Stream: Independent Submission
RFC: 9223
Category: Informational
Published: April 2022
ISSN: 2070-1721
Authors: W. Zia
Qualcomm CDMA Technologies GmbH
T. Stockhammer
Qualcomm CDMA Technologies GmbH
L. Chaponniere
Qualcomm Technologies Inc.
G. Mandyam
Qualcomm Technologies Inc.
M. Luby
BitRipple, Inc.

RFC 9223
Real-Time Transport Object Delivery over Unidirectional Transport (ROUTE)

Abstract

The Real-time Transport Object delivery over Unidirectional Transport (ROUTE) protocol is specified for robust delivery of Application Objects, including Application Objects with real-time delivery constraints, to receivers over a unidirectional transport. Application Objects consist of data that has meaning to applications that use the ROUTE protocol for delivery of data to receivers; for example, it can be a file, a Dynamic Adaptive Streaming over HTTP (DASH) or HTTP Live Streaming (HLS) segment, a WAV audio clip, etc. The ROUTE protocol also supports low-latency streaming applications.

The ROUTE protocol is suitable for unicast, broadcast, and multicast transport. Therefore, it can be run over UDP/IP, including multicast IP. The ROUTE protocol can leverage the features of the underlying protocol layer, e.g., to provide security, it can leverage IP security protocols such as IPsec.

This document specifies the ROUTE protocol such that it could be used by a variety of services for delivery of Application Objects by specifying their own profiles of this protocol (e.g., by adding or constraining some features).

This is not an IETF specification and does not have IETF consensus.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

The Real-time Transport Object delivery over Unidirectional Transport (ROUTE) protocol can be used for robust delivery of Application Objects, including Application Objects with real-time delivery constraints, to receivers over a unidirectional transport. Unidirectional transport in this document has identical meaning to that in RFC 6726 [RFC6726], i.e., transport in the direction of receiver(s) from a sender. The robustness is enabled by a built-in mechanism, e.g., signaling for loss detection, enabling loss recovery, and optionally integrating application-layer Forward Error Correction (FEC).

Application Objects consist of data that has meaning to applications that use the ROUTE protocol for delivery of data to receivers, e.g., an Application Object can be a file, an MPEG Dynamic Adaptive Streaming over HTTP (DASH) [DASH] video segment, a WAV audio clip, an MPEG Common Media Application Format (CMAF) [CMAF] addressable resource, an MPEG-4 video clip, etc.

The ROUTE protocol is designed to enable delivery of sequences of related Application Objects in a timely manner to receivers, e.g., a sequence of DASH video segments associated to a Representation or a sequence of CMAF addressable resources associated to a CMAF Track. The applications of this protocol target services enabled on media consumption devices such as smartphones, tablets, television sets, and so on. Most of these applications are real-time in the sense that they are sensitive to and rely upon such timely reception of data. The ROUTE protocol also supports chunked delivery of real-time Application Objects to enable low-latency streaming applications (similar in its properties to chunked delivery using HTTP). The protocol also enables low-latency delivery of DASH and Apple HTTP Live Streaming (HLS) content with CMAF Chunks.

Content not intended for rendering in real time as it is received (e.g., a downloaded application), a file comprising continuous or discrete media and belonging to an app-based feature, or a file containing (opaque) data to be consumed by a Digital Rights Management (DRM) system client can also be delivered by ROUTE.

The ROUTE protocol supports a caching model where Application Objects are recovered into a cache at the receiver and may be made available to applications via standard HTTP requests from the cache. Many current day applications rely on using HTTP to access content; hence, this approach enables such applications in broadcast/multicast environments.

ROUTE is aligned with File Delivery over Unidirectional Transport (FLUTE) as defined in RFC 6726 [RFC6726] as well as the extensions defined in Multimedia Broadcast/Multicast Service (MBMS) [MBMS], but it also makes use of some principles of FCAST (Object Delivery for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols) as defined in RFC 6968 [RFC6968]; for example, object metadata and the object content may be sent together in a compound object.

The alignment to FLUTE is enabled since in addition to reusing several of the basic FLUTE protocol features, as referred to by this document, certain optimizations and restrictions are added that enable optimized support for real-time delivery of media data; hence, the name of the protocol. Among others, the source ROUTE protocol enables or enhances the following functionalities:

- Real-time delivery of object-based media data
- Flexible packetization, including enabling media-aware packetization as well as transport-aware packetization of delivery objects
- Independence of Application Objects and delivery objects, i.e., a delivery object may be a part of a file or may be a group of files.

Advanced Television Systems Committee (ATSC) 3.0 specifies the ROUTE protocol integrated with an ATSC 3.0 services layer. That specification will be referred to as ATSC-ROUTE [ATSCA331] for the remainder of this document. Digital Video Broadcasting (DVB) has specified a profile of ATSC-ROUTE in DVB Adaptive Media Streaming over IP Multicast (DVB-MABR) [DVBMABR]. This document specifies the Application Object delivery aspects (delivery protocol) for such services, as the corresponding delivery protocol could be used as a reference by a variety of services by specifying profiles of ROUTE in their respective fora, e.g., by adding new optional features atop or by restricting various optional features specified in this document in a specific service

standard. Hence, in the context of this document, the aforementioned ATSC-ROUTE and DVB-MABR are the services using ROUTE. The definition of profiles by the services also have to give due consideration to compatibility issues, and some related guidelines are also provided in this document.

This document is not an IETF specification and does not have IETF consensus. It is provided here to aid the production of interoperable implementations.

1.2. Protocol Stack for ROUTE

ROUTE delivers Application Objects such as MPEG DASH or HLS segments and optionally the associated repair data, operating over UDP/IP networks, as depicted in Table 1. The session metadata signaling to realize a ROUTE session as specified in this document **MAY** be delivered out of band or in band as well. Since ROUTE delivers objects in an application cache at the receiver from where the application can access them using HTTP, an application like DASH may use its standardized unicast streaming mechanisms in conjunction with ROUTE over broadcast/multicast to augment the services.

Application (DASH and HLS segments, CMAF Chunks, etc.)
ROUTE
UDP
IP

Table 1: Protocol Layering

1.3. Data Model

The ROUTE data model is constituted by the following key concepts.

Application Object: data that has meaning to the application that uses the ROUTE protocol for delivery of data to receivers, e.g., an Application Object can be a file, a DASH video segment, a WAV audio clip, an MPEG-4 video clip, etc.

Delivery Object: an object on course of delivery to the application from the ROUTE sender to ROUTE receiver.

Transport Object: an object identified by the Transport Object Identifier (TOI) in RFC 5651 [RFC5651]. It **MAY** be either a source or a repair object, depending on if it is carried by a Source Flow or a Repair Flow, respectively.

Transport Session: a Layered Coding Transport (LCT) channel, as defined by RFC 5651 [RFC5651]. A Transport Session **SHALL** be uniquely identified by a unique Transport Session Identifier (TSI) value in the LCT header. The TSI is scoped by the IP address of the sender, and the IP address of the sender together with the TSI uniquely identify the session. Transport Sessions are a subset of a ROUTE session. For media delivery, a

Transport Session would typically carry a media component, for example, a DASH Representation. Within each Transport Session, one or more objects are carried, typically objects that are related, e.g., DASH segments associated to one Representation.

ROUTE Session: an ensemble or multiplex of one or more Transport Sessions. Each ROUTE session is associated with an IP address/port combination. A ROUTE session typically carries one or more media components of streaming media e.g., Representations associated with a DASH Media Presentation.

Source Flow: a Transport Session carrying source data. Source Flow is independent of the Repair Flow, i.e., the Source Flow **MAY** be used by a ROUTE receiver without the ROUTE Repair Flows.

Repair Flow: a Transport Session carrying repair data for one or more Source Flows.

1.4. Architecture and Scope of Specification

The scope of the ROUTE protocol is to enable robust and real-time transport of delivery objects using LCT packets. This architecture is depicted in Figure 1.

The normative aspects of the ROUTE protocol focus on the following aspects:

- The format of the LCT packets that carry the transport objects.
- The robust transport of the delivery object using a repair protocol based on Forward Error Correction (FEC).
- The definition and possible carriage of object metadata along with the delivery objects. Metadata may be conveyed in LCT packets and/or separate objects.
- The ROUTE session, LCT channel, and delivery object description provided as service metadata signaling to enable the reception of objects.
- The normative aspects (formats, semantics) of the delivery objects conveyed as a content manifest to be delivered along with the objects to optimize the performance for specific applications e.g., real-time delivery. The objects and manifest are made available to the application through an Application Object cache. The interface of this cache to the application is not specified in this document; however, it will typically be enabled by the application acting as an HTTP client and the cache as the HTTP server.

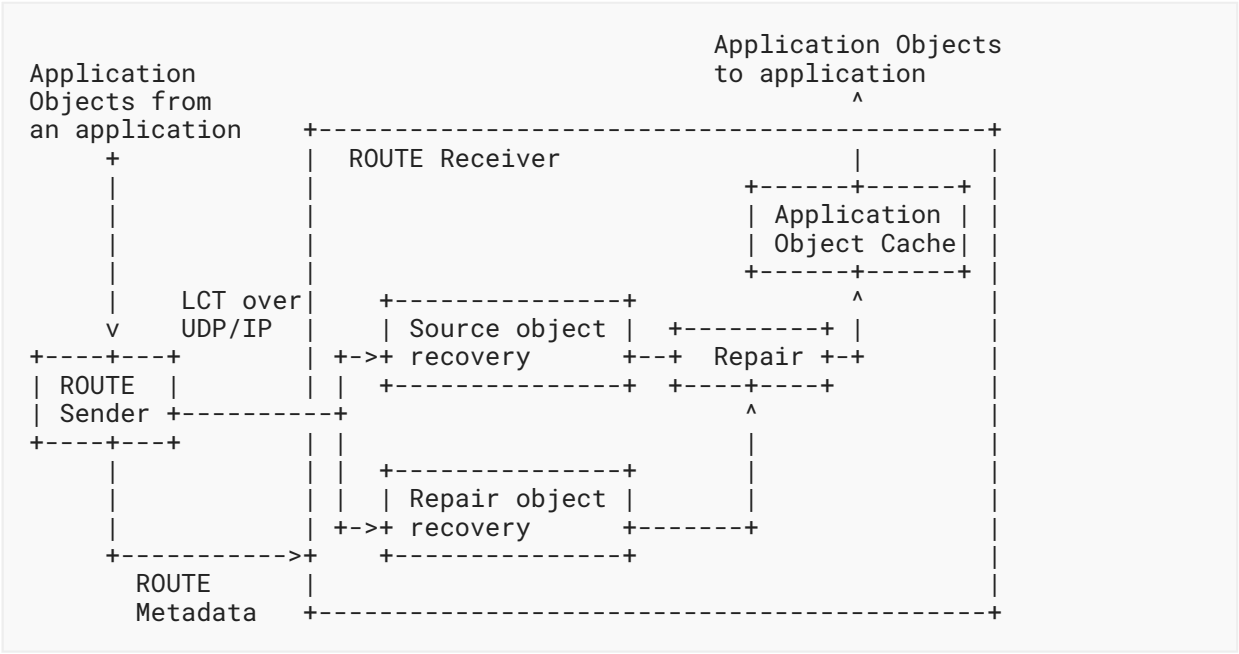

Figure 1: Architecture/Functional Block Diagram

1.5. Conventions Used in This Document

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. ROUTE Packet Format

2.1. Packet Structure and Header Fields

The packet format used by ROUTE Source Flows and Repair Flows follows the ALC packet format specified in RFC 5775 [RFC5775] with 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 2.

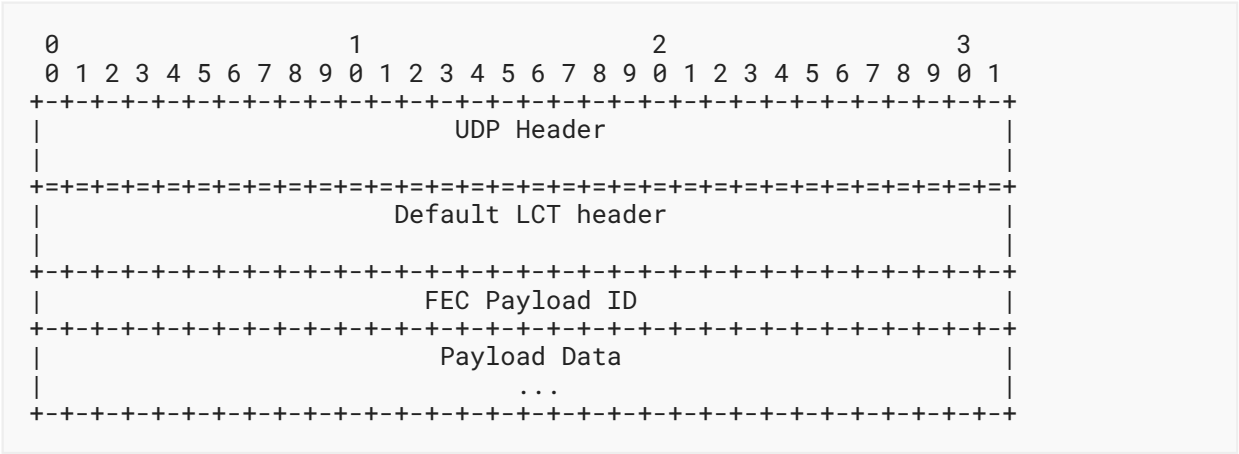

Figure 2: Overall ROUTE Packet Format

The Default LCT header is as defined in the LCT building block in RFC 5651 [RFC5651].

The LCT packet header fields **SHALL** be used as defined by the LCT building block in RFC 5651 [RFC5651]. The semantics and usage of the following LCT header fields **SHALL** be further constrained in ROUTE as follows:

Version number (V): This 4-bit field indicates the protocol version number. The version number **SHALL** be set to '0001', as specified in RFC 5651 [RFC5651].

Congestion Control flag (C) field: This 2-bit field, as defined in RFC 5651 [RFC5651], **SHALL** be set to '00'.

Protocol-Specific Indication (PSI): The most significant bit of this 2-bit flag is called the Source Packet Indicator (SPI) and indicates whether the current packet is a source packet or a FEC repair packet. The SPI **SHALL** be set to '1' to indicate a source packet and **SHALL** bet set to '0' to indicate a repair packet.

Transport Session Identifier flag (S): This 1-bit field **SHALL** be set to '1' to indicate a 32-bit word in the TSI field.

Transport Object Identifier flag (O): This 2-bit field **SHALL** be set to '01' to indicate the number of full 32-bit words in the TOI field.

Half-word flag (H): This 1-bit field **SHALL** be set to '0' to indicate that no half-word field sizes are used.

Codepoint (CP): This 8-bit field is used to indicate the type of the payload that is carried by this packet; for ROUTE, it is defined as shown below to indicate the type of delivery object carried in the payload of the associated ROUTE packet. The remaining unmapped Codepoint values can be used by a service using ROUTE. In this case, the Codepoint values **SHALL** follow the semantics specified in the following table. "IS" stands for Initialization Segment of the media

content such as the DASH Initialization Segment [DASH]. The various modes of operation in the table (File/Entity/Package Mode) are specified in Section 4. The table also lists a Codepoint value range that is reserved for future service-specific uses.

Codepoint value	Semantics
0	Reserved (not used)
1	Non Real Time (NRT) - File Mode
2	NRT - Entity Mode
3	NRT - Unsigned Package Mode
4	NRT - Signed Package Mode
5	New IS, timeline changed
6	New IS, timeline continued
7	Redundant IS
8	Media Segment, File Mode
9	Media Segment, Entity Mode
10	Media Segment, File Mode with CMAF Random Access chunk
11 - 255	Reserved, service-specific

Table 2: Codepoint Values

Congestion Control Information (CCI): For packets carrying DASH segments, CCI **MAY** convey the 32-bit earliest presentation time [DASH] of the DASH segment contained in the ROUTE packet. In this case, this information can be used by a ROUTE receiver for fast stream acquisition (details in Section 6.2). Otherwise, this field **SHALL** be set to 0.

Transport Session Identifier (TSI): This 32-bit field identifies the Transport Session in ROUTE. The context of the Transport Session is provided by signaling metadata. The value TSI = 0 **SHALL** only be used for service-specific signaling.

Transport Object Identifier (TOI): This 32-bit field **SHALL** identify the object within this session to which the payload of the current packet belongs. The mapping of the TOI field to the object is provided by the Extended File Delivery Table (FDT).

2.2. LCT Header Extensions

The following LCT header extensions are defined or used by ROUTE:

EXT_FTI: as specified in RFC 5775.

EXT_TOL: the length in bytes of the multicast transport object shall be signaled using EXT_TOL as specified by ATSC-ROUTE [ATSCA331] 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 the Close Object flag (B) [RFC5651].

NOTE: The transport object length can also be determined without the use of EXT_TOL by examining the LCT packet with the Close Object flag (B). However, if this packet is lost, then the EXT_TOL information can be used by the receiver to determine the transport object length.

EXT_TIME Header: as specified in RFC 5651 [RFC5651]. The Sender Current Time **SHALL** be signaled using EXT_TIME.

2.3. FEC Payload ID for Source Flows

The syntax of the FEC Payload ID for the Compact No-Code FEC Scheme used in ROUTE Source Flows is a 32-bit unsigned integer value that **SHALL** express the start_offset as an octet number corresponding to the first octet of the fragment of the delivery object carried in this packet. The start_offset value for the first fragment of any delivery object **SHALL** be set to 0. Figure 3 shows the 32-bit start_offset field.

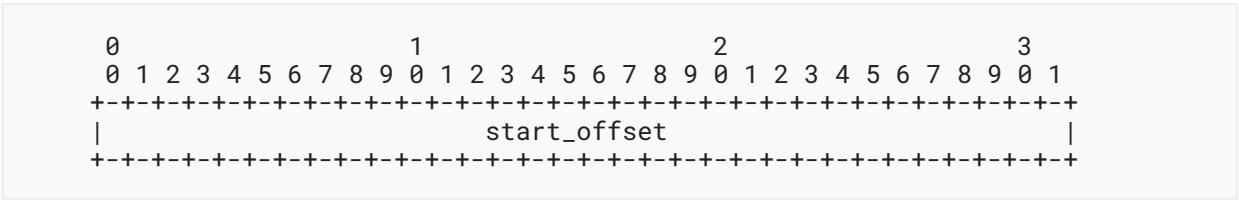

Figure 3: FEC Payload ID for Source Flows

2.4. FEC Payload ID for Repair Flows

FEC Payload ID for Repair Flows is specified in RFC 6330 [RFC6330].

3. Session Metadata

The required session metadata for Source and Repair Flows is specified in the following sections. The list specified here is not exhaustive; a service **MAY** signal more metadata to meet its needs. The data format is also not specified beyond its cardinality; the exact format of specifying the data is left for the service, e.g., by using XML encoding format, as has been done by [DVBMABR] and [ATSCA331]. It is specified in the following if an attribute is mandatory (m), conditional mandatory (cm) or optional (o) to realize a basic ROUTE session. A mandatory field **SHALL** always be present in the session metadata, and a conditional mandatory field **SHALL** be present if the specified condition is true. The delivery of the session metadata to the ROUTE receiver is beyond the scope of this document.

3.1. Generic Metadata

Generic metadata is applicable to both Source and Repair Flows as follows. Before a receiver can join a ROUTE session, the receiver needs to obtain this generic metadata that contains at least the following information:

ROUTE version number (m): the version number of ROUTE used in this session. The version number conforming to this document **SHALL** be 1.

Connection ID (m): the unique identifier of a Connection, usually consisting of the following 4-tuple: source IP address/source port number, destination IP address/destination port number. The IP addresses can be IPv4 or IPv6 addresses depending upon which IP version is used by the deployment.

3.2. Session Metadata for Source Flows

stsi (m): The LCT TSI value corresponding to the Transport Session for the Source Flow.

rt (o): A Boolean flag that **SHALL** indicate whether the content component carried by this Source Flow corresponds to real-time streaming media or non-real-time content. When set to "true", it **SHALL** be an indication of real-time content, and when absent or set to "false", it **SHALL** be an indication of non-real-time (NRT) content.

minBufferSize (o): A 32-bit unsigned integer that **SHALL** represent, in kilobytes, the minimum required storage size of the receiver transport buffer for the parent LCT channel of this Source Flow. The buffer holds the data belonging to a source object until its complete reception. This attribute is only applicable when rt = "true".

A service that chooses not to signal this attribute relies on the receiver implementation, which must discard the received data beyond its buffering capability. Such discarding of data will impact the service quality.

EFDT (cm): When present, **SHALL** contain a single instance of an FDT-Instance element per RFC 6726 FLUTE [RFC6726], which **MAY** contain the optional FDT extensions as defined in Section 4.1. The optional EFDT element **MAY** only be present for File Mode of delivery. In File Mode, it **SHALL** be present if this Source Flow transports streaming media segments.

contentType (o): A string that **SHALL** represent the media type for the media content. It **SHALL** obey the semantics of the Content-Type header as specified by the HTTP/1.1 protocol in RFC 7231 [RFC7231]. This document does not define any new contentType strings. In its absence, the signaling of media type for the media content is beyond the scope of this document.

applicationMapping (m): A set of identifiers that provide an application-specific mapping of the received Application Objects to the Source Flows. For example, for DASH, this would provide the mapping of a Source Flow to a specific DASH Representation from a Media Presentation Description (MPD), the latter identified by its Representation and corresponding Adaptation Set and Period IDs.

3.3. Session Metadata for Repair Flows

minBuffSize (o): A 32-bit unsigned integer whose value **SHALL** represent a required size of the receiver transport buffer for AL-FEC decoding processing. When present, this attribute **SHALL** indicate the minimum buffer size that is required to handle all associated objects that are assigned to a super-object, i.e., a delivery object formed by the concatenation of multiple FEC transport objects in order to bundle these FEC transport objects for AL-FEC protection.

A service that chooses not to signal this attribute relies on the receiver implementation, which must discard the received repair data beyond its buffering capability. Such discarding of data will impact the service quality.

fecOTI (m): A parameter consisting of the concatenation of Common and Scheme-Specific FEC Object Transmission Information (FEC OTI) as defined in Sections 3.3.2 and 3.3.3 of [RFC6330] and that corresponds to the delivery objects carried in the Source Flow to which this Repair Flow is associated, with the following qualification: the 40-bit Transfer Length (F) field may either represent the actual size of the object, or it is encoded as all zeroes. In the latter case, the FEC transport object size either is unknown or cannot be represented by this attribute. In other words, for the all-zeroes format, the delivery objects in the Source Flow correspond to streaming content, either a live Service whereby content encoding has not yet occurred at the time this session data was generated or pre-recorded streaming content whose delivery object sizes, albeit known at the time of session data generation, are variable and cannot be represented as a single value by the fecOTI attribute.

ptsi (m): TSI value(s) of each Source Flow protected by this Repair Flow.

mappingTOIx (o): Values of the constant X for use in deriving the TOI of the delivery object of each protected Source Flow from the TOI of the FEC (super-)object. The default value is "1". Multiple mappingTOIx values **MAY** be provided for each protected Source Flow depending upon the usage of FEC (super-)object.

mappingTOIy (o): The corresponding constant Y to each mappingTOIx, when present, for use in deriving the parent SourceTOI value from the above equation. The default value is "0".

4. Delivery Object Mode

ROUTE provides several different delivery object modes, and one of these modes may suit the application needs better for a given Transport Session. A delivery object is self contained for the application, typically associated with certain properties, metadata, and timing-related information relevant to the application. The signaling of the delivery object mode is done on an object basis using Codepoint as specified in Section 2.1.

4.1. File Mode

File Mode uses an out-of-band Extended FDT (EFDT) signaling for recovery of delivery objects with the following extensions and considerations.

4.1.1. Extensions to FDT

The following extensions are specified to FDT, as specified in RFC 6726 [RFC6726]. An Extended FDT-Instance is an instance of FLUTE FDT, as specified in [RFC6726], plus optionally one or more of the following extensions:

efdtVersion: A value that **SHALL** represent the version of this Extended FDT-Instance.

maxExpiresDelta: Let "tp" represent the wall clock time at the receiver when the receiver acquires the first ROUTE packet carrying data of the object described by this Extended FDT-Instance. maxExpiresDelta, when present, **SHALL** represent a time interval that when added to "tp" **SHALL** represent the expiration time of the associated Extended FDT-Instance "te". The time interval is expressed in number of seconds. When maxExpiresDelta is not present, the expiration time of the Extended FDT-Instance **SHALL** be given by the sum of a) the value of the ERT field in the EXT_TIME LCT header extension in the first ROUTE packet carrying data of that file, and b) the current receiver time when parsing the packet header of that ROUTE packet. See Sections 5.4 and 6.3.3 on additional rules for deriving the Extended FDT-Instance expiration time. Hence, `te = tp + maxExpiresDelta`

maxTransportSize: An attribute that **SHALL** represent the maximum transport size in bytes of any delivery object described by this Extended FDT-Instance. This attribute **SHALL** be present if a) the fileTemplate is present in Extended FDT-Instance, or b) one or more File elements, if present in this Extended FDT-Instance, do not include the Transfer-Length attribute. When maxTransportSize is not present, the maximum transport size is not signaled, while other signaling such as the Transfer-Length attribute signal the exact Transfer Length of the object.

fileTemplate: A string value, which when present and in conjunction with parameter substitution, is used in deriving the Content-Location attribute for the delivery object described by this Extended FDT-Instance. It **SHALL** include the "TOI" identifier. Each identifier **MAY** be suffixed as needed by specific file names within the enclosing '$' characters following this prototype: `%0[width]d`

The width parameter is an unsigned integer that provides the minimum number of characters to be printed. If the value to be printed is shorter than this number, the result **SHALL** be padded with leading zeroes. The value is not truncated even if the result is larger. When no format tag is present, a default format tag with width=1 **SHALL** be used.

Strings other than identifiers **SHALL** only contain characters that are permitted within URIs according to RFC 3986 [RFC3986].

$$ is an escape sequence in fileTemplate value, i.e., "$$" is non-recursively replaced with a single "$".

The usage of fileTemplate is described in Sender and Receiver operations in Sections 5.4 and 6.3, respectively.

4.1.2. Constraints on Extended FDT

The Extended FDT-Instance **SHALL** conform to an FDT-Instance according to RFC 6726 [RFC6726] with the following constraints: at least one File element and the @Expires attribute **SHALL** be present.

Content encoding **MAY** be used for delivery of any file described by an FDT-Instance.File element in the Extended FDT-Instance. The content encoding defined in the present document is gzip [RFC1952]. When content encoding is used, the File@Content-Encoding and File@Content-Length attributes **SHALL** be present in the Extended FDT-Instance.

4.2. Entity Mode

For Entity Mode, the following applies:

- Delivery object metadata **SHALL** be expressed in the form of entity headers as defined in HTTP/1.1, which correspond to one or more of the representation header fields, payload header fields, and response header fields as defined in Sections 3.1, 3.3, and 7, respectively, of [RFC7231].
- The entity headers sent along with the delivery object provide all information about that multicast transport object.
- Sending a media object (if the object is chunked) in Entity Mode may result in one of the following options:
  - If the length of the chunked object is known at the sender, the ROUTE Entity Mode delivery object **MAY** 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 ----|
```

  - If the length of the chunked object is unknown at the sender when starting to send the object, HTTP/1.1 chunked transfer coding format **SHALL** be used:

```
|HTTP Header||Separator+Length||---chunk ----
||Separator+Length||---chunk ----||Separator+Length||---chunk
----||Separator+Length||---chunk ----||Separator+Length=0|
```

    Note, however, that it is not required to send a CMAF Chunk in exactly one HTTP chunk.

4.3. Unsigned Package Mode

In this delivery mode, the delivery object consists of a group of files that are packaged for delivery only. If applied, the client is expected to unpack the package and provide each file as an independent object to the application. Packaging is supported by Multipart Multipurpose Internet Mail Extensions (MIME) [RFC2557], where objects are packaged into one document for transport, with Content-Type set to multipart/related. When binary files are included in the package, Content-Transfer-Encoding of "binary" should be used for those files.

4.4. Signed Package Mode

In Signed Package Mode delivery, the delivery object consists of a group of files that are packaged for delivery, and the package includes one or more signatures for validation. Signed packaging is supported by RFC 8551 Secure MIME (S/MIME) [RFC8551], where objects are packaged into one document for transport and the package includes objects necessary for validation of the package.

5. Sender Operation

5.1. Usage of ALC and LCT for Source Flow

ROUTE Source Flow carries the source data as specified in RFC 5775 [RFC5775]. There are several special considerations that ROUTE introduces to the usage of the LCT building block as outlined in the following:

- ROUTE limits the usage of the LCT building block to a single channel per session. Congestion control is thus sender driven in ROUTE. It also signifies that there is no specific congestion-control-related signaling from the sender to the receiver; the CCI field is either set to 0 or used for other purposes as specified in Section 2.1. The functionality of receiver-driven layered multicast may still be offered by the application, allowing the receiver application to select the appropriate delivery session based on the bandwidth requirement of that session.

Further, the following details apply to LCT:

- The Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [RFC5651] is used with the following constraints:
  - The TSI in the LCT header **SHALL** be set equal to the value of the stsi attribute in Section 3.2.
  - The Codepoint (CP) in the LCT header **SHALL** be used to signal the applied formatting as defined in the signaling metadata.

- In accordance with ALC, a source FEC Payload ID header is used to identify, for FEC purposes, the encoding symbols of the delivery object, or a portion thereof, carried by the associated ROUTE packet. This information may be sent in several ways:
  - As a simple new null FEC scheme with the following usage:
    - The value of the source FEC Payload ID header **SHALL** be set to 0 in case the ROUTE packet contains the entire delivery object, or
    - The value of the source FEC Payload ID header **SHALL** be set as a direct address (start offset) corresponding to the starting byte position of the portion of the object carried in this packet using a 32-bit field.
  - In a compatible manner to RFC 6330 [RFC6330] where the SBN and ESI defines the start offset together with the symbol size T.
  - The signaling metadata provides the appropriate parameters to indicate any of the above modes using the srcFecPayloadId attribute.

- The LCT Header EXT_TIME extension as defined in RFC 5651 [RFC5651] **MAY** be used by the sender in the following manner:
  - The Sender Current Time (SCT), depending on the application, **MAY** be used to occasionally or frequently signal the sender current time possibly for reliever time synchronization.
  - The Expected Residual Time (ERT) **MAY** be used to indicate the expected remaining time for transmission of the current object in order to optimize detection of a lost delivery object.
  - The Sender Last Changed (SLC) flag is typically not utilized but **MAY** be used to indicate the addition/removal of Segments.

Additional extension headers **MAY** be used to support real-time delivery. Such extension headers are defined in Section 2.1.

5.2. ROUTE Packetization for Source Flow

The following description of the ROUTE sender operation on the mapping of the Application Object to the ROUTE packet payloads logically represents an extension of RFC 5445 [RFC5445], which in turn inherits the context, language, declarations, and restrictions of the FEC building block in RFC 5052 [RFC5052].

The data carried in the payload of a given ROUTE packet constitutes a contiguous portion of the Application Object. ROUTE source delivery can be considered as a special case of the use of the Compact No-Code Scheme associated with FEC Encoding ID = 0 according to Sections 3.4.1 and 3.4.2 of [RFC5445], in which the encoding symbol size is exactly one byte. As specified in Section 2.1, for ROUTE Source Flows, the FEC Payload ID **SHALL** deliver the 32-bit start_offset. All receivers are expected to support, at minimum, operation with this special case of the Compact No-Code FEC.

Note that in the event the source object size is greater than 2^{32} bytes (approximately 4.3 GB), the applications (in the broadcaster server and the receiver) are expected to perform segmentation/reassembly using methods beyond the scope of this document.

Finally, in some special cases, a ROUTE sender **MAY** need to produce ROUTE packets that do not contain any payload. This may be required, for example, to signal the end of a session. These dataless packets do not contain FEC Payload ID or payload data, but only the LCT header fields. The total datagram length, conveyed by outer protocol headers (e.g., the IP or UDP header), enables receivers to detect the absence of the LCT header, FEC Payload ID, and payload data.

5.2.1. Basic ROUTE Packetization

In the basic operation, it is assumed that the Application Object is fully available at the ROUTE sender.

1. 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 Application Object being sent, whichever is smaller. The transfer unit is determined either by knowledge of underlying transport block sizes or by other constraints.
2. The start_offset field in the LCT header of the ROUTE packet indicates the byte offset of the carried data in the Application Object being sent.
3. The Close Object flag (B) is set to 1 if this is the last ROUTE packet carrying the data of the Application Object.

The order of packet delivery is arbitrary, but in the absence of other constraints, delivery with increasing start_offset value is recommended.

5.2.2. ROUTE Packetization for CMAF Chunked Content

The following additional guidelines should be followed for ROUTE packetization of CMAF Chunked Content in addition to the guidelines of Section 5.2.1:

1. If it is the first ROUTE packet carrying a CMAF Random Access chunk, except for the first CMAF Chunk in the segment, the Codepoint value **MAY** be set to 10, as specified in the Codepoint value table in Section 2.1. The receiver **MAY** use this information for optimization of random access.
2. As soon as the total length of the media object is known, potentially with the packaging of the last CMAF Chunk of a segment, the EXT_TOL extension header **MAY** be added to the LCT header to signal the Transfer Length, so that the receiver may know this information in a timely fashion.

5.3. Timing of Packet Emission

The sender **SHALL** use the timing information provided by the application to time the emission of packets for a timely reception. This information may be contained in the Application Objects e.g., DASH segments and/or the presentation manifest. Hence, such packets of streaming media with real-time constraints **SHALL** be sent in such a way as to enable their timely reception with respect to the presentation timeline.

5.4. Extended FDT Encoding for File Mode Sending

For File Mode sending:

- The TOI field in the ROUTE packet header **SHALL** be set such that Content-Location can be derived at the receiver according to File Template substitution specified in Section 6.3.1.
- After sending the first packet with a given TOI value, none of the packets pertaining to this TOI **SHALL** be sent later than the wall clock time as derived from maxExpiresDelta. The EXT_TIME header with Expected Residual Time (ERT) **MAY** be used in order to convey more accurate expiry time.

5.5. FEC Framework Considerations

The FEC framework uses concepts of the FECFRAME work as defined in RFC 6363 [RFC6363], as well as the FEC building block, RFC 5052 [RFC5052], which is adopted in the existing FLUTE/ALC/LCT specifications.

The FEC design adheres to the following principles:

- FEC-related information is provided only where needed.
- Receivers not capable of this framework can ignore repair packets.
- The FEC is symbol based with fixed symbol size per protected Source Flow. The ALC protocol and existing FEC schemes are reused.
- A FEC Repair Flow provides protection of delivery objects from one or more Source Flows.

The FEC-specific components of the FEC framework are:

- FEC Repair Flow declaration including all FEC-specific information.
- A FEC transport object that is the concatenation of a delivery object, padding octets, and size information in order to form a chunk of data that has a size in symbols of N, where N >= 1.
- A FEC super-object that is the concatenation of one or more FEC transport objects in order to bundle FEC transport objects for FEC protection.
- A FEC protocol and packet structure.

A receiver needs to be able to recover delivery objects from repair packets based on available FEC information.

5.6. FEC Transport Object Construction

In order to identify a delivery object in the context of the repair protocol, the following information is needed:

- TSI and TOI of the delivery object. In this case, the FEC object corresponds to the (entire) delivery object.
- Octet range of the delivery object, i.e., start offset within the delivery object and number of subsequent and contiguous octets of delivery object that constitutes the FEC object (i.e., the

FEC-protected portion of the source object). In this case, the FEC object corresponds to a contiguous byte range portion of the delivery object.

Typically, for real-time object delivery with smaller delivery object sizes, the first mapping is applied, i.e., the delivery object is a FEC object.

Assuming that the FEC object is the delivery object, for each delivery object, the associated FEC transport object is comprised of the concatenation of the delivery object, padding octets (P), and the FEC object size (F) in octets, where F is carried in a 4-octet field.

The FEC transport object size S, in FEC encoding symbols, **SHALL** be an integer multiple of the symbol size Y. S is determined from the session information and/or the repair packet headers.

F is carried in the last 4 octets of the FEC transport object. Specifically, let:

- F be the size of the delivery object in octets,
- F' be the F octets of data of the delivery object,
- f' denote the four octets of data carrying the value of F in network octet order (high-order octet first),
- S be the size of the FEC transport object with S=ceil((F+4)/Y), where the ceil() function rounds the result upward to its nearest integer,
- P' be S*Y-4-F octets of data, i.e., padding placed between the delivery object and the 4-byte field conveying the value of F and located at the end of the FEC transport object, and
- O' be the concatenation of F', P', and f'.

O' then constitutes the FEC transport object of size S*Y octets. Note that padding octets and the object size F are not sent in source packets of the delivery object but are only part of a FEC transport object that FEC decoding recovers in order to extract the FEC object and thus the delivery object or portion of the delivery object that constitutes the FEC object. In the above context, the FEC transport object size in symbols is S.

The general information about a FEC transport object that is conveyed to a FEC-enabled receiver is the source TSI, source TOI, and the associated octet range within the delivery object comprising the associated FEC object. However, as the size in octets of the FEC object is provided in the appended field within the FEC transport object, the remaining information can be conveyed as:

- The TSI and TOI of the delivery object from which the FEC object associated with the FEC transport object is generated
- The start octet within the delivery object for the associated FEC object
- The size in symbols of the FEC transport object, S

5.7. Super-Object Construction

From the FEC Repair Flow declaration, the construction of a FEC super-object as the concatenation of one or more FEC transport objects can be determined. The FEC super-object includes the general information about the FEC transport objects as described in the previous sections, as well as the placement order of FEC transport objects within the FEC super-object.

Let:

- N be the total number of FEC transport objects for the FEC super-object construction.
- For i = 0, ..., N-1, let S[i] be the size in symbols of FEC transport object i.
- B' be the FEC super-object that is the concatenation of the FEC transport objects in numerical order, comprised of K = Sum of N source symbols, each symbol denoted as S[i].

For each FEC super-object, the remaining general information that needs to be conveyed to a FEC-enabled receiver, beyond what is already carried in the FEC transport objects that constitute the FEC super-object, comprises:

- The total number of FEC transport objects N.
- For each FEC transport object:
  - The TSI and TOI of the delivery object from which the FEC object associated with the FEC transport object is generated,
  - The start octet within the delivery object for the associated FEC object, and
  - The size in symbols of the FEC transport object.

The carriage of the FEC repair information is discussed below.

5.8. Repair Packet Considerations

The repair protocol is based on Asynchronous Layered Coding (ALC) as defined in RFC 5775 [RFC5775] and the Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [RFC5651] with the following details:

- The Layered Coding Transport (LCT) Building Block as defined in RFC 5651 [RFC5651] is used as defined in Asynchronous Layered Coding (ALC), Section 2.1. In addition, the following constraint applies:
  - The TSI in the LCT header **SHALL** identify the Repair Flow to which this packet applies by the matching the value of the ptsi attribute in the signaling metadata among the LCT channels carrying Repair Flows.
- The FEC building block is used according to RFC 6330 [RFC6330], but only repair packets are delivered.
  - Each repair packet within the scope of the Repair Flow (as indicated by the TSI field in the LCT header) **SHALL** carry the repair symbols for a corresponding FEC transport object/

super-object as identified by its TOI. The repair object/super- object TOI **SHALL** be unique for each FEC super-object that is created within the scope of the TSI.

5.9. Summary FEC Information

For each super-object (identified by a unique TOI within a Repair Flow that is in turn identified by the TSI in the LCT header) that is generated, the following information needs to be communicated to the receiver:

- The FEC configuration consisting of:
  - FEC Object Transmission Information (OTI) per RFC 5052 [RFC5052].
  - Additional FEC information (see Section 3.3).
  - The total number of FEC objects included in the FEC super-object, N.
- For each FEC transport object:
  - TSI and TOI of the delivery object used to generate the FEC object associated with the FEC transport object,
  - The start octet within the delivery object of the associated FEC object, if applicable, and
  - The size in symbols of the FEC transport object, S.

The above information is delivered:

- Statically in the session metadata as defined in Section 3.3, and
- Dynamically in an LCT extension header.

6. Receiver Operation

The receiver receives packets and filters those packets according to the following. From the ROUTE session and each contained LCT channel, the receiver regenerates delivery objects from the ROUTE session and each contained LCT channel.

In the event that the receiver receives data that does not conform to the ROUTE protocol specified in this document, the receiver **SHOULD** attempt to recover gracefully by e.g., informing the application about the issues using means beyond the scope of this document. The ROUTE packetization specified in Section 5.2.1 implies that the receiver **SHALL NOT** receive overlapping data; if such a condition is encountered at the receiver, the packet **SHALL** be assumed to be corrupted.

The basic receiver operation is provided below (it assumes an error-free scenario), while repair considerations are provided in Section 7.

6.1. Basic Application Object Recovery for Source Flows

Upon receipt of each ROUTE packet of a Source Flow, the receiver proceeds with the following steps in the order listed.

1) The ROUTE receiver is expected to parse the LCT and FEC Payload ID to verify that it is a valid header. If it is not valid, then the payload is discarded without further processing.
2) All ROUTE packets used to recover a specific delivery object carry the same TOI value in the LCT header.
3) The ROUTE receiver is expected to assert that the TSI and the Codepoint represent valid operation points in the signaling metadata, i.e., the signaling contains a matching entry to the TSI value provided in the packet header, as well as for this TSI, and the Codepoint field in the LCT header has a valid Codepoint mapping.
4) The ROUTE receiver should process the remainder of the payload, including the appropriate interpretation of the other payload header fields, using the source FEC Payload ID (to determine the start_offset) and the payload data to reconstruct the corresponding object as follows:

    a. For File Mode, upon receipt of the first ROUTE packet payload for an object, the ROUTE receiver uses the File@Transfer-Length attribute of the associated Extended FDT-Instance, when present, to determine the length T of the object. When the File@Transfer-Length attribute is not present in the Extended FDT-Instance, the receiver uses the maxTransportSize attribute of the associated Extended FDT-Instance to determine the maximum length T' of the object. Alternatively, and specifically for delivery modes other than File Mode, the EXT_TOL header can be used to determine the length T of the object.
    b. The ROUTE receiver allocates buffer space for the T or T' bytes that the object will or may occupy.
    c. The ROUTE receiver computes the length of the payload, Y, by subtracting the payload header length from the total length of the received payload.
    d. The ROUTE receiver allocates a Boolean array RECEIVED[0..T-1] or RECEIVED[0..T'-1], as appropriate, with all entries initialized to false to track received object symbols. The ROUTE receiver continuously acquires packet payloads for the object as long as all of the following conditions are satisfied:

        i. there is at least one entry in RECEIVED still set to false,
        ii. the object has not yet expired, and
        iii. the application has not given up on reception of this object.

        More details are provided below.

    e. For each received ROUTE packet payload for the object (including the first payload), the steps to be taken to help recover the object are as follows:

        i. If the packet includes an EXT_TOL or EXT_FTI header, modify the Boolean array RECEIVED[0..T'-1] to become RECEIVED[0..T-1].
        ii. Let X be the value of the start_offset field in the ROUTE packet header and let Y be the length of the payload, Y, computed by subtracting the LCT header size and the FEC Payload ID size from the total length of the received packet.
        iii. The ROUTE receiver copies the data into the appropriate place within the space reserved for the object and sets RECEIVED[X ... X+Y-1] = true.

iv. If all T entries of RECEIVED are true, then the receiver has recovered the entire object.

Upon recovery of both the complete set of packet payloads for the delivery object associated with a given TOI value, and the metadata for that delivery object, the reception of the delivery object, now a fully received Application Object, is complete.

Given the timely reception of ROUTE packets belonging to an Application Object, the receiver **SHALL** make the Application Objects available to the application in a timely fashion using the application-provided timing data (e.g., the timing data signaled via the presentation manifest file). For example, HTTP/1.1 chunked transfer may need to be enabled to transfer the Application Objects if MPD@availabilityTimeOffset is signaled in the DASH presentation manifest in order to allow for the timely sending of segment data to the application.

6.2. Fast Stream Acquisition

When the receiver 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; 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 receiver checks for the first received packet with the Codepoint value set to 10, indicating the start of a CMAF Random Access chunk.
- The receiver **MAY** make the partially received object (a partial DASH segment starting from the packet above) available to the application for fast stream acquisition.
- It **MAY** recover the earliest presentation time of this CMAF Random Access chunk from the ROUTE packet LCT Congestion Control Information (CCI) field as specified in Section 2.1 to be able to add a new Period element in the MPD exposed to the application containing just the partially received DASH segment with period continuity signaling.

6.3. Generating Extended FDT-Instance for File Mode

An Extended FDT-Instance conforming to RFC 6726 [RFC6726], is produced at the receiver using the service metadata and in-band signaling in the following steps:

6.3.1. File Template Substitution for Content-Location Derivation

The Content-Location element of the Extended FDT for a specific Application Object is derived as follows:

"TOI" is substituted with the unique TOI value in the LCT header of the ROUTE packets used to recover the given delivery object (as specified in Section 6.1).

After the substitution, the fileTemplate **SHALL** be a valid URL corresponding to the Content-Location attribute of the associated Application Object.

An example @fileTemplate using a width of 5 is: fileTemplate="myVideo$TOI%05d$.mps", resulting in file names with exactly five digits in the number portion. The Media Segment file name for TOI=33 using this template is myVideo00033.mps.

6.3.2. File@Transfer-Length Derivation

Either the EXT_FTI header (per RFC 5775 [RFC5775]) or the EXT_TOL header, when present, is used to derive the Transport Object Length (TOL) of the File. If the File@Transfer-Length parameter in the Extended FDT-Instance is not present, then the EXT_TOL header or the or EXT_FTI header **SHALL** be present. Note that a header containing the transport object length (EXT_TOL or EXT_FTI) need not be present in each packet header. If the broadcaster does not know the length of the transport object at the beginning of the transfer, an EXT_TOL or EXT_FTI header **SHALL** be included in at least the last packet of the file and should be included in the last few packets of the transfer.

6.3.3. FDT-Instance@Expires Derivation

When present, the maxExpiresDelta attribute **SHALL** be used to generate the value of the FDT-Instance@Expires attribute. The receiver is expected to add this value to its wall clock time when acquiring the first ROUTE packet carrying the data of a given delivery object to obtain the value for @Expires.

When maxExpiresDelta is not present, the EXT_TIME header with Expected Residual Time (ERT) **SHALL** be used to derive the expiry time of the Extended FDT-Instance. When both maxExpiresDelta and the ERT of EXT_TIME are present, the smaller of the two values should be used as the incremental time interval to be added to the receiver's current time to generate the effective value for @Expires. When neither maxExpiresDelta nor the ERT field of the EXT_TIME header is present, then the expiration time of the Extended FDT-Instance is given by its @Expires attribute.

7. FEC Application

7.1. General FEC Application Guidelines

It is up to the receiver to decide to use zero, one, or more of the FEC streams. Hence, the application assigns a recovery property to each flow, which defines aspects such as the delay and the required memory if one or the other is chosen. The receiver **MAY** decide whether or not to utilize Repair Flows based on the following considerations:

- The desired start-up and end-to-end latency. If a Repair Flow requires a significant amount of buffering time to be effective, such Repair Flow might only be used in time-shift operations or in poor reception conditions, since use of such Repair Flow trades off end-to-end latency against DASH Media Presentation quality.
- FEC capabilities, i.e., the receiver **MAY** pick only the FEC algorithm that it supports.
- Which Source Flows are being protected; for example, if the Repair Flow protects Source Flows that are not selected by the receiver, then the receiver may not select the Repair Flow.
- Other considerations such as available buffer size, reception conditions, etc.

If a receiver decides to acquire a certain Repair Flow, then the receiver must receive data on all Source Flows that are protected by that Repair Flow to collect the relevant packets.

7.2. TOI Mapping

When mappingTOIx/mappingTOIy are used to signal X and Y values, the TOI value(s) of the one or more source objects (sourceTOI) protected by a given FEC transport object or FEC super-object with a TOI value rTOI is derived through an equation sourceTOI = X*rTOI + Y.

When neither mappingTOIx nor mappingTOIy is present, there is a 1:1 relationship between each delivery object carried in the Source Flow as identified by ptsi to a FEC object carried in this Repair Flow. In this case, the TOI of each of those delivery objects **SHALL** be identical to the TOI of the corresponding FEC object.

7.3. Delivery Object Reception Timeout

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

- The latest time that the file repair procedure may start is bound by the @Expires attribute of the FDT-Instance.
- The receiver may choose to start the file repair procedure earlier if it detects the occurrence of any of the following events:
  - Presence of the Close Object flag (B) in the LCT header [RFC5651] for the file of interest;
  - Presence of the Close Session flag (A) in the LCT header [RFC5651] before the nominal expiration of the Extended FDT-Instance as defined by the @Expires attribute.

7.4. Example FEC Operation

To be able to recover the delivery objects that are protected by a Repair Flow, a receiver needs to obtain the necessary Service signaling metadata fragments that describe the corresponding collection of delivery objects that are covered by this Repair Flow. A Repair Flow is characterized by the combination of an LCT channel, a unique TSI number, as well as the corresponding protected Source Flows.

If a receiver acquires data of a Repair Flow, the receiver is expected to collect all packets of all protected Transport Sessions. Upon receipt of each packet, whether it is a source or repair packet, the receiver proceeds with the following steps in the order listed.

1. The receiver is expected to parse the packet header and verify that it is a valid header. If it is not valid, then the packet **SHALL** be discarded without further processing.
2. The receiver is expected to parse the TSI field of the packet header and verify that a matching value exists in the Service signaling for the Repair Flow or the associated Protected Source Flow. If no match is found, the packet **SHALL** be discarded without further processing.

3. The receiver processes the remainder of the packet, including interpretation of the other header fields, and using the source FEC Payload ID (to determine the start_offset byte position within the source object), the Repair FEC Payload ID, as well as the payload data, reconstructs the decoding blocks corresponding to a FEC super-object as follows:
   a. For a source packet, the receiver identifies the delivery object to which the received packet is associated using the session information and the TOI carried in the payload header. Similarly, for a repair object, the receiver identifies the FEC super-object to which the received packet is associated using the session information and the TOI carried in the payload header.
   b. For source packets, the receiver collects the data for each FEC super-object and recovers FEC super-objects in the same way as a Source Flow in Section 6.1. The received FEC super-object is then mapped to a source block and the corresponding encoding symbols are generated.
   c. With the reception of the repair packets, the FEC super-object can be recovered.
   d. Once the FEC super-object is recovered, the individual delivery objects can be extracted.

8. Considerations for Defining ROUTE Profiles

Services (e.g., ATSC-ROUTE [ATSCA331], DVB-MABR [DVBMABR], etc.) may define specific ROUTE "profiles" based on this document in their respective standards organizations. An example is noted in the overview section: DVB has specified a profile of ATSC-ROUTE in DVB Adaptive Media Streaming over IP Multicast (DVB-MABR) [DVBMABR]. The definition has the following considerations. Services **MAY**

- Restrict the signaling of certain values signaled in the LCT header and/or provision unused fields in the LCT header.
- Restrict using certain LCT header extensions and/or add new LCT header extensions.
- Restrict or limit usage of some Codepoints and/or assign semantics to service-specific Codepoints marked as reserved in this document.
- Restrict usage of certain Service signaling attributes and/or add their own service metadata.

Services **SHALL NOT** redefine the semantics of any of the ROUTE attributes in LCT headers and extensions, as well as Service signaling attributes already specified in this document.

By following these guidelines, services can define profiles that are interoperable.

9. ROUTE Concepts

9.1. ROUTE Modes of Delivery

Different ROUTE delivery modes specified in Section 4 are optimized for delivery of different types of media data. For example, File Mode is specifically optimized for delivering DASH content using Segment Template with number substitution. Using File Template in EFDT avoids the need for the repeated sending of metadata as outlined in the following section. Same optimizations,

however, cannot be used for time substitution and segment timeline where the addressing of each segment is time dependent and in general does not follow a fixed or repeated pattern. In this case, Entity Mode is more optimized since it carries the file location in band. Also, Entity Mode can be used to deliver a file or part of the file using HTTP Partial Content response headers.

9.2. File Mode Optimizations

In File Mode, the delivery object represents an Application Object. This mode replicates FLUTE as defined in RFC 6726 [RFC6726] but with the ability to send static and pre-known file metadata out of band.

In FLUTE, FDT-Instances are delivered in band and need to be generated and delivered in real time if objects are generated in real time at the sender. These FDT-Instances have some differences as compared to the FDT specified in Section 3.4.2 of [RFC6726] and Section 7.2.10 of MBMS [MBMS]. The key difference is that besides separated delivery of file metadata from the delivery object it describes, the FDT functionality in ROUTE may be extended by additional file metadata and rules that enable the receiver to generate the Content-Location attribute of the File element of the FDT, on the fly. This is done by using information in both the extensions to the FDT and the LCT header. The combination of pre-delivery of static file metadata and receiver self generation of dynamic file metadata avoids the necessity of continuously sending the FDT-Instances for real-time objects. Such modified FDT functionality in ROUTE is referred to as the Extended FDT.

9.3. In-Band Signaling of Object Transfer Length

As an extension to FLUTE, ROUTE allows for using EXT_TOL LCT header extension with 24 bits or, if required, 48 bits to signal the Transfer Length directly within the ROUTE packet.

The transport object length can also be determined without the use of EXT_TOL by examining the LCT packet with the Close Object flag (B). However, if this packet is lost, then the EXT_TOL information can be used by the receiver to determine the transport object length.

Applications using ROUTE for delivery of low-latency streaming content may make use of this feature for sender-end latency optimizations: the sender does not have to wait for the completion of the packaging of a whole Application Object to find its Transfer Length to be included in the FDT before the sending can start. Rather, partially encoded data can already be started to be sent via the ROUTE sender. As the time approaches when the encoding of the Application Object is nearing completion, and the length of the object becomes known (e.g., the time of writing the last CMAF Chunk of a DASH segment), only then the sender can signal the object length using the EXT TOL LCT header. For example, for a 2-second DASH segment with 100-millisecond chunks, it may result in saving up to 1.9 second latency at the sending end.

9.4. Repair Protocol Concepts

The ROUTE repair protocol is FEC-based and is enabled as an additional layer between the transport layer (e.g., UDP) and the object delivery layer protocol. The FEC reuses concepts of the FEC Framework defined in RFC 6363 [RFC6363], but in contrast to the FEC Framework in RFC

6363 [RFC6363], the ROUTE repair protocol does not protect packets but instead protects delivery objects as delivered in the source protocol. In addition, as an extension to FLUTE, it supports the protection of multiple objects in one source block which is in alignment with the FEC Framework as defined in RFC 6363 [RFC6363]. Each FEC source block may consist of parts of a delivery object, as a single delivery object (similar to FLUTE) or multiple delivery objects that are bundled prior to FEC protection. ROUTE FEC makes use of FEC schemes in a similar way as those defined in RFC 5052 [RFC5052] and uses the terminology of that document. The FEC scheme defines the FEC encoding and decoding as well as the protocol fields and procedures used to identify packet payload data in the context of the FEC scheme.

In ROUTE, all packets are LCT packets as defined in RFC 5651 [RFC5651]. Source and repair packets may be distinguished by:

- Different ROUTE sessions, i.e., they are carried on different UDP/IP port combinations.
- Different LCT channels, i.e., they use different TSI values in the LCT header.
- The most significant PSI bit in the LCT, if carried in the same LCT channel. This mode of operation is mostly suitable for FLUTE-compatible deployments.

10. Interoperability Chart

As noted in prevision sections, ATSC-ROUTE [ATSCA331] and DVB-MABR [DVBMABR] are considered services using this document that constrain specific features as well as add new ones. In this context, the following table is an informative comparison of the interoperability of ROUTE as specified in this document with ATSC-ROUTE [ATSCA331] and DVB-MABR [DVBMABR]:

Element	ATSC-ROUTE	This Document	DVB-MABR
LCT header field	PSI LSB set to 0 for Source Flow	Not defined	Set to 1 for Source Flow for CMAF Random Access chunk
	CCI may be set to 0	CCI may be set to EPT for Source Flow	
LCT header extensions	EXT_ROUTE_ PRESENTATION_TIME Header used for Media Delivery Event (MDE) mode	Not defined; may be added by a profile.	Shall not be used.
	EXT_TIME Header linked to MDE mode in Annex A. 3.7.2 [ATSCA331]	EXT_TIME Header may be used regardless (for FDT-Instance@Expires calculation)	

Element	ATSC-ROUTE	This Document	DVB-MABR
Codepoints	Full set	Does not specify range 11 - 255 (leaves to profiles)	Restricted to 5 - 9
Session metadata	Full set	Only defines a small subset of data necessary for setting up Source and Repair Flows. Does not define format or encoding of data except if data is integral/ alphanumerical. Leaves rest to profiles.	Reuses A/331 metadata, duplicated from its own Service signaling.
Extended FDT	Instance shall not be sent with Source Flow	Not restricted, may be restricted by a profile.	Instance shall not be sent with Source Flow
	No restriction	Only allowed in File Mode	
Delivery Object Mode	File, Entity, Signed/unsigned package		Signed/ unsigned package not allowed
Sender operation: Packetization	Defined for DASH segment	Defined for DASH segment and CMAF Chunks	
Receiver object recovery	Object handed to application upon complete reception	Object may be handed before completion if MPD@availabilityTimeOffset signaled	
	-	Fast Stream acquisition guidelines provided	

Table 3: Interoperability Chart

11. Security and Privacy Considerations

11.1. Security Considerations

As noted in Section 9, ROUTE is aligned with FLUTE as specified in RFC 6726 [RFC6726] and only diverges in certain signaling optimizations, especially for the real-time object delivery case. Hence, most of the security considerations documented in RFC 6726 [RFC6726] for the data flow

itself, the session metadata (session control parameters in RFC 6726 [RFC6726]), and the associated building blocks apply directly to ROUTE as elaborated in the following along with some additional considerations.

Both encryption and integrity protection applied either on file or packet level, as recommended in the file corruption considerations of RFC 6726 [RFC6726], **SHOULD** be used for ROUTE. Additionally, RFC 3740 [RFC3740] documents multicast security architecture in great detail with clear security recommendations that **SHOULD** be followed.

When ROUTE is carried over UDP and a reverse channel from receiver to sender is available, the security mechanisms provided in RFC 9147 [RFC9147] **SHOULD** be applied.

In regard to considerations for attacks against session description, this document does not specify the semantics or mechanism of delivery of session metadata, though the same threats apply for service using ROUTE as well. Hence, a service using ROUTE **SHOULD** take these threats into consideration and address them appropriately following the guidelines provided by RFC 6726 [RFC6726]. Additionally, to the recommendations of RFC 6726 [RFC6726], for Internet connected devices, services **SHOULD** enable clients to access the session description information using HTTPS with customary authentication/authorization, instead of sending this data via multicast/broadcast, since considerable security work has been done already in this unicast domain, which can enable highly secure access of session description data. Accessing via unicast, however, will have different privacy considerations, noted in Section 11.2. Note that in general the multicast/broadcast stream is delayed with respect to the unicast stream. Therefore, the session description protocol **SHOULD** be time synchronized with the broadcast stream, particularly if the session description contains security-related information.

In regard to FDT, there is one key difference for File Mode when using File Template in EFDT, which avoids repeated sending of FDT-Instances and hence, the corresponding threats noted in RFC 6726 [RFC6726] do not apply directly to ROUTE in this case. The threat, however, is shifted to the ALC/LCT headers, since they carry the additional signaling that enables determining Content-Location and File@Transfer-Length in this case. Hence, integrity protection recommendations of ALC/LCT header **SHOULD** be considered with higher emphasis in this case for ROUTE.

Finally, attacks against the congestion control building block for the case of ROUTE can impact the optional fast stream acquisition specified in Section 6.2. Receivers **SHOULD** have robustness against timestamp values that are suspicious, e.g., by comparing the signaled time in the LCT headers with the approximate time signaled by the MPD, and **SHOULD** discard outlying values. Additionally, receivers **MUST** adhere to the expiry timelines as specified in Section 6. Integrity protection mechanisms documented in RFC 6726 [RFC6726] **SHOULD** be used to address this threat.

11.2. Privacy Considerations

Encryption mechanisms recommended for security considerations in Section 11.1 **SHOULD** also be applied to enable privacy and protection from snooping attacks.

Since this protocol is primarily targeted for IP multicast/broadcast environments where the end user is mostly listening, identity protection and user data retention considerations are more protected than in the unicast case. Best practices for enabling privacy on IP multicast/broadcast **SHOULD** be applied by the operators, e.g., "Recommendations for DNS Privacy Service Operators" in RFC 8932 [RFC8932].

However, if clients access session description information via HTTPS, the same privacy considerations and solutions **SHALL** apply to this access as for regular HTTPS communication, an area that is very well studied and the concepts of which are being integrated directly into newer transport protocols such as IETF QUIC [RFC9000] enabling HTTP/3 [HTTP3]. Hence, such newer protocols **SHOULD** be used to foster privacy.

Note that streaming services **MAY** contain content that may only be accessed via DRM (digital rights management) systems. DRM systems can prevent unauthorized access to content delivered via ROUTE.

12. IANA Considerations

This document has no IANA actions.

13. References

13.1. Normative References

[ATSCA331] Advanced Television Systems Committee, "Signaling, Delivery, Synchronization, and Error Protection", ATSC Standard A/331:2022-03, March 2022.

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, DOI 10.17487/RFC1952, May 1996, <https://www.rfc-editor.org/info/rfc1952>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2557] Palme, J., Hopmann, A., and N. Shelness, "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", RFC 2557, DOI 10.17487/RFC2557, March 1999, <https://www.rfc-editor.org/info/rfc2557>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <https://www.rfc-editor.org/info/rfc5052>.

[RFC5445] Watson, M., "Basic Forward Error Correction (FEC) Schemes", RFC 5445, DOI 10.17487/RFC5445, March 2009, <https://www.rfc-editor.org/info/rfc5445>.

[RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", RFC 5651, DOI 10.17487/RFC5651, October 2009, <https://www.rfc-editor.org/info/rfc5651>.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, DOI 10.17487/RFC5775, April 2010, <https://www.rfc-editor.org/info/rfc5775>.

[RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <https://www.rfc-editor.org/info/rfc6330>.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <https://www.rfc-editor.org/info/rfc6363>.

[RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, DOI 10.17487/RFC6726, November 2012, <https://www.rfc-editor.org/info/rfc6726>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/info/rfc7231>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

13.2. Informative References

[CMAF] International Organization for Standardization, "Information technology -- Multimedia application format (MPEG-A) -- Part 19: Common media application format (CMAF) for segmented media", First edition, ISO/IEC FDIS 23000-19, January 2018, <https://www.iso.org/standard/71975.html>.

[DASH] International Organization for Standardization, "Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats", Fourth edition, ISO/IEC 23009-1:2019, December 2019, <https://www.iso.org/standard/79329.html>.

[DVBMABR] ETSI, "Digital Video Broadcasting (DVB); Adaptive media streaming over IP multicast", version 1.1.1, ETSI TS 103 769, November 2020.

[HTTP3] Bishop, M., Ed., "Hypertext Transfer Protocol Version 3 (HTTP/3)", Work in Progress, Internet-Draft, draft-ietf-quic-http-34, 2 February 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34>.

[MBMS] ETSI, "Universal Mobile Telecommunications Systems (UMTS); LTE; 5G; Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs", version 16.9.1, ETSI TS 126 346, May 2021.

[RFC3740] Hardjono, T. and B. Weis, "The Multicast Group Security Architecture", RFC 3740, DOI 10.17487/RFC3740, March 2004, <https://www.rfc-editor.org/info/rfc3740>.

[RFC6968] Roca, V. and B. Adamson, "FCAST: Object Delivery for the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 6968, DOI 10.17487/RFC6968, July 2013, <https://www.rfc-editor.org/info/rfc6968>.

[RFC8932] Dickinson, S., Overeinder, B., van Rijswijk-Deij, R., and A. Mankin, "Recommendations for DNS Privacy Service Operators", BCP 232, RFC 8932, DOI 10.17487/RFC8932, October 2020, <https://www.rfc-editor.org/info/rfc8932>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

Acknowledgments

As outlined in the introduction and in ROUTE concepts in Section 9, the concepts specified in this document are the culmination of the collaborative work of several experts and organizations over the years. The authors would especially like to acknowledge the work and efforts of the following people and organizations to help realize the technologies described in this document (in no specific order): Mike Luby, Kent Walker, Charles Lo, and other colleagues from Qualcomm Incorporated, LG Electronics, Nomor Research, Sony, and BBC R&D.

Authors' Addresses

Waqar Zia
Qualcomm CDMA Technologies GmbH
Anzinger Str. 13
81671 Munich
Germany
Email: wzia@qti.qualcomm.com

Thomas Stockhammer
Qualcomm CDMA Technologies GmbH
Anzinger Str. 13
81671 Munich
Germany
Email: tsto@qti.qualcomm.com

Lenaig Chaponniere
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, CA 92121
United States of America
Email: lguellec@qti.qualcomm.com

Giridhar Mandyam
Qualcomm Technologies Inc.
5775 Morehouse Drive
San Diego, CA 92121
United States of America
Email: mandyam@qti.qualcomm.com

Michael Luby
BitRipple, Inc.
1133 Miller Ave
Berkeley, CA 94708
United States of America
Email: luby@bitripple.com