-metrics:
          $ref: '#/components/schemas/MABR_InSessionMetrics'
        session-end-metrics:
          $ref: '#/components/schemas/MABR_SessionEndMetrics'
    MABR_InSessionMetrics:
      type: object
      description: 'Metrics collected during a playback session'
      required:
        - cache-rep-a
        - cache-rep-f
        - rep-switch-nb
        - bytes-received-m
        - bytes-received-a
      properties:
        errors-l:
          description: 'Number of errors returned by the Multicast gateway instance so far during
the playback session at reference point L during the playback session.'
          type: integer
```

```
          minimum: 0
        errors-a:
          description: 'Number of errors so far during the playback session at reference point
A.'
          type: integer
          minimum: 0
        errors-m:
          description: 'Number of errors so far during the playback session at reference point
M.'
          type: integer
          minimum: 0
        errors-u:
          description: 'Number of errors so far during the playback session at reference point
U.'
          type: integer
          minimum: 0
        cache-rep-a:
          description: 'Number of KBytes of unicast repair data received so far during the
playback session over reference point A.'
          type: integer
          minimum: 0
        cache-rep-f:
          description: 'Number of KBytes of lost multicast transport object data successfully
recovered so far during the playback session through the use of AL FEC.'
          type: integer
          minimum: 0
        cache-rep-u:
          description: 'Number of KBytes of unicast repair data received so far during the
playback session over reference point U.'
          type: integer
          minimum: 0
        rep-switch-nb:
          description: 'Number of representation changes so far during the playback session.'
          type: integer
          minimum: 0
        segment-req-nb:
          description: 'Number of segment requests received so far during the playback session at
reference point L.'
          type: integer
          minimum: 0
        bytes-received-m:
          description: 'Number of bytes received so far during the playback session via reference
point M.’
          type: integer
          minimum: 0
        bytes-received-a:
          description: ‘Number of bytes received so far during the playback session over
reference point A.'
          type: integer
          minimum: 0
        bytes-received-l:
          description: 'Number of bytes received so far during the playback session over
reference point L.'
          type: integer
          minimum: 0
        bytes-sent-l:
          description: 'Number of bytes delivered so far during the playback session over
reference point L'
          type: integer
          minimum: 0
    MABR_SessionEndMetrics:
      type: object
      description: 'Final metrics at the end of a playback session.'
      required:
        - service-component-information
      properties:
        service-component-information:
          description: 'List of statistics per service component.'
          type: array
          items:
            $ref: '#/components/schemas/MABR_ServiceComponentInformation'
    MABR_ServiceComponentInformation:
      type: object
      description: 'Final metrics about a service component of a playback session.'
      required:
        - service-component-identifier
        - total-bytes
        - bit-rate
```

```
        - cache-miss-expired
        - cache-miss-incomplete
        - cache-miss-filter
        - cache-miss-nodata-m
        - cache-miss-nodata-s
        - cache-miss-nodata-j
        - cache-miss-nodata-o
        - cache-hit-m
        - cache-hit-a
        - cache-hit-mr
      properties:
        service-component-identifier:
          description: 'An object that uniquely identifies a service component in the
presentation manifest associated with the playback session.'
          $ref: '#/components/schemas/MABR_ComponentIdentifier'
        multicast-endpoint-address:
          description: 'Multicast endpoint address of the service component.'
          type: array
          items:
            $ref: '#/components/schemas/MABR_MulticastAddress'
        segment-duration:
          description: 'The segment duration in milliseconds as declared in the presentation
manifest.'
          type: integer
          minimum: 1
        total-bytes:
          description: 'Number of bytes delivered over reference point L for this service
component.'
          type: integer
          minimum: 0
        bit-rate:
          description: 'The bit rate of the service component as conveyed in the presentation
manifest, expressed in bits per second.'
          type: integer
          minimum: 1
        cache-miss-expired:
          description: 'Total number of media objects for this service component retrieved at
reference point A because the media object expired from the Asset storage subfunction, for
example because playback is significantly behind the multicast live edge.'
          type: integer
          minimum: 0
        cache-miss-incomplete:
          description: 'Total number of media objects repaired using reference point A and/or U
because only partly received at reference point M.'
          type: integer
          minimum: 0
        cache-miss-filter:
          description: 'Total number of segments fetched at reference point A and corresponding
to an inactive multicast session.'
          type: integer
          minimum: 0
        cache-miss-nodata-m:
          description: 'Total number of media objects fetched at reference point A because no
data related to that playback session was received via reference point M.'
          type: integer
          minimum: 0
        cache-miss-nodata-s:
          description: 'Total number of media objects fetched at reference point A because they
were not received via reference point M although other data was received in relation with that
playback session.'
          type: integer
          minimum: 0
        cache-miss-nodata-j:
          description: 'Total number of media objects fetched at reference point A before the
Multicast gateway subscribed to the multicast transport session.'
          type: integer
          minimum: 0
        cache-miss-nodata-o:
          description: 'Total number of media objects fetched at reference point A for any other
unspecified reason.'
          type: integer
          minimum: 0
        cache-miss-nodata-l:
          description: 'Total number of requests at reference point L that the Multicast gateway
could not fulfil because the requested playback delivery object was not received in time via
reference point M and nor could it be retrieved via reference point A.'
          type: integer
          minimum: 0
```

```
        cache-hit-m:
          description: 'Total number of media objects retrieved from the Asset storage cache
received via reference point M.'
          type: integer
          minimum: 0
        cache-hit-a:
          description: 'Total number of media objects retrieved from the Asset storage cache
received via reference point A.'
          type: integer
          minimum: 0
        cache-hit-mr:
          description: 'Total number of media objects retrieved from the Asset storage cache
received via reference point M but repaired over reference point A or reference point U.'
          type: integer
          minimum: 0

    MABR_ObjectDeliveryStatus:
      description: 'Delivery status of a media object.'
      type: string
      enum: [cache-hit-m, cache-hit-a, cache-miss-expired, cache-miss-incomplete, cache-miss-
filter, cache-miss-nodata-s, cache-miss-nodata-m, cache-miss-nodata-j, cache-miss-nodata-o,
cache-miss-expired, ]
    MABR_ComponentIdentifier:
      type: object
      description: 'An object that uniquely identifies a service component in the presentation
manifest associated with the playback session.'
      oneOf:
        - $ref: '#/components/schemas/DashComponentIdentifier'
        - $ref: '#/components/schemas/HlsComponentIdentifier'
    DashComponentIdentifier:
      type: object
      description: 'Container unambiguously identifying an MPEG-DASH representation.'
      required:
        - period-id
        - adaptation-sed-id
        - representation-id
      properties:
        period-id:
          description: 'A Period/@id from the MPD referenced in the manifest-id attribute of the
parent structure.'
          type: string
        adaptation-set-id:
          description: 'An AdaptationSet/@id from the MPD referenced in the manifest-id attribute
of the parent structure.'
          type: string
        representation-id:
          description: 'A Representation/@id from the MPD referenced in the manifest-id attribute
of the parent structure.'
          type: string
    HlsComponentIdentifier:
      type: object
      description: 'Container identifying an HLS Media Playlist.'
      required:
        - media-playlist-locator
      properties:
        media-playlist-locator:
          description: 'The absolute URL of an HLS Media Playlist appearing in the HLS Master
Playlist referenced in the manifest-id attribute of the parent structure.'
          type: string
    MABR_MulticastAddress:
      type: object
      description: ''
      required:
        - group
        - port
      properties:
        source:
          description: 'If the multicast transport session uses source-specific multicast
transport, this gives the host name or IP address specifying the multicast group source address.'
          type: string
        group:
          description: 'The IP address specifying the multicast group for the multicast transport
session.'
          type: string
        port:
          description: 'The UDP port number of the multicast transport session.'
          type: integer
          minimum: 0
```

```
          maximum: 65535
        transport-session-id:
          description: 'The multicast transport session identifier, if configured.'
          type: string
```

History

Document history		
V1.1.1	November 2020	ETSI Publication
V1.2.1 Interim Draft (A176r2)	January 2022	DVB BlueBook A176 third edition (A176r2 - Revision 2). Phase 2b technical specification with reference architecture updated to disambiguate reference points A, and adding: - Clarification of URL signalling in multicast session configuration instance documents. - Guidance on acquisition of media objects by the Multicast server. - Support for in-band carriage of presentation manifests and initialisation segments in individual multicast transport streams, including signalling of carousel repetition frequency. - Means to explicitly signal the name of media objects in multicast transport sessions, including a well-known name for the multicast gateway configuration object when conveyed in the multicast gateway configuration transport session. - Means to explicitly signal the presentation manifest playback path pattern to the Multicast gateway function. - Advice on use of sticky redirects between the Multicast rendezvous service and the Multicast gateway. Multicast session configuration schema and examples updated to reflect additions. Updated schema namespace identifier in annexes A and C.
V1.2.1 Interim Draft (A176r3)	March 2023	Added macro expansion mechanism to clauses 10.2.3.13 and 10.2.5.2. Added text to allow optional multicast transport protocol support in clause 4, and specify the behaviour of the Multicast gateway when a manifest request does not match its configuration in clause. Revised text to allow optional multicast transport protocol support in clause 4, clarified what AL-FEC schemes are valid for each multicast transport protocol in clause X, and prevent request of media segments not yet available in clause 8.4.1.0. Updated TS 26.346 reference to release 17 to allow GZip compression in the FLUTE FDT. Added dynamic registration of Multicast gateway instances in clause 7.6, and added the optional NORM multicast transport protocol in annexes J and K. Added the optional MSync multicast transport protocol in annexes L and M. Updated all HTTP specification references to the new June 2022 series and added references to HTTP/3. Added the service reporting specification in clause 11 and annex N, as well as the requisite changes in clause 10.2, annex A.2, annex C.1, and annex C.3. Corrections to YAML syntax in annex N. Added integrity and authenticity specification for all four specified multicast transport protocols. Implemented feedback received immediately prior to DVB TM123 meeting.

V1.2.1 Interim Draft (A176r4)	July 2023	Fixed spelling mistakes in clause A.1 schema and annex C examples. Corrected MIME type for DASH MPD in annex C.1 and C.3 examples. Corrected BasePath to BaseURL in figure 10.2-1.
V1.2.1 Interim Draft (A176r5)	February 2024	Corrected abbreviation definitions for TOI and TSI. Corrected the **`NetworkSourceAddress`** element in the XML schema to be optional as described in clause 10.2. Updated the multicast session configuration examples in annex C with more believable dates. Added the ability to externally reference instances of the **`MulticastGatewayConfigurationTransportSession`**. Added the ability to provide hints to *Multicast gateway* functions of the processing requirements for objects carried in an object carousel. Added the ability for the *Provisioning* function to indicate a preference for the *Multicast server* to compress objects carried on an object carousel. Added the ability to classify the objects carried on a given multicast transport session. Corrected references for the `@targetAcquisitionLatency` attribute in tables 10.2.3.1-1 and 10.2.5.1-1.
V1.2.1 Draft (A176r6)	August 2024	Updated references to the Digest Fields and HTTP Message Signatures Internet Drafts to their published RFCs, RFC 9530 and RFC 9421 respectively. Updated references to the MSync Internet Draft to the latest version 15 publication and aligned clauses L.2.1 and L.2.2 with this. Fixed incorrect phase 2b schema declaration in the example in clause C.1. Fixed incorrect child element type for the **`ReferencingObjectCarouselType`** in the schema in clause A.1. Extensibility mechanism added to the XML schema for the multicast gateway configuration and the multicast server configuration in clause 10.2.1, annex A and C. Designated new baseline multicast session configuration schema as version 2 in clause A.0. Corrected examples in annex C. Updated copyright year. Corrections and clarifications to service reporting. Clarified unicast repair URL handling in clause 9.2.2.