# ETSI TS 126 346 V6.1.0 (2005-06)

*Technical Specification*

**Universal Mobile Telecommunications System (UMTS);
Multimedia Broadcast/Multicast Service (MBMS);
Protocols and codecs
(3GPP TS 26.346 version 6.1.0 Release 6)**

Reference

RTS/TSGS-0426346v610

Keywords

UMTS

# Intellectual Property Rights

# Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp .

# Contents

Intellectual Property Rights ....2

Foreword ....2

Foreword ....8

Introduction ....8

1 Scope ....9

2 References ....9

3 Definitions and abbreviations ....12
3.1 Definitions ....12
3.2 Abbreviations ....13

4 MBMS system description ....14
4.1 MBMS functional layers ....14
4.2 MBMS User Service Entities ....14
4.3 MBMS bearer service architecture ....15
4.4 Functional Entities to support MBMS User Services ....15
4.4.1 MBMS User Service Architecture ....15
4.4.1a Content Provider / Multicast Broadcast Source ....18
4.4.2 MBMS Key Management Function ....18
4.4.3 MBMS Session and Transmission Function ....18
4.4.4 User Service Discovery / Announcement function ....18
4.4.5 Interactive Announcement Function ....18
4.4.6 MBMS UE ....19
4.5 Usage of identity of MBMS session ....19

5 Procedures and protocol ....20
5.1 Introduction ....20
5.2 User Service Discovery/Announcement ....20
5.2.1 Introduction ....20
5.2.2 MBMS User Service Description metadata fragments ....20
5.2.2.1 Session Description ....21
5.2.2.2 Associated Delivery Procedure Description ....22
5.2.2.3 Service Protection Description ....22
5.2.2.4 XML-Schema for MBMS User Service Bundle Description ....23
5.2.2.5 Example MBMS User Service Description Instances ....25
5.2.2.6 FEC Repair Stream Description ....25
5.2.3 User service announcement over a MBMS bearer ....26
5.2.3.1 Supported Metadata Syntaxes ....26
5.2.3.2 Consistency Control and Syntax Independence ....26
5.2.3.3 Metadata Envelope Definition ....26
5.2.3.4 Delivery of the Metadata Envelope ....27
5.2.3.5 Metadata Envelope Transport ....28
5.2.3.6 Metadata Envelope and Metadata Fragment Association with FLUTE ....28
5.2.4 User service announcement using Interactive Announcement Function ....28
5.2.4.1.1 Supported Metadata Syntaxes ....28
5.2.4.1.2 Consistency Control and Syntax Independence ....29
5.2.4.1.3 Metadata Envelope Definition ....29
5.2.4.1.4 Delivery of the Metadata Envelope ....29
5.2.4.1.5 Metadata Envelope Transport ....29
5.2.4.1.6 User service announcement over SMS bearers ....29
5.2.4.1.7 User service announcement over HTTP push bearers ....29
5.2.5 Metadata fragment encapsulation to aggregate Service Announcement documents ....29
5.3 User Service Initiation/Termination ....30
5.3.1 Initiation ....30
5.3.2 MBMS User Service termination procedure ....30

5.4 MBMS Data Transfer Procedure ....31
5.5 MBMS Protocols ....32

6 Introduction on Delivery Methods ....32

7 Download Delivery Method ....33
7.1 Introduction ....33
7.2 FLUTE usage for MBMS download ....33
7.2.1 Fragmentation of Files ....34
7.2.2 Symbol Encoding Algorithm ....34
7.2.3 Blocking Algorithm ....34
7.2.4 Congestion Control ....34
7.2.5 Content Encoding of Files for Transport ....34
7.2.6 Transport File Grouping ....34
7.2.7 Signalling of Parameters with Basic ALC/FLUTE Headers ....35
7.2.8 Signalling of Parameters with FLUTE Extension Headers ....35
7.2.9 Signalling of Parameters with FDT Instances ....35
7.2.10 FDT Schema ....36
7.2.11 MBMS Session Identify ....37
7.2.12 FEC Scheme definition ....38
7.2.12.1 General ....38
7.2.12.2 FEC payload ID ....38
7.2.12.3 FEC Object Transmission Information ....38
7.3 SDP for Download Delivery Method ....39
7.3.1 Introduction ....39
7.3.2 SDP Parameters for MBMS download session ....39
7.3.2.1 Sender IP address ....39
7.3.2.2 Number of channels ....39
7.3.2.3 Destination IP address and port number for channels ....40
7.3.2.4 Transport Session Identifier (TSI) of the session ....40
7.3.2.5 Multiple objects transport indication ....40
7.3.2.6 Session Timing Parameters ....40
7.3.2.7 Mode of MBMS bearer per media ....40
7.3.2.8 FEC capabilities and related parameters ....41
7.3.2.9 Service-language(s) per media ....41
7.3.2.10 Bandwidth Specification ....41
7.3.3 SDP Examples for FLUTE Session ....42

8 Streaming delivery method ....42
8.1 Introduction ....42
8.2 Transport protocol ....42
8.2.1 RTP payload formats for media ....42
8.2.2 FEC mechanism for RTP ....43
8.2.2.1 Sending Terminal Operation (Informative) ....44
8.2.2.2 Receiving Terminal Operation (Informative) ....45
8.2.2.3 (Void) ....46
8.2.2.4 Packet format for FEC source packets ....46
8.2.2.5 Packet Format for Repair packets ....46
8.2.2.6 Structure of the FEC source block ....47
8.2.2.7 FEC block Construction algorithm and example (informative) ....47
8.2.2.8 MBMS FEC scheme definition ....48
8.2.2.9 Source FEC Payload ID ....48
8.2.2.10 Repair FEC payload ID ....49
8.2.2.10a FEC Object Transmission information ....49
8.2.2.11 Hypothetical FEC Decoder ....49
8.2.2.12 FEC encoding procedures ....50
8.2.2.13 Signalling ....50
8.2.2.13a SDP for FEC source packet streams ....51
8.2.2.14 SDP for FEC repair packet streams ....51
8.2.2.15 Signalling example for FEC ....51
8.3 Session description ....52
8.3.1 SDP Parameters for MBMS streaming session ....53
8.3.1.1 Sender IP address ....53

8.3.1.2 Destination IP address and port number for channels ....53
8.3.1.3 Media Description ....53
8.3.1.4 Session Timing Parameters ....54
8.3.1.5 Mode of MBMS bearer per media ....54
8.3.1.6 Service-language(s) per media ....54
8.3.1.7 Bandwidth specification ....54
8.3.1.8 FEC Parameters ....54
8.3.1.9 FEC Flow ID attribute ....55
8.3.2 SDP Example for Streaming Session ....55
8.3.2.1 SDP Description for QoE Metrics ....56
8.4 Quality of Experience ....56
8.4.1 General ....56
8.4.2 QoE Metrics ....56
8.4.2.1 Corruption duration metric ....57
8.4.2.2 Rebuffering duration metric ....57
8.4.2.3 Initial buffering duration metric ....58
8.4.2.4 Successive loss of RTP packets ....58
8.4.2.5 Frame rate deviation ....58
8.4.2.6 Jitter duration ....58
8.4.3 Example metrics initiation with SDP ....59

9 Associated delivery procedures ....59
9.1 Introduction ....59
9.2 Associated Procedure Description ....60
9.3 File Repair Procedure ....60
9.3.1 Introduction ....60
9.3.2 Identification of End of Transmission for MBMS Download Delivery ....61
9.3.3 Identification of Missing Data from an MBMS Download ....62
9.3.4 Back-off Timing the Procedure Initiation Messaging for Scalability ....62
9.3.4.1 Offset time ....62
9.3.4.2 Random Time Period ....62
9.3.4.3 Back-off Time ....63
9.3.4.4 Reset of the Back-off Timer ....63
9.3.5 File Repair Server Selection ....63
9.3.5.1 List of Server URIs ....63
9.3.5.2 Selection from the Server URI List ....63
9.3.6 File Repair Request Message ....63
9.3.6.1 File Repair Request Message Format ....63
9.3.7 File Repair Response Message ....65
9.3.7.1 File Repair Response Messages Codes ....65
9.3.7.2 File Repair Response Message Format for HTTP Carriage of Repair Data ....66
9.3.7.3 File Repair Response for Broadcast/Multicast of Repair Data ....67
9.3.8 Server Not Responding Error Case ....67
9.4 The Reception Reporting Procedure ....68
9.4.1 Identifying Complete File Reception from MBMS Download ....68
9.4.2 Identifying Complete MBMS Delivery Session Reception ....68
9.4.3 Determining Whether a Reception Report Is Required ....69
9.4.4 Request Time Selection ....69
9.4.5 Reception Report Server Selection ....70
9.4.6 Reception Report Message ....70
9.4.7 Reception Report Response Message ....71
9.5 XML-Schema for Associated Delivery Procedures ....71
9.5.1 Generic Associated Delivery Procedure Description ....71
9.5.2 Example Associated Delivery Procedure Description Instance ....71
9.5.3 XML Syntax for a Reception Report Request ....72
9.5.3.1 Use of Specific Values ....72
9.5.3.2 Example XML for the Reception Report Request ....72

10 Media codecs and formats ....73
10.1 General ....73
10.2 Speech ....73
10.3 Audio ....73

10.4 Synthetic audio ....73
10.5 Video ....74
10.6 Still images ....74
10.7 Bitmap graphics ....74
10.8 Vector graphics ....75
10.9 Text ....75
10.10 Timed text ....75
10.11 3GPP file format ....75

**Annex A (normative): FLUTE Support Requirements ....76**

**Annex B (normative): FEC encoder and decoder specification ....78**

B.1 Definitions, Symbols and Abbreviations ....78
B.1.1 Definitions ....78
B.1.2 Symbols ....79
B.1.3 Abbreviations ....80

B.2 Overview ....80

B.3 File download ....80
B.3.1 Source block construction ....80
B.3.1.1 General ....80
B.3.1.2 Source block and sub-block partitioning ....81
B.3.2 Encoding packet construction ....82
B.3.2.1 General ....82
B.3.2.2 Encoding packet construction ....82
B.3.3 Transport ....82
B.3.4 Example parameters ....83
B.3.4.1 Parameter derivation algorithm ....83
B.3.4.2 Examples ....84

B.4 Streaming ....84
B.4.1 Source block construction ....84
B.4.2 Encoding packet construction ....84
B.4.3 Transport ....85
B.4.4 Example parameters ....85
B.4.4.1 Parameter derivation algorithm ....85
B.4.4.2 Examples ....86

B.5 Systematic Raptor encoder ....86
B.5.1 Encoding overview ....86
B.5.2 First encoding step: Intermediate symbol Generation ....87
B.5.2.1 General ....87
B.5.2.2 Source symbol triples ....87
B.5.2.3 Pre-coding relationships ....87
B.5.2.4 Intermediate symbols ....88
B.5.2.4.1 Definition ....88
B.5.2.4.2 Example method for calculation of intermediate symbols ....88
B.5.3 Second encoding step: LT encoding ....89
B.5.4 Generators ....90
B.5.4.1 Random Generator ....90
B.5.4.2 Degree Generator ....90
B.5.4.3 LT Encoding Symbol Generator ....90
B.5.4.4 Triple generator ....91

B.6 Systematic Indices J($K$)) ....92

B.7 Random Numbers ....93
B.7.1 The table $V_0$ ....93
B.7.2 The table $V_1$ ....93

B.8 Example FEC decoder ....93
B.8.1 General ....93
B.8.2 Decoding a source block ....93

B.8.2.1 General......93
B.8.2.2 First Phase......94
B.8.2.3 Second Phase......95
B.8.2.4 Third Phase......95
B.8.2.5 Fourth Phase......96

**Annex C (informative): IANA registration......97**

C.1 Registration of SDP Protocol Identifiers for Source packet......97

C.2 Registration of SDP Protocol identifier for repair packets......97

C.3 Registration of MIME type "application/simpleSymbolContainer"......97

C.4 Registration of MIME type "application/mbms-user-service-description-parameter"......97

C.5 Registration of MIME type "application/mbms-envelope"......98

C.6 Registration of MIME type "application/mbms-protection-description"......98

C.7 Registration of MIME type "application/mbms-associated-procedure-parameter"......98

C.8 Registration of MIME type "application/vnd.3gpp.mbms-msk+xml"......98

C.9 Registration of MIME type "application/vnd.3gpp.mbms-register+xml"......98

C.10 Registration of MIME type "application/vnd.3gpp.mbms-deregister+xml"......98

**Annex D (informative): RTP packetization guidelines......99**

**Annex E (informative): Change history......100**

History......101

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

# Introduction

MBMS is a point-to-multipoint service in which data is transmitted from a single source entity to multiple recipients. Transmitting the same data to multiple recipients allows network resources to be shared.

The MBMS bearer service offers two modes:

- Broadcast Mode.
- Multicast Mode.

MBMS user services can be built on top of the MBMS bearer service. The present document specifies two delivery methods for the MBMS user services: download and streaming. Examples of applications using the download delivery method are news and software upgrades. Delivery of live music is an example of an application using the streaming delivery method.

There can be several MBMS user services. The objective of the present document is the definition of a set of media codecs, formats and transport/application protocols to enable the deployment of MBMS user services. The present document takes into consideration the need to maximize the reuse of components of already specified services like PSS and MMS.

# 1 Scope

The present document defines a set of media codecs, formats and transport/application protocols to enable the deployment of MBMS user services over the MBMS bearer service within the 3GPP system.

In this version of the specification, only MBMS download and streaming delivery methods are specified. The present document does not preclude the use of other delivery methods.

The present document includes information applicable to network operators, service providers and manufacturers.

# 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 22.146: "Multimedia Broadcast/Multicast Service; Stage 1".

[3] 3GPP TS 22.246: "Multimedia Broadcast/Multicast Service (MBMS) user services; Stage 1".

[4] 3GPP TS 23.246: "Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description".

[5] 3GPP TS 25.346: "Introduction of Multimedia Broadcast/Multicast Service (MBMS) in the Radio Access Network (RAN); Stage 2".

[6] IETF STD 0064/RFC 3550 (July 2003): "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson.

[7] IETF STD 0006/RFC 0768 (August 1980): "User Datagram Protocol", J. Postel.

[8] IETF STD 0005/RFC 0791 (September 1981): "Internet Protocol", J. Postel.

[9] IETF RFC 3926 (October 2004): "FLUTE - File Delivery over Unidirectional Transport", T. Paila, M. Luby, R. Lehtonen, V. Roca, R. Walsh.

[10] IETF RFC 3450 (December 2002): "Asynchronous Layered Coding (ALC) Protocol Instantiation", M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft.

[11] IETF RFC 3451 (December 2002): "Layered Coding Transport (LCT) Building Block", M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, J. Crowcroft.

[12] IETF RFC 3452 (December 2002): "Forward Error Correction (FEC) Building Block", M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, J. Crowcroft.

[13] IETF RFC 3695 (February 2004): "Compact Forward Error Correction (FEC) Schemes", M. Luby, L. Vicisano.

[14] IETF RFC 2327 (April 1998): "SDP: Session Description Protocol", M. Handley and V. Jacobson.

| | |
|---|---|
| [15] | IETF draft-ietf-mmusic-sdp-srcfilter-06: "Session Description Protocol (SDP) Source Filters". |
| [16] | IETF RFC 3266 (June 2002): "Support for IPv6 in Session Description Protocol (SDP)", S. Olson, G. Camarillo, A. B. Roach. |
| [17] | IETF RFC 3048 (January 2001): "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby. |
| [18] | IETF RFC 2616 (June 1999): "Hypertext Transfer Protocol -- HTTP/1.1". |
| [19] | IETF STD 0066/RFC 3986 (January 2005): "Uniform Resource Identifier (URI)". |
| [20] | 3GPP TS 33.246: "3G Security; Security of Multimedia Broadcast/Multicast Service (MBMS)". |
| [21] | OMG: "Unified Modeling Language (UML), version 1.5" (formal/03-03-01). |
| [22] | W3C Recommendation 28 October 2004: "XML Schema Part 2: Datatypes Second Edition". |
| [23] | IETF RFC 2234 (November 1997): "Augmented BNF for Syntax Specifications: ABNF", D. Crocker and P. Overell. |
| [24] | 3GPP TS 26.290: "Audio codec processing functions; Extended Adaptive Multi-Rate - Wideband (AMR-WB+) codec; Transcoding functions". |
| [25] | 3GPP TS 26.304: "Floating-point ANSI-C code for the Extended Adaptive Multi-Rate - Wideband (AMR-WB+) codec". |
| [26] | 3GPP TS 26.273: "Speech codec speech processing functions; Extended Adaptive Multi-Rate - Wideband (AMR-WB+) speech codec; Fixed-point ANSI-C code". |
| [27] | Void. |
| [28] | 3GPP TS 26.401: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; General description". |
| [29] | 3GPP TS 26.410: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; Floating-point ANSI-C code". |
| [30] | 3GPP TS 26.411: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; Fixed-point ANSI-C code". |
| [31] | W3C Recommendation 04 February 2004: "Extensible Markup Language (XML) 1.1", T. Bray, J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau and J. Cowan. |
| [32] | 3GPP TS 26.244: "Transparent end-to-end streaming service; 3GPP file format (3GP)". |
| [33] | IETF RFC 3267 (June 2002): "Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs", J. Sjoberg, M. Westerlund, A. Lakaniemi, Q. Xie. |
| [34] | IETF draft-ietf-avt-rtp-amrwbplus-06 (September 2004): "RTP Payload Format for Extended AMR Wideband (AMR-WB+) Audio Codec", J. Sjoberg, M. Westerlund and A. Lakaniemi. |
| [35] | IETF RFC 3984 (February 2005): "RTP payload Format for H.264 Video", S. Wenger, M.M. Hannuksela, T. Stockhammer, M. Westerlund, D. Singer. |
| [36] | Void. |
| [37] | IETF RFC 2557 (March 1999): "MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)", J. Palme, A. Hopmann, N. Shelness. |
| [38] | IETF RFC 3890 (September 2004): "A Transport Independent Bandwidth Modifier for the Session Description Protocol (SDP)", M. Westerlund. |
| [39] | IETF RFC 3556 (July 2003): "Session Description Protocol (SDP) Bandwidth Modifiers for RTP Control Protocol (RTCP) Bandwidth", S. Casner. |

[40] 3GPP TS 24.008: "Mobile radio interface Layer 3 specification; Core network protocols; Stage 3".

[41] IETF RFC 3640 (November 2003): "RTP Payload Format for Transport of MPEG-4 Elementary Streams", J. van der Meer, D. Mackie, V. Swaminathan, D. Singer, P. Gentric.

[42] IETF RFC 1952 (May 1996): "GZIP file format specification version 4.3", P. Deutsch.

[43] ITU-T Recommendation H.264 (2003): "Advanced video coding for generic audiovisual services" | ISO/IEC 14496-10 (2003): "Information technology - Coding of audio-visual objects - Part 10: Advanced Video Coding".

[44] ISO/IEC 14496-10/FDAM1: "AVC Fidelity Range Extensions".

[45] ITU-T Recommendation H.263 (1998): "Video coding for low bit rate communication".

[46] ITU-T Recommendation H.263 - Annex X (04/01): "Annex X: Profiles and levels definition".

[47] 3GPP TS 26.234: "Transparent end-to-end streaming service; Protocols and codecs".

[48] 3GPP TS 26.071: "AMR speech codec; General description".

[49] 3GPP TS 26.090: "AMR speech codec; Transcoding functions".

[50] 3GPP TS 26.073: "AMR speech Codec; C-source code".

[51] 3GPP TS 26.104: "ANSI-C code for the floating-point Adaptive Multi-Rate (AMR) speech codec".

[52] 3GPP TS 26.171: "AMR speech codec, wideband; General description".

[53] 3GPP TS 26.190: "Mandatory Speech Codec speech processing functions AMR Wideband speech codec; Transcoding functions".

[54] 3GPP TS 26.173: "ANCI-C code for the Adaptive Multi Rate - Wideband (AMR-WB) speech codec".

[55] 3GPP TS 26.204: "ANSI-C code for the floating-point Adaptive Multi-Rate Wideband (AMR-WB) speech codec".

[56] Scalable Polyphony MIDI Specification Version 1.0, RP-34, MIDI Manufacturers Association, Los Angeles, CA, February 2002.

[57] Scalable Polyphony MIDI Device 5-to-24 Note Profile for 3GPP Version 1.0, RP-35, MIDI Manufacturers Association, Los Angeles, CA, February 2002.

[58] "Standard MIDI Files 1.0", RP-001, in "The Complete MIDI 1.0 Detailed Specification, Document Version 96.1", The MIDI Manufacturers Association, Los Angeles, CA, USA, February 1996.

[59] Mobile DLS, MMA specification v1.0, RP-41 Los Angeles, CA, USA. 2004.

[60] Mobile XMF Content Format Specification, MMA specification v1.0, RP-42, Los Angeles, CA, USA. 2004.

[61] ITU-T Recommendation T.81 (1992) | ISO/IEC 10918-1:1993: "Information technology - Digital compression and coding of continuous-tone still images - Requirements and guidelines".

[62] C-Cube Microsystems (September 1992): "JPEG File Interchange Format", Version 1.02.

[63] CompuServe Incorporated (1987): "GIF Graphics Interchange Format: A Standard defining a mechanism for the storage and transmission of raster-based graphics information", Columbus, OH, USA. See at http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/GIF87a.txt.

[64] CompuServe Incorporated (1990): "Graphics Interchange Format: Version 89a", Columbus, OH, USA.

[65] IETF RFC 2083 (March 1997): "PNG (Portable Networks Graphics) Specification Version 1.0", T. Boutell.

[66] W3C Working Draft 27 October 2004: "Scalable Vector Graphics (SVG) 1.2", http://www.w3.org/TR/2004/WD-SVG12-20041027/.

[67] W3C Working Draft 13 August 2004: "Mobile SVG Profile: SVG Tiny, Version 1.2", http://www.w3.org/TR/2004/WD-SVGMobile12-20040813/.

[68] Standard ECMA-327 (June 2001): "ECMAScript 3rd Edition Compact Profile".

[69] WAP Forum Specification (Octobeer 2001): "XHTML Mobile Profile", http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf.

[70] ISO/IEC 10646-1 (2000): "Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane".

[71] The Unicode Consortium: "The Unicode Standard", Version 3.0 Reading, MA, Addison-Wesley Developers Press, 2000, ISBN 0-201-61633-5.

[72] 3GPP TS 26.245: "Transparent end-to-end Packet switched Streaming Service (PSS); Timed text format".

[73] IETF RFC 3066: "Tags for the Identification of Languages".

[74] ISO 639: "Codes for the representation of names of languages".

[75] ISO 3661: "End-suction centrifugal pumps -- Baseplate and installation dimensions".

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

[77] 3GPP TS 23.003: 'Numbering, addressing and identification'

[78] IETF RFC 1305 (March 1992): "Network Time Protocol (Version 3) Specification, Implementation".

[79] OMA Push OTA Protocol (25-April-2001): WAP-235-PushOTA-20010425-a http://www.openmobilealliance.org/tech/affiliates/LicenseAgreement.asp?DocName=/wap/wap-235-pushota-20010425-a.pdf

[80] IETF RFC 3711 (March 2004): "The Secure Real-time Transport Protocol (SRTP)", M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman.

[81] IETF STD065/RFC 3551: "RTP Profile for Audio and Video Conferences with Minimal Control", Schulzrinne H. and Casner S., July 2003.

# 3 Definitions and abbreviations

## 3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply:

**Broadcast session:** See 3GPP TS 22.146 [2].

**Forward Error Correction (FEC):** in the context of MBMS, a FEC mechanism is used at the application layer to allow MBMS receivers to recover lost SDUs

**FLUTE channel:** equivalent to an ALC/LCT channel
An ALC/LCT channel is defined by the combination of a sender and an address associated with the channel by the sender (RFC 3926 [9]).

**Multicast joining:** See 3GPP TS 22.146 [2].

**Multicast session:** See 3GPP TS 22.146 [2].

**Multimedia Broadcast/Multicast Service (MBMS):** See 3GPP TS 22.146 [2].

**MBMS user services:** MBMS User Service may use more than one Multimedia Broadcast/Multicast Service (bearer service) and more than one Broadcast and/or Multicast session

See 3GPP TS 22.246 [3].

**MBMS user service discovery/announcement:** user service discovery refers to methods for the UE to obtain the list of available MBMS user services along with information on the user service and the user service announcement refers to methods for the MBMS service provider to make the list of available MBMS user services along with information on the user service available to the UE

**MBMS user service initiation:** UE mechanisms to setup the reception of MBMS user service data
The initiation procedure takes place after the discovery of the MBMS user service

**MBMS delivery method:** mechanism used by a MBMS user service to deliver content
An MBMS delivery method uses MBMS bearers in delivering content and may make use of associated procedures.

**MBMS download delivery method:** delivery of discrete objects (e.g. files) by means of a MBMS download session

**MBMS streaming delivery method:** delivery of continuous media (e.g. real-time video) by means of a MBMS streaming session

**MBMS download session:** time, protocols and protocol state (i.e. parameters) which define sender and receiver configuration for the download of content files

**MBMS streaming session:** time, protocols and protocol state (i.e. parameters) which define sender and receiver configuration for the streaming of content

**RTP Session**: The RTP and RTCP traffic sent to a specific IP multicast address and port pair (one port each for RTP and RTCP) during the time period the session is specified to exist. An RTP session is used to transport a single media type (e.g. audio, video, or text). An RTP session may contain several different streams of RTP packets using different SSRCs.

## 3.2 Abbreviations

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

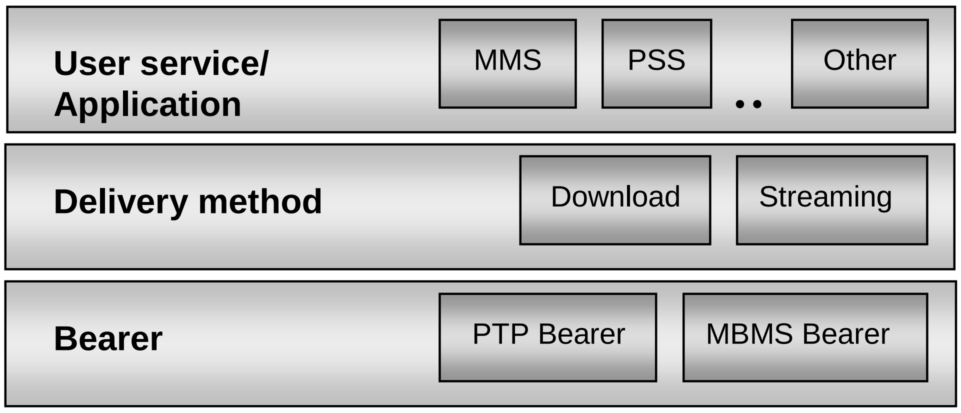
| | |
|---|---|
| ALC | Asynchronous Layered Coding |
| APN | Access Point Name |
| AVC | Advanced Video Coding |
| BM-SC | Broadcast-Multicast - Service Centre |
| CC | Congestion Control |
| ERT | Expected Residual Time |
| ESI | Encoding Symbol ID |
| FDT | File Delivery Table |
| FEC | Forward Error Correction |
| FLUTE | File deLivery over Unidirectional Transport |
| GGSN | Gateway GPRS Serving Node |
| GPRS | General Packet Radio Service |
| IP | Internet Protocol |
| LCT | Layered Coding Transport |
| MBMS | Multimedia Broadcast/Multicast Service |
| MIME | Multipurpose Internet Mail Extensions |
| MS | Mobile Station |
| MSK | MBMS Service Key |
| MTK | MBMS Traffic Key |
| MUK | MBMS User Key |
| PSS | Packet Switch Streaming |
| PTM | Point To Multipoint |
| PTP | Point To Point |
| RTP | Real-Time transport Protocol |
| SBN | Source Block Number |

| | |
|---|---|
| SCT | Sender Current Time |
| SDP | Session Description Protocol |
| TMGI | Temporary Mobile Group Identity |
| TOI | Transport Object Identifier |
| TSI | Transport Session Identifier |
| UDP | User Datagram Protocol |
| UE | User Equipment |
| URI | Uniform Resource Identifier |
| URL | Uniform Resource Locator |
| XML | eXtensible Markup Language |

# 4 MBMS system description

## 4.1 MBMS functional layers

Delivering MBMS-based services 3 distinct functional layers are identified - Bearers, Delivery method and User service. Figure 1 depicts these layers with examples of bearer types, delivery methods and applications.



| | |
|---|---|
| **Bearers:** | Bearers provide the mechanism by which IP data is transported. MBMS bearers as defined in 3GPP TS 23.246 [4] and 3GPP TS 22.146 [3] are used to transport multicast and broadcast traffic in an efficient one-to-many manner and are the foundation of MBMS-based services. MBMS bearers may be used jointly with unicast PDP contexts in offering complete service capabilities. |
| **Delivery Method:** | When delivering MBMS content to a receiving application one or more delivery methods are used. The delivery layer provides functionality such as security and key distribution, reliability control by means of forward-error-correction techniques and associated delivery procedures such as file-repair, delivery verification. Two delivery methods are defined, namely download and streaming. Delivery methods may be added beyond release 6. Delivery methods may use MBMS bearers and may make use of point-to-point bearers through a set of MBMS associated procedures. |
| **User service:** | The MBMS User service enables applications. Different application impose different requirements when delivering content to MBMS subscribers and may use different MBMS delivery methods. As an example a messaging application such as MMS would use the download delivery method while a streaming application such as PSS would use the streaming delivery method. |

**Figure 1: Functional Layers for MBMS User Service**

## 4.2 MBMS User Service Entities

Figure 2 shows the MBMS user service entities and their inter-relations. Relation cardinality is depicted as well.

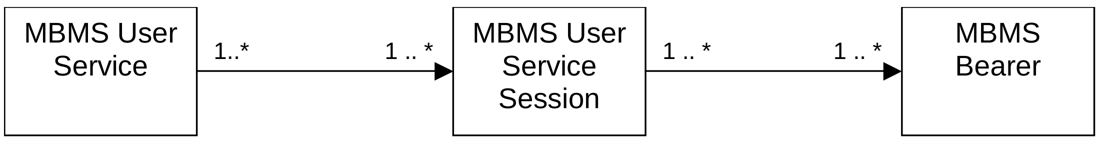


**Figure 2: Entities and Relations**

An MBMS user service is an entity that is used in presenting a complete service offering to the end-user and allowing him to activate or deactivate the service. It is typically associated with short descriptive material presented to the end-user, which would potentially be used by the user to decide whether and when to activate the offered service.

A single service entity can contain multiple distinct multimedia objects or streams, which may need to be provided over various MBMS download or MBMS streaming sessions. A download session or a streaming session is associated with its MBMS bearers and a set of delivery method parameters specifying how content is to be received on the mobile side.

A set of one or more MBMS bearers can be used for delivering data as part of an MBMS download or streaming session. As an example, the audio and visual part of video stream can be carried on separate MBMS bearers. However, it is recommendated to transfer MBMS download and/or streaming sessions, which belong to the same MBMS user service on the same MBMS bearer service.

An MBMS bearer (identified by IP multicast address and APN) might be used in providing data to more than one MBMS download or streaming session (3GPP TS 22.246 [3], clause 5).

## 4.3 MBMS bearer service architecture

The MBMS Bearer Service Architecture is defined in 3GPP TS 23.246 [4]. The MBMS User Service interfaces to the MBMS system via 3 entities.

- The BM-SC.
- The GGSN.
- The UE.

The BM-SC provides functions for MBMS user service provisioning and delivery to the content provider. It can also serve as an entry point for IP MBMS data traffic from the MBMS User Service source.

The GGSN serves as an entry point for IP multicast traffic as MBMS data from the BM-SC.

## 4.4 Functional Entities to support MBMS User Services

### 4.4.1 MBMS User Service Architecture

Figure 3 depicts the MBMS network architecture showing MBMS related entities involved in providing MBMS user services.

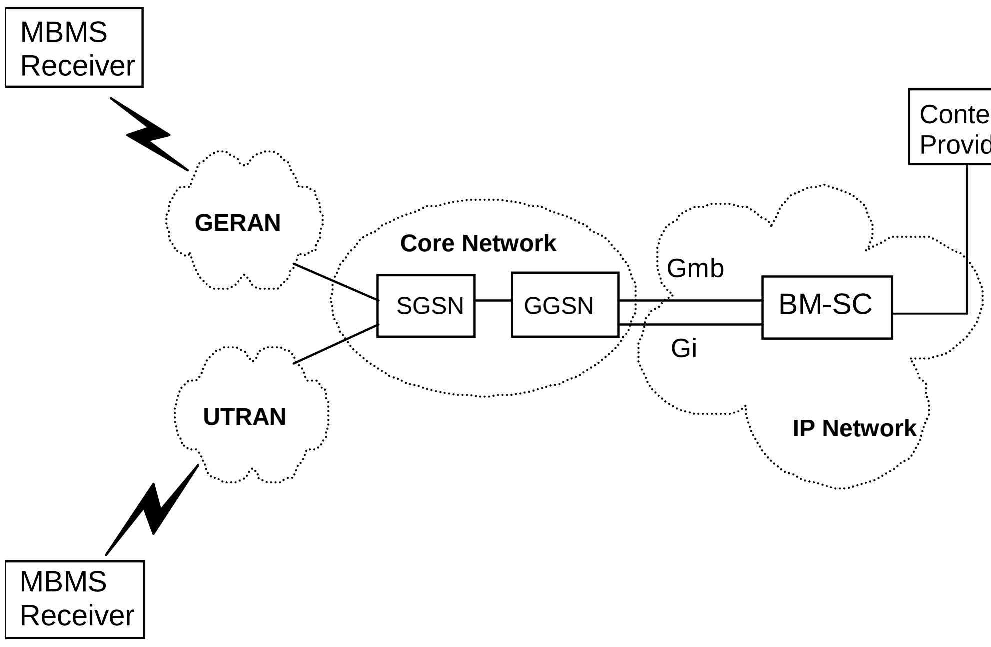


**Figure 3: MBMS network architecture model**

MBMS User Service architecture is based on an MBMS receiver on the UE side and a BM-SC on the network side.

The use of the Gmb and Gi interface in providing IP multicast traffic and managing MBMS bearer sessions is described in detailed in 3GPP TS 23.246 [4].

Details about the BM-SC functional entities are given in figure 4.

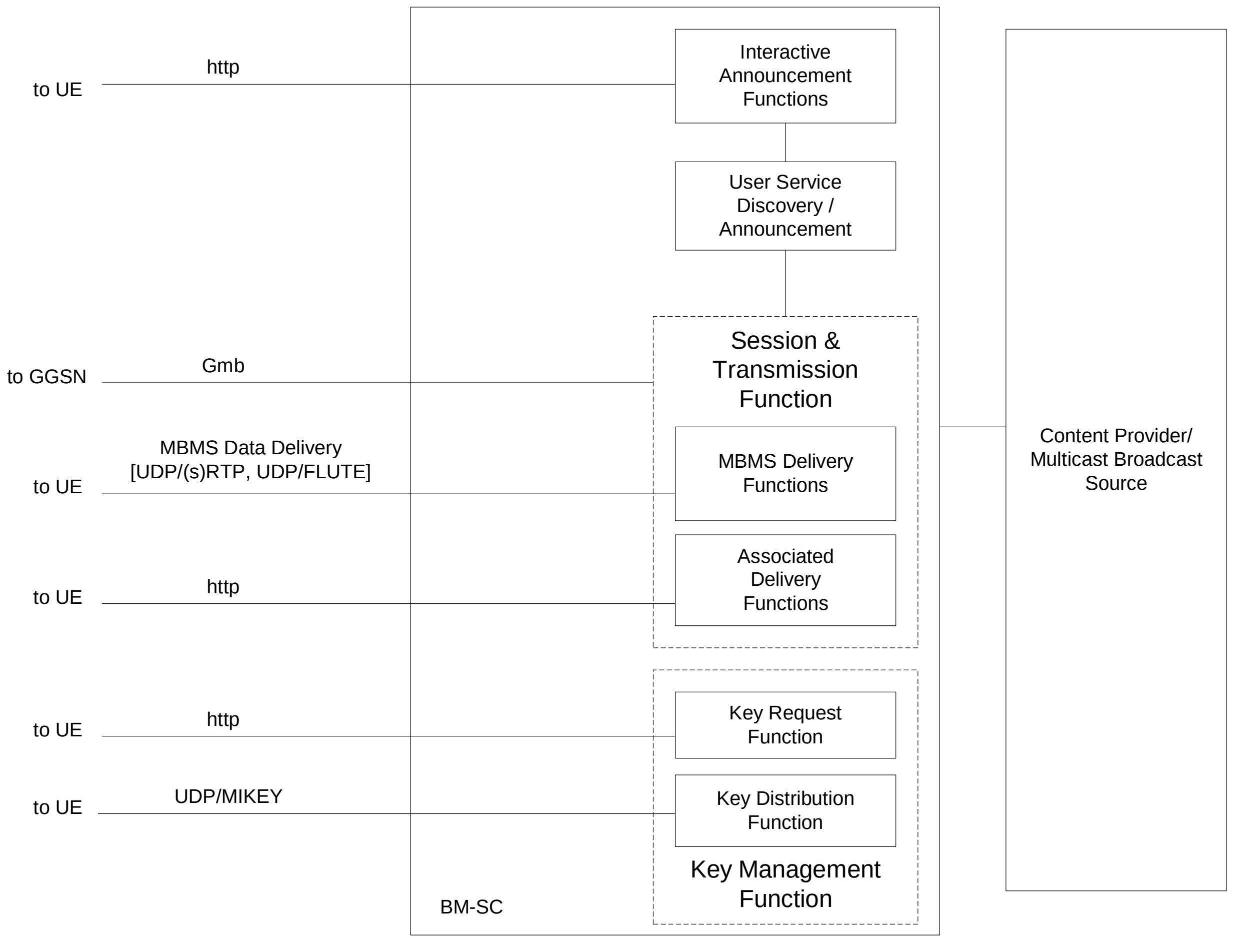

to UE
http
Interactive Announcement Functions
User Service Discovery / Announcement
to GGSN
Gmb
Session & Transmission Function
MBMS Data Delivery [UDP/(s)RTP, UDP/FLUTE]
to UE
MBMS Delivery Functions
to UE
http
Associated Delivery Functions
to UE
http
Key Request Function
to UE
UDP/MIKEY
Key Distribution Function
Key Management Function
BM-SC
Content Provider/ Multicast Broadcast Source

**Figure 4: BM-SC sub-functional structure**

The Session and Transmission function is further subdivided into the MBMS Delivery functions and the Associated Delivery functions.

The BM-SC and UE may exchange service and content related information either over point-to-point bearers or MBMS bearers whichever is suitable. To that end the following MBMS procedures are provided:

- User Service Discovery / Announcement providing service description material to be presented to the end-user as well as application parameters used in providing service content to the end-user.
- MBMS-based delivery of data/content from the BM-SC to the UE over IP multicast.
  - The data/content is optionally confidentiality and/or integrity protected
  - The data/content is optionally protected by an forward error correction code
- Key Request and Registration procedure for receiving keys and key updates.
- Key distribution procedures whereby the BM-SC distributes key material required to access service data and delivered content.
- Associated Delivery functions are invoked by the UE in relation to the MBMS data transmission. The following associated delivery functions are available:
  - File repair for download delivery method used to complement missing data.
  - Delivery verification and reception statistics collection procedures.

The interfaces between internal BM-SC functions are outside the scope of the present document.

A "Proxy and Transport function" may be located between the "Session and Transmission Function" and the GGSN. The "Proxy and Transport function" is transparent to the "Session and Transmission function". The 'Proxy and Transport' function is defined in clause 5.1.3 of [4].

### 4.4.1a Content Provider / Multicast Broadcast Source

The Content Provider/Multicast Broadcast Source may provide discrete and continuous media, as well as service descriptions and control data, to the BM-SC to offer services via MBMS broadcast- and multicast bearer services at a time. An MBMS User Service may use one or several MBMS delivery methods simultaneously. The Content Provider/Multicast Broadcast Source may also be a 3rd Party Content Provider/Multicast Broadcast Source.

The Content Provider/Multicast Broadcast Source function may reside within the operator's network or may be provided from outside the operator's network. The Content Provider/Multicast Broadcast Source can also configure the Session and Transmission functions (e.g. delivery or associated delivery). The interface between the Content Provider/Multicast Broadcast Source and the BM-SC is outside the scope of the present document.

### 4.4.2 MBMS Key Management Function

The MBMS Key Management function is used for distributing MBMS keys (Key Distribution subfunction) to authorized UEs. Before the UE can receive MBMS keys, the UE needs to register to the Key Request subfunction of the Key Management function by indicating the MBMS User Service Id. Once registered, the UE can request missing MBMS keys from the BM-SC by indicating the specific MBMS key id. In order for the UE to stop the BM-SC to send MBMS key updates a deregistration with the MBMS User Service Id is needed.

If the MBMS User Service does not require any MBMS data protection, then the UE shall not register for key management purposes.

A detailed description of all key management procedures is provided in 3GPP TS 33.246 [20].

### 4.4.3 MBMS Session and Transmission Function

The MBMS Session and Transmission function transfers the actual MBMS session data to the group of MBMS UEs. The MBMS Session and Transmission function interacts with the GGSN through the Gmb Proxy function to activate and release the MBMS transmission resources.

The function contains the MBMS delivery methods, which use the MBMS bearer service for distribution of content. Further this function contains a set of Associated-Delivery Functions, which may be invoked by the UE in relation to the MBMS data transmission (e.g. after the MBMS data transmission).

The BM-SC Session and Transmission function is further described in later clauses of the present document as well as in 3GPP TS 23.246 [4].

MBMS user services data may be integrity and/or confidentiality protected as specified within 3GPP TS 33.246 [20], and protection is applied between the BM-SC and the UE. This data protection is based on symmetric keys, which are shared between the BM-SC and the UEs accessing the service.

MBMS user services may also be protected against packet loss between BM-SC and UE using a forward error correction code.

### 4.4.4 User Service Discovery / Announcement function

The User Service Discovery / Announcement provides service description information, which may be delivered via the Session and Transmission function or via the Interactive Announcement function. This includes information, which is necessary to initiate an MBMS user service as described in clause 5.3.1. Metadata for the service descriptions are described in clause 5.2.

### 4.4.5 Interactive Announcement Function

An Interactive Announcement Function may offer alternative means to provide service descriptions to the UE using HTTP or be distributed through other interactive transport methods.

### 4.4.6 MBMS UE

The MBMS UE hosts the MBMS User Services receiver function. The MBMS receiver function may receive data from several MBMS User Services simultaneously. According to the MBMS UE capabilities, some MBMS UEs may be able to receive data, belonging to one MBMS User Service from several MBMS Bearer Services simultaneously. The MBMS receiver function uses interactive bearers for user service initiation / termination, user service discovery and associated delivery procedures.

In case the MBMS user service is secured, the UE needs one or more cryptographic MBMS service keys, therefore the UE requests the relevant cryptographic MBMS service keys using the BM-SC Key Request function. The received keys (i.e. MSK) are then used for securing the MBMS session.

## 4.5 Usage of identity of MBMS session

The Session Identity of the MBMS session is provided with the MBMS session start procedure from the BM-SC to the GGSN via the Gmb protocol in the MBMS Session Identity information element. The 'MBMS Session Identity' information element is specified in [77]. The size of the Session Identity field is 1 octet. The MBMS Session Identity is forwarded with the MBMS SESSION START REQUEST message through the system and received by the MBMS UE with the paging message.

The usage of the MBMS Session Identity is optional. The MBMS Session Identity is only applicable to MBMS download delivery sessions. The MBMS transmission resources are activated as described in clause 5.4. Each MBMS session of the MBMS User Service may be activated using a different MBMS session identifier. The MBMS UE determines, based on the MBMS Session Identity value, whether the files of the upcoming MBMS download session were already received. If the files have already been completely received, the MBMS UE does not respond to the notification of the MBMS Session.

The association of MBMS Session Identities to files is determined by the BM-SC and communicated within the File Delivery Table. This association of a MBMS Session Identity to files is valid until a particular expiry time, also signalled within the File Delivery Table. If a UE has not received a File Delivery Table associating a given MBMS Session Identity to a specific file or set of files, or a previously received association has expired, then the UE shall assume that the MBMS Session Identity value is associated to new files which has not yet been received and shall respond as normal to MBMS notifications with that Session Identity value.

A single MBMS Session Identity value may be associated with a single file or with a set of files. Once a MBMS Session Identity value has been associated with a particular file or a set of files, this association shall not be changed before the expiry of the validity time for that MBMS Session Identity value. In particular, a File Delivery Table including some files that has previously been associated with a particular Session Identity value must include all files previously associated with that value, even if it is not intended to include all the files within the MBMS transmission session.

An FDT instance includes the MBMS Session Identity expiry time and associates the MBMS Session Identity expiry times with particular MBMS Session Identity values.

If the MBMS Session Identity is used by the BM-SC, the BM-SC shall also provide the session repetition number of that MBMS transmission session on the Gmb interface.

If the BM-SC starts using the MBMS Session Identity for one MBMS Bearer Service, the BM-SC may still decide not to use the MBMS Session Identity for a later MBMS transmission on that MBMS bearer service (e.g. when an MBMS session is transmitted only once).

After determining that all files for a MBMS Session Identity value has been received, the UE shall not respond to MBMS notifications for the MBMS Bearer Service with that MBMS Session Identity value until the MBMS Session Identity is expired.

The UE shall recover its interest in a given MBMS Session Identity value when the MBMS Session Identity validity time for that Session Identity value has expired.

The BM-SC may send FDT instances on a separate transmission session or interleaved with other data packets of the same transmission session. An FDT instance may describe more files than the files to be transmitted over the same transmission session as that FDT instance.

# 5 Procedures and protocol

## 5.1 Introduction

This clause specifies the MBMS User service procedures and protocols.

## 5.2 User Service Discovery/Announcement

### 5.2.1 Introduction

User service discovery refers to methods for the UE to obtain a list of available MBMS user services or user service bundles along with information on the user services. Part of the information may be presented to the user to enable service selection.

User service announcement refers to methods for the MBMS service provider to announce the list of available MBMS user services and user service bundles, along with information on the user services, to the UE.

In order for the user to be able to initiate a particular service, the UE needs certain metadata information. The required metadata information is described in clause 5.2.2.

According to 3GPP TS 23.246 [4], in order for this information to be available to the UE operators/service providers may consider several service discovery mechanisms. User service announcement may be performed over a MBMS bearer or via other means. The download delivery method is used for the user service announcement over a MBMS bearer. The user service announcement mechanism based on the download delivery method is described in clause 5.2.3. Other user service announcement and discovery mechanisms by other means than the download delivery method are out of scope of the present document.

### 5.2.2 MBMS User Service Description metadata fragments

MBMS User Service Discovery/ Announcement is needed in order to advertise MBMS Streaming and MBMS Download User Services and User Service Bundles in advance of, and potentially during, the User Service sessions described. The User Services are described by metadata (objects/files) delivered using the download delivery method as defined in clause 7 or using interactive announcement functions.

MBMS User Service Discovery/Announcement involves the delivery of fragments of metadata to many receivers in a suitable manner. The metadata itself describes details of services. A *metadata fragment* is a single uniquely identifiable block of metadata. An obvious example of a metadata fragment would be a single SDP file (RFC 2327 [14]).

The metadata consists of:

- a metadata fragment object describing details of a single or a bundle of MBMS user services;
- a metadata fragment object(s) describing details of MBMS user service sessions;
- a metadata fragment object(s) describing details of Associated delivery methods;
- a metadata fragment object(s) describing details of service protection;
- a metadata fragment object describing details of the FEC repair data stream.

Metadata management information consists of:

- a metadata envelope object(s) allowing the identification, versioning, update and temporal validity of a metadata fragment.

The metadata envelope and metadata fragment objects are transported as file objects in the same download session either as separate referencing files or as a single embedding file - see clause 5.2.3.6). A single metadata envelope shall describe a single metadata fragment, and thus instances of the two are paired. An service announcement sender shall make a metadata envelope instance available for each metadata fragment instance. The creation and use of both an embedded envelope instance and a referenced envelope instance for a particular fragment instance is not recommended.

The metadata envelope and metadata fragment objects may be compressed using the generic GZip algorithm RFC 1952 [42] as content/transport encoding for transmission. Where used over an MBMS bearer, this shall be according to Download delivery content encoding using FLUTE - see clause 7.2.5.

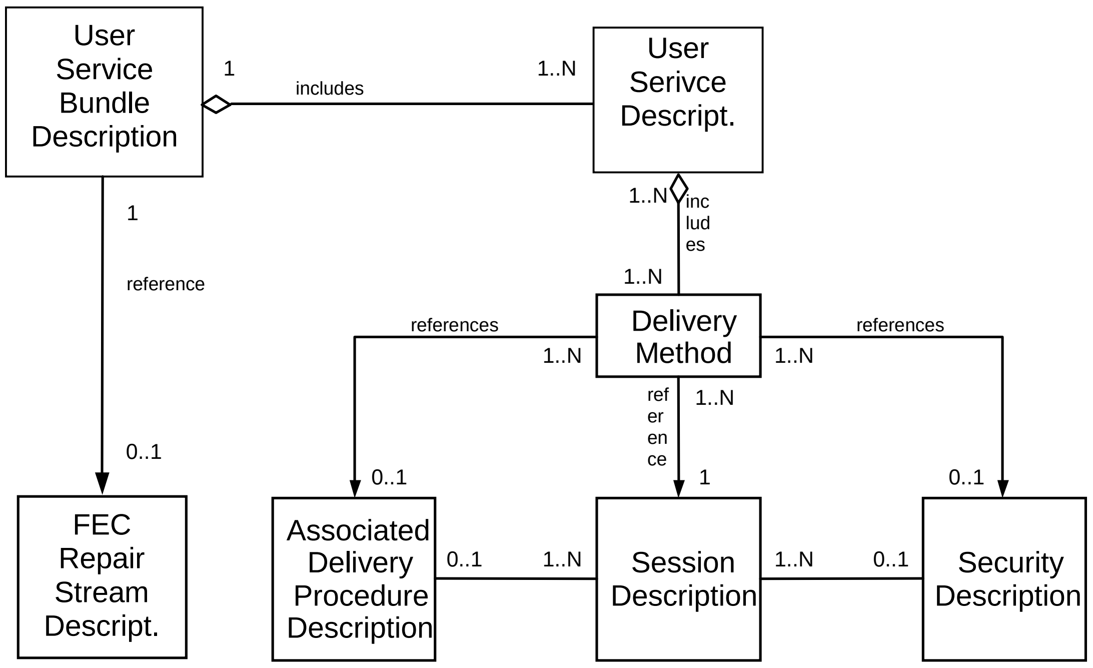


**Figure 5: Simple Description Data Model**

Figure 5 illustrates the simple data model relation between these description instances using UML [21] for a single User Service Bundle Description.

NOTE: "N" means any number in each instance.

One MBMS User Service Bundle Description shall contain at least one User Service Description instances and may contain several. The User Service Bundle Description may refer to a single FEC Repair Stream Description.

One MBMS User Service Description instance shall include at least one delivery method description instance. The delivery method description shall refer to one session description instance.

The delivery method description may contain references to a service protection description and an associated delivery procedure description. Several delivery methods may reference the same service protection description, in case the same encryption keys are used across delivery methods.

If the associated delivery procedure description is present in the user service description instance, it may be referenced by one or more delivery methods.

If the service protection description is present in the user service description instance, it may be referenced by one or more delivery methods.

An MBMS user service description allows the association of delivery methods to one or more access systems. The association is used to describe the use of separate access systems for the same MBMS user service. One delivery method may be offered throughout one or more radio access system. The use of separate MBMS bearer services for the same MBMS user service is described in clause 5.1.5.2 of [4].

Multipart MIME may be used to concatenate the descriptions one file for transport.

### 5.2.2.1 Session Description

One or more session descriptions are contained in one session description object. The session description instance shall be formatted according to the Session Description Protocol (SDP) [14]. Each session description instance must describe either one Streaming session or one FLUTE Download session. A session description for a Streaming session may include multiple media descriptions for RTP sessions. The *sessionDescriptionURI* references the session description object. The session description is specified in clause 7.3 for the MBMS download delivery method and in clause 8.3 for the MBMS streaming delivery method.

#### 5.2.2.2 Associated Delivery Procedure Description

The description and configuration of associated delivery procedures is specified in clause 9. The *associatedProcedureDescriptionURI* references the associated delivery procedure instance.

An associated delivery procedure description may be delivered on a dedicated announcement channel and updated on a dedicated announcement channel as well as in-band with an MBMS download session.

If an associated delivery procedure description for File-Repair operations is available, then the MBMS receiver may use the file repair service as specified in clause 9.3.

If an associated delivery procedure description for reception reporting is available, then the MBMS receiver shall provide reception reports as specified in clause 9.4.

#### 5.2.2.3 Service Protection Description

The security description fragment contains the key identifiers and procedure descriptions for one delivery method. When different delivery methods use the same security description, the same security description document is referenced from the different delivery method elements.

The security description is reference by the protectionDescriptionURI of the deliveryMethod element. The security description fragment shall use the MIME type application/mbms-protection-description.

The security description contains key identifiers and the server address to request the actual key material. To avoid overload situations, the same load balancing principles as in the associated delivery procedures are used. The key management server shall be selected as defined in clause 9.3.5. The back-off time shall be determined as defined in clause 9.3.4.

The root element of the security description is the securityDescription element. It contains the key identities, which are required for one delivery method. Further the security description contains one or more key management server addresses (i.e. BM-SC).

The keyManagement element defines the list of key management servers (i.e. BM-SC). The MBMS UE must register with the key management server to receive key material.

The attribute confidentialityProtection defines whether a confidentiality protection scheme is used.

The attribute integrityProtection defines whether an integrity protection scheme is used.

The attribute uiccKeyManagement defines the UICC key management in the MBMS.

The element keyId contain the key identifications and the mapping to RTP sessions or FLUTE channels. The identity element identifies the key as defined in clause 6.3.2.1 of 3GPP TS 33.246 [20]. The mediaFlow attribute specifies the RTP session or FLUTE channel. The value shall be of form <IP-destination-address>:<destination-port>. The mediaFlow element shall be present when more than one RTP session or FLUTE channel is defined in one session description element as defined in clause 5.2.2.1. When only one RTP session or one FLUTE channel is defined is the session description, then the mediaFlow attribute may not be present. The delivery method element defines the mapping between the RTP session or the FLUTE channel and the key identification.

The presence of the *fecProtection* element indicates that any MIKEY packet with an multicast destination IP address equal to any of the used destination address in the User Service Bundle Description instance"s delivery methods, are FEC protected and encapsulated in FEC source packets, see section 8.2.2.3. The attributes *fecEncodingId*, *fecInstanceID*, and *fecOtiExtension* specifies the FEC payload ID used in the source packet. All service protection descriptions referenced by a User Service Bundle Description instance shall use the same FEC parameters.

XML schema for Security Description:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
    targetNamespace="urn:3gpp:metadata:2004:securitydescription"
    xmlns="urn:3gpp:metadata:2004:securitydescription"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="securityDescription">
        <xs:element name="keyManagement" type=" keyManagementType" minOccurs="0" maxOccurs="1"/>
        <xs:sequence>
            <xs:element name="keyId" type="keyIdType" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="confidentialityProtection"
                    type="xs:boolean" use="optional" default="true"/>
        <xs:attribute name="integrityProtection"
                    type="xs:boolean" use="optional" default="true"/>
        <xs:attribute name="uiccKeyManagement"
                    type="xs:boolean" use="optional" default="true"/>
        < xs:element name="fecProtection" type="fecProtectionType" minOccurs="0" maxOccurs="1"/>
    </xs:element>

    <xs:complexType name="keyManagementType">
    <xs:sequence>
        <xs:element name="serverURI" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
        <xs:attribute name="offsetTime" type="xs:unsignedLong" use="optional" default="0"/>
        <xs:attribute name="maxBackOff" type="xs:unsignedLong" use="optional" default="0"/>
    </xs:complexType>
    <xs:complexType name="keyIdType">
        <xs:attribute name="identity" type="xs:string" use="required"/>
        <xs:attribute name="mediaFlow" type="xs:string" use="optional"/>
    </xs:complexType>
    <xs:complexType name="fecProtectionType">
        <xs:attribute name="fecEncodingId" type="xs:unsignedLong" use="required" default="0"/>
        <xs:attribute name="fecInstanceId" type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="fecOtiExtension" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:schema>
```

Example of a security description:

```
<?xml version="1.0" encoding="UTF-8"?>
<securityDescription
    xmlns="www.example.com/3gppSecurityDescription"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    confidentialityProtection="true"
    integrityProtection="true"
    uiccKeyManagement="true">
    <keyManagement
        offsetTime="5"
        maxBackOff="10">
        <serverURI ="http://register.operator.umts/"/>
        <serverURI ="http:// register2.operator.umts/"/>
    </keyManagement>
    <keyId identity="<someMSKidA>" mediaFlow=224.1.2.3:4002 />
    <keyId identity="<someMSKidB>" mediaFlow=224.1.2.3:4004 />
    <fecProtection
        fecEncodingId="130"
        fecInstanceId="0"
        fecOtiExtension="1SCxWEMNe397m24SwgyRhg=="/>
</securityDescription>
```

## 5.2.2.4 XML-Schema for MBMS User Service Bundle Description

The root element of the MBMS User Service Bundle description is the *bundleDescription* element. The element is of the *bundleDescriptionType*. The *bundleDescription* contains one or several *userServiceDescription* elements and optionally a reference to the FEC repair stream description.

Each *userServiceDescription* element shall have a unique identifier. The unique identifier shall be offered as *serviceId* attribute within the *userServiceDescription* element and shall be of URN format.

The *userServiceDescription* element may contain one or more *name* elements. The intention of a *Name* element is to offer a title of the user service. For each name elements, the language shall be specified according to XML datatypes (XML Schema Part 2 [22]).

The *userServiceDescription* element may contain one or more *ServiceLanguage* elements. Each *serviceLanguage* element represents the available languages of the user services. The language shall be specified according to XML datatypes (XML Schema Part 2 [22]).

Each *userServiceDescription* element shall contain at least one *deliveryMethod* element. The *deliveryMethod* element contains the description of one delivery method. The element shall contain one reference to a session description and may contain references to one associated delivery procedure and/or one service protection descriptions. The session description is further specified in clause 5.2.2.1.

The *deliveryMethod* element may contain a reference to an associated delivery procedure description. The description and configuration of associated delivery procedures is specified in clause 5.2.2.5.

The *deliveryMethod* element may contain a reference to a service protection description. The service protection description is specified in clause 5.2.2.3.

A *userServiceDescription* element contains zero or more *accessGroup* elements. An *accessGroup* element defines a list of access networks and is uniquely identified by its id attribute. An *accessGroup* element describes whether separate access systems for the same MBMS user service are used (see clause 5.1.5.2 of [4]) by including one or more *accessBearer* elements, each describing one of those access systems and no two describing the same. Possible *accessBearer* values are '3GPP.R6.UTRAN' and '3GPP.R6.GERAN' which indicate transport by 3GPP release 6 MBMS bearers according to specification [4]. For forward compatibility, other values are allowed but their definition and use are out of scope of this specification and a 3GPP release 6 UE may silently ignore other values.

Each *deliveryMethod* element contains zero or one *accessGroupId* attributes. One specific *accessGroupId* value maps to one specific *accessGroup* element id value. For each unique *accessGroupId* attribute value presented in a *deliveryMethod* element of a *userServiceDescription* instance, exactly one associated *accessGroup* element shall be present and the id attribute of the *accessGroup* element and the *accessGroupId* attribute shall have the same value. For each *deliveryMethod* element without an *accessGroupId* attribute, the UE should assume that the delivery method is offered through all available MBMS access systems.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema elementFormDefault="qualified"
    targetNamespace="urn:3gpp:metadata:2004:userservicedescription"
    xmlns="urn:3gpp:metadata:2004:userservicedescription"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="bundleDescription" type="bundleDescriptionType"/>
    <xs:complexType name="bundleDescriptionType">
        <xs:sequence>
            <xs:element name="userServiceDescription" type="userServiceDescriptionType"/>
                        minOccurs="1" maxOccurs="unbound"/>
        </xs:sequence>
        <xs:attribute name="fecDescriptionURI" type="xs:anyURI" use="optional"/>
    </xs:complexType>

    <xs:complexType name="userServiceDescriptionType">
        <xs:sequence>
            <xs:element name="name" type="nameType" minOccurs="0"
                        maxOccurs="unbounded"/>
            <xs:element name="serviceLanguage" type="xs:language" minOccurs="0"
                        maxOccurs="unbounded"/>
            <xs:element name="deliveryMethod" type="deliveryMethodType"
                        maxOccurs="unbounded"/>
            <xs:element name="accessGroup" minOccurs="0" maxOccurs="unbounded">
              <xs:sequence>
                 <xs:element name="accessBearer" type="xs:string" minOccurs="1"
                             maxOccurs="unbounded">
              </xs:sequence>
              <xs:attribute name="id" type="accessGroupIdType" use="required"/>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="serviceId" type="xs:anyURI" use="required"/>
    </xs:complexType>

    <xs:complexType name="accessGroupIdType" type="xs:integer"/>

    <xs:complexType name="deliveryMethodType">
        <xs:attribute name="accessGroupId"
                            type="accessGroupIdType" use="optional"/>
        <xs:attribute name="associatedProcedureDescriptionURI"
                            type="xs:anyURI" use="optional"/>
        <xs:attribute name="protectionDescriptionURI" type="xs:anyURI"
                            use="optional"/>
        <xs:attribute name="sessionDescriptionURI" type="xs:anyURI"
                            use="required"/>
    </xs:complexType>

    <xs:complexType name="nameType">
```

```
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="lang" type="xs:language" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>

</xs:schema>
```

### 5.2.2.5 Example MBMS User Service Description Instances

The following User Service Bundle Description instance is an example of a simple fragment. This fragment includes only the mandatory elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<bundleDescription  xmlns="www.example.com/3gppUserServiceDescription"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <userServiceDescription
        userServiceId="urn:3gpp:0010120123hotdog">
            <deliveryMethod
            sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>
    </userServiceDescription>
</bundleDescription>
```

The following User Service Description instance is an example of a fuller fragment.

```
<?xml version="1.0" encoding="UTF-8"?>
<bundleDescription
    xmlns="www.example.com/3gppUserServiceDescription"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    fecDescriptionURI="http://www.example.com/3gpp/mbms/session1-fec.sdp">
    <userServiceDescription
        serviceId="urn:3gpp:1234567890coolcat">
            <name lang="EN">Welcome</name>
            <name lang="DE">Willkommen</name>
        <name lang="FR">Bienvenue</name>
            <name lang="FI">Tervetuloa</name>
            <serviceLanguage>EN</serviceLanguage>
            <serviceLanguage>DE</serviceLanguage>
            <deliveryMethod
            accessGroupId="1"
                sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>
            <deliveryMethod
                sessionDescriptionURI="http://www.example.com/3gpp/mbms/session2.sdp"
            associatedProcedureDescriptionURI=
                    "http://www.example.com/3gpp/mbms/procedureX.xml"/>
            <deliveryMethod
                sessionDescriptionURI="http://www.example.com/3gpp/mbms/session3.sdp"
            associatedProcedureDescriptionURI=
                    "http://www.example.com/3gpp/mbms/procedureY.xml"/>
            <deliveryMethod
            accessGroupId="2"
                sessionDescriptionURI="http://www.example.com/3gpp/mbms/session4.sdp"/>

       <accessGroup id="1">
            <accessBearer>3GPP.R6.GERAN</accessBearer>
            <accessBearer>3GPP.R6.UTRAN</accessBearer>
       </accessGroup>
       <accessGroup id="2">
            <accessBearer>3GPP.R6.UTRAN</accessBearer>
       </accessGroup>

    </userServiceDescription>
</bundleDescription>
```

### 5.2.2.6 FEC Repair Stream Description

The streaming delivery method"s FEC has separate stream for repair data, which is described by the FEC Repair Stream Description. The FEC Repair Stream Description shall be done using SDP [14]. This SDP file is referenced by the *bundleDescription* element in the service description. The FEC Repair Stream described is common for all FEC protected packet flows within the MBMS User Service Bundle Description instance.

## 5.2.3 User service announcement over a MBMS bearer

Both the metadata envelope and metadata fragment objects are transported as file objects in the same download session.

This clause covers both metadata transport and metadata fragmentation aspects of Service Announcement. Service Announcement over MBMS bearers is specified.

To receive a Service Announcement User Service the client shall obtain the session parameters for the related MBMS download session transport. This may be using a separate Service Announcement session.

NOTE: The user service announcements are not protected when sent over MBMS bearer. See 3GPP TS 33.246 [20]

### 5.2.3.1 Supported Metadata Syntaxes

The MBMS metadata syntax supports the following set of features:

- Support of carriage of SDP descriptions, and SDP is expected to sufficiently describe at least: MBMS Streaming sessions and, MBMS download sessions.
- Support for multiple metadata syntaxes, such that the delivery and use of more than one metadata syntax is possible.
- Consistency control of metadata versions, between senders and receivers, independent of the transport and bearer use for delivery.
- Metadata fragments are identified, versioned and time-limited (expiry described) in a metadata fragment syntax-independent manner (which is a consequence of the previous two features).

### 5.2.3.2 Consistency Control and Syntax Independence

The *Metadata Envelope* provides information to identify, version and expire metadata fragments. This is specified to be independent of metadata fragments syntax and of transport method (thus enabling the use of more than one syntaxes and enable delivery over more than a single transport and bearer).

A metadata envelope may update the time validity of its metadata fragment without changing version if the metadata fragment itself has not changed. A newer version (higher version number) of a metadata envelope shall automatically expire the earlier version. If the content type (metadata fragment syntax) is recognized and valid, the UE shall use the new metadata fragment description. However, if the content type is not recognized or valid, the UE may maintain the expired version data until the newer version is correctly received.

Service announcement senders shall increment the version by one for each subsequent transported version of a metadata fragment. However, a UE shall also accept versions with an increment greater than one (so that they do not fail in the case that an intermediate version was not successfully transported).

### 5.2.3.3 Metadata Envelope Definition

The attributes for a metadata envelope and their description is as follows. These attributes shall be supported:

- *metadataURI*: A URI providing a unique identifier for the metadata fragment. The *metadataURI* attribute shall be present.
- *version*: The version number of the associated instance of the metadata fragment. The version number should be initialized to one. The version number shall be increased by one whenever the metadata fragment is updated. The *version* attribute shall be present.
- *validFrom*: The date and time from which the metadata fragment file is valid. The *validFrom* attribute may or not be present. If not present, the UE should assume the metadata fragment version is valid immediately.
- *validUntil*: The date and time when the metadata fragment file expires. The *validUntil* attribute may or not be present. If not present the UE should assume the associated metadata fragment is valid for all time, or until it receives a newer metadata envelope for the same metadata fragment describing a validUntil value.

- *contentType*: The MIME type of the metadata fragment which shall be used as defined for "Content-Type" in RFC 2616 [18]. The *contentType* attribute shall be present for embedding metadata envelopes. The *contentType* attribute may be present for referencing metadata envelopes.

The metadata envelope is instantiated using an XML structure. This XML contains a URI referencing the associated metadata fragment. The formal schema for the metadata envelope is defined as an XML Schema as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xs:element name="metadataEnvelope" type="metadataEnvelopeType" minOccurs="1"
                        maxOccurs="unbounded"/>
<xs:complexType name="metadataEnvelopeType">
        <xs:sequence>
            <xs:element name="metadataFragment"
                        type="xs:string"
                        minOccurs="0"
                        maxOccurs="1">
            </xs:element>
        </xs:sequence>
        <xs:attribute name="metadataURI"
                        type="xs:anyURI"
                        use="required"/>
        <xs:attribute name="version"
                        type="xs:positiveInteger"
                        use="required"/>
        <xs:attribute name="validFrom"
                        type="xs:dateTime"
                        use="optional"/>
        <xs:attribute name="validUntil"
                        type="xs:dateTime"
                        use="optional"/>
        <xs:attribute name="contentType"
                        type="xs:string"
                        use="optional"/>
        <xs:anyAttribute processContents="skip"/>

    </xs:element>
</xs:schema>
```

The element "metadataFragment" shall be encapsulated in the metadata envelope for embedded metadata fragments, and shall not be encapsulated where the metadata fragment is not embedded. In the embedded case, "metadataFragment" shall contain exactly one embedded metadata fragment as specified by the metadata envelope syntax and only one instance of the envelope element shall be used for encapsulating envelopes.

An embedded metadata fragment shall be escaped. Generally, an embedded metadata fragment should be escaped by placing inside a CDATA section [31]. Everything starting after "<![CDATA[" string and ending at the "]]>" string would be ignored by the XML envelope parser (quotes not included). Thus, the embedded parts would appear as "<![CDATA[" + metadata_fragment + "]]>". In this case, the complete metadata envelope with embedded metadata fragment shall not violate the rules of CDATA section sage [31].

In the case of an metadata fragment including the XML for a CDATA section, the embedded metadata fragment may be escaped by replacing illegal characters with their ampersand-escaped equivalents [31] (instead of encapsulating the whole fragment in a CDATA section). For instance "<" is an illegal character that would be replaced by "&lt;". This method is useful to avoid nesting CDATA sections (which is not allowed).

An metadata fragment which does not adhere to either of these two methods shall not be embedded in a metadata envelope, thus it may only be referenced from an referencing metadata envelope.

### 5.2.3.4 Delivery of the Metadata Envelope

An instance of metadata envelope shall be associated with an instance of an metadata fragment by one of two methods:

- Embedded: The metadata fragment is embedded within the metadata envelope.
- Referenced: The metadata fragment is referenced from the metadata envelope.

In the embedded case, the envelope and fragment are, by definition, transported together and in-band of one another. In the referenced case, the envelope and fragment shall be transported together in-band of the same transport session.

MBMS Service Announcement transports shall support delivery of the metadata envelope as a discrete object (XML file) for the referenced case. In the referenced case, the MIME type of the metadata fragment should be provided by the transport protocol (e.g. as a Content-Type text string). In both cases, the MIME type of the metadata envelope should be provided by the transport protocol.

The Metadata Envelope includes a reference (*metadataURI*) to the associated metadata fragment using the same URI as the fragment file is identified by in the Service Announcement. Thus, Metadata Envelope can be mapped to its associated metadata fragment.

#### 5.2.3.5 Metadata Envelope Transport

When FLUTE is used as the Service Announcement transport, the metadata envelope object is transported as a file object in the same MBMS service announcement download session as its metadata fragment file object (i.e. in-band with the metadata fragment session).

#### 5.2.3.6 Metadata Envelope and Metadata Fragment Association with FLUTE

The FLUTE service announcement session FDT Instances provide URIs for each transported object. The metadata envelope *metadataURI* field shall use the same URI for the metadata fragment as is used in the FDT Instances for that metadata fragment file. Thus, the fragment can be mapped to its associated envelope in-band of a single MBMS download session.

In the referencing case, each metadata envelope and corresponding metadata fragment shall be grouped together by the FDT using the grouping mechanism described by clause 7.2.5. This reduces the complexity of requesting both fragment and envelope for each pair, thus it is recommended that only the metadata fragment (fileURI) be requested from the download client (which will result in both fragment and envelope being received using the grouping mechanism).

### 5.2.4 User service announcement using Interactive Announcement Function

User service descriptions may be transported to the UE using HTTP and other interactive transport methods. A BM-SC may provide the service descriptions on request. Further protocol specifications of interactive announcement functions are outside of the scope of this specification.

Aggregated MBMS service announcement documents as specified in clause 5.2.5 may be used with the interactive announcement functions. UEs shall support the disassembly of aggregated MBMS service announcement documents. UEs shall support Gzip decoding of MBMS service description objects for interactive transport (BM-SC use of Gzip is optional in accordance with clause 5.2.2).

The BM-SC may use Metadata Envelopes as described in clauses 5.2.3.1 to 5.2.3.4, and UEs shall support their use with the Interactive Announcement Function. Where metadata envelopes are not used, only the latest delivery of a metadata fragment shall be used by the UE, and the BM-SC shall ensure timely, consistent, size-limited and secure delivery of metadata by means outside the scope of this document.5.2.4.1 User service announcement over point-to-point push bearers.

User service announcement over point-to-point push bearers have several characteristics that differ from user service announcement over a MBMS bearer. It is not essential that the metadata envelope made available by the service announcement sender is transmitted to the MBMS terminal. In the case that both the metadata envelope and metadata fragment objects are transported, it is a limitation of the solution that the metadata fragment must either be embedded within the metadata envelope, or that the metadata fragment must be referenced by the metadata envelope and they are both contained within a multipart MIME container. In either configuration, the both the metadata envelope and metadata fragment objects are transported as file objects in the same download session.

This clause covers both metadata transport and metadata fragmentation aspects of Service Announcement. Service Announcement over point-to-point push bearers is specified.

NOTE: The user service announcements are not protected when sent over point-to-point push bearers. See 3GPP TS 33.246 [20]

##### 5.2.4.1.1 Supported Metadata Syntaxes

The supported metadata syntaxes are as defined in section 5.2.3.1 of this document.

#### 5.2.4.1.2 Consistency Control and Syntax Independence

The consistency control and syntax independence is as defined in section 5.2.3.2 of this document.

#### 5.2.4.1.3 Metadata Envelope Definition

The metadata envelope definition is as defined in section 5.2.3.3 of this document.

#### 5.2.4.1.4 Delivery of the Metadata Envelope

An instance of metadata fragment shall be embedded within the metadata envelope. The envelope and fragment are, by definition, transported together and in-band of the same transport session.

The Metadata Envelope includes a reference (*metadataURI*) to the associated metadata fragment using the same URI as the fragment file is identified by in the Service Announcement. Thus, Metadata Envelope can be mapped to its associated metadata fragment.

#### 5.2.4.1.5 Metadata Envelope Transport

The metadata envelope object is transported as a file object in the same MBMS service announcement download session as its metadata fragment file object (i.e. in-band with the metadata fragment session).

#### 5.2.4.1.6 User service announcement over SMS bearers

User service announcements over SMS bearers are formatted according to the OMA Push OTA specification [79].

OTA-WSP shall be used over the SMS bearer. Application port addressing shall be used as specified in [79]. The destination application port shall be allocated by OMA. The port number is TBD.

Either confirmed or unconfirmed push may be used. In either case, the primitive shall contain the Push Headers parameter. Within this parameter, the Content-Type header shall be included and the Content-Encoding header shall be included if GZip is used.

#### 5.2.4.1.7 User service announcement over HTTP push bearers

User service announcements over HTTP push bearers are formatted according to the OMA Push OTA specification [79].

OTA-HTTP shall be used over the HTTP push bearer. Application port addressing shall be used as specified in [79]. The destination application port shall be allocated by OMA. The port number is TBD.

The Content-Encoding header shall be included if GZip is used.

### 5.2.5 Metadata fragment encapsulation to aggregate Service Announcement documents

The present document defines a number of metadata fragments to describe MBMS user services. A metadata fragment is a single uniquely identifiable block of metadata. Generally, more than one metadata fragment is necessary to provide all necessary parameters to initiate an MBMS User Service. Typically, metadata fragments are provided in separate documents. Each metadata fragment is labelled with its MIME type.

Multipart MIME may be used to encapsulate metadata fragments into an aggregate service announcement document. The aggregate document may contain metadata fragments of several MBMS user services. It is recommended, that any such aggregate service announcement document contains all the referenced metadata fragments of each MBMS user service description it contains (i.e. in the same multipart MIME structure).

An aggregate service announcement document shall encapsulate metadata fragments according to RFC 2557 [37]. The first encapsulated file of an aggregate service announcement document is the root resource. The root resource shall be either an MBMS user service description or a metadata envelope (as a referencing index). The service description metadata is defined in clause 5.2.2.4. The metadata envelope is defined in clause 5.2.3.3.

The type field of the multipart/related header shall be set to application/mbms-user-service-description-parameter in case the root resource is a user service description instance. The type field of the multipart/related header shall be set to application/mbms-envelope in case the root resource is a metadata envelope.

## 5.3 User Service Initiation/Termination

### 5.3.1 Initiation

MBMS User Service initiation refers to UE mechanisms to set-up the reception of MBMS user service data. During the User Service Initiation procedure, a set of MBMS Bearers may be activated. The User Service Initiation procedure takes place after the discovery of the MBMS user service.

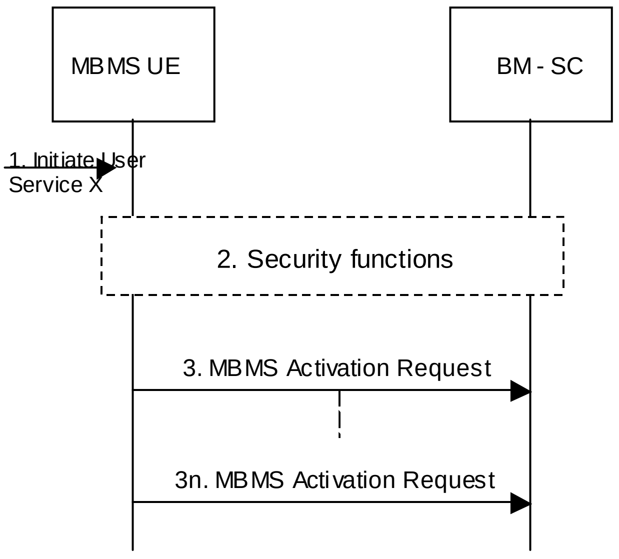

MBMS UE
BM - SC
1. Initiate User Service X
2. Security functions
3. MBMS Activation Request
3n. MBMS Activation Request

**Figure 6: Initiation of an MBMS User Service**

1. The User Service Initiation Procedure is triggered and takes a User Service Description as input that has been obtained e.g. by executing the MBMS User Service discovery and announcement functions.

2. The MBMS UE requests MBMS service keys, if security functions are activated for the MBMS User Service. The keys are sent to the UE, after the user is authorized to receive the MBMS service. The request shall be authenticated. Details on the security functions are described in 3GPP TS 33.246 [20].

3. The MBMS UE uses the MBMS activation procedure to activate the MBMS Bearer Service. The MBMS activation procedure is the MBMS Multicast Service activation procedure and the MBMS Broadcast activation procedure as defined in 3GPP TS 23.246 [4]. In case the MBMS Broadcast Mode is activated, there is no activation message sent from the UE to the BM-SC. The activation is locally in the UE. Note that the MBMS Bearer Services may already by active and in use by another MBMS User Service.

3n. In case the MBMS User Service uses several MBMS Bearer Services, the User Service Description contains several description items. In that case, the MBMS receiver function repeats the activation procedure for each MBMS Bearer Service as described in 2.

### 5.3.2 MBMS User Service termination procedure

MBMS user service termination refers to the UE mechanisms to terminate the reception of MBMS user services. A set of MBMS Bearers may be deactivated during this procedure.

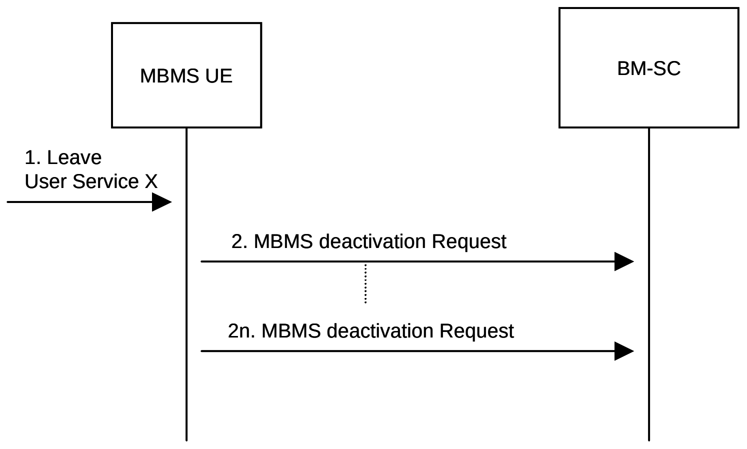


**Figure 7: Termination of an MBMS user service**

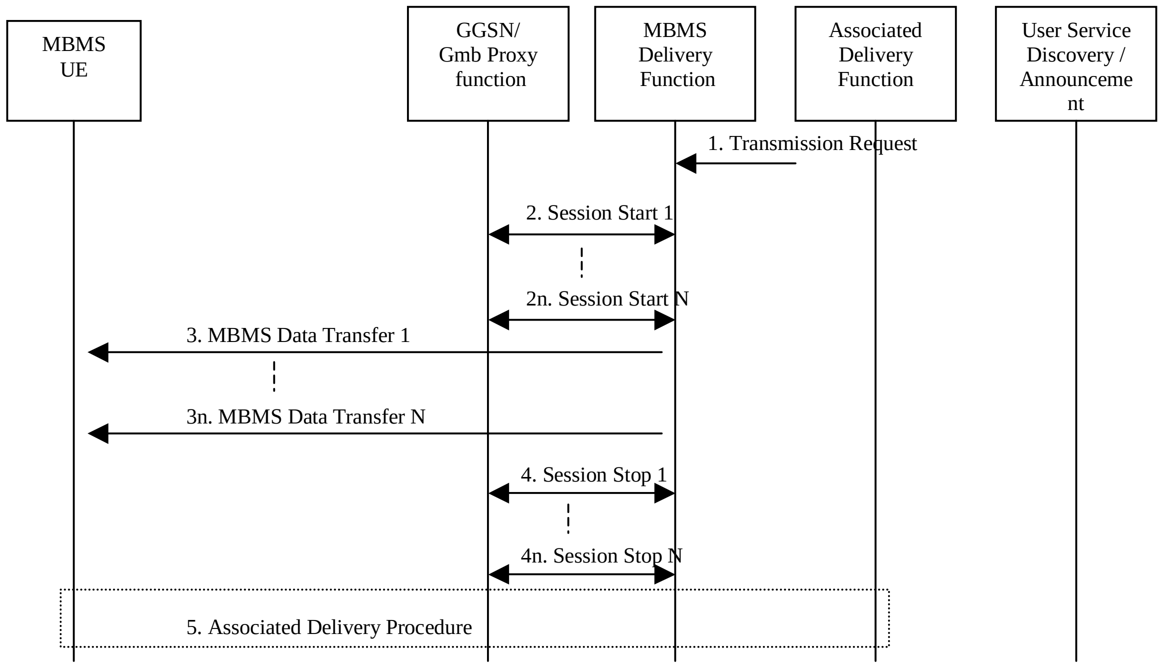
1. The User Service termination Procedure is triggered. A reference to the User Service to terminate is provided as parameter.

2. If no other MBMS User Service uses the MBMS Bearer service, the MBMS UE uses the MBMS deactivation procedure to deactivate the MBMS Bearer Services. The MBMS deactivation procedure represents the MBMS Multicast service deactivation procedure and the MBMS Broadcast deactivation procedure as described in 3GPP TS 23.246 [4]. In case the MBMS Broadcast Mode is deactivated, there is no message sent to the BM-SC. The deactivation is only locally in the UE.

2n. In case the MBMS User Service uses several Bearer Services, the UE repeats the deactivation procedure for each Bearer Service as described in 2.

## 5.4 MBMS Data Transfer Procedure

MBMS Data Transfer procedure refers to the network (and UE) mechanism to transfer (and receive) data for one MBMS User Service on one or several MBMS Bearer Services.



NOTE: Security related interactions are not depicted in the sequence.

**Figure 8: Procedure of MBMS Data Transfer**

1. The MBMS Delivery Method for the MBMS User Service is triggered by the MBMS User Service Provider. Note, details of the trigger are beyond of the present document.

2. - 2n. The MBMS Delivery function uses the MBMS Session Start Procedure to the GGSN, possibly through the Gmb Proxy function to activate all MBMS Bearer Services, which belong to the MBMS User Service. The MBMS Bearer service to be activated is uniquely identified by the TMGI.

Note, MBMS Bearer services might be activated only to a subset the available access systems (see 3GPP TS 23.246 [4]). In case MBMS User Services or delivery methods are not available throughout all access systems, the BM-SC describes this transmission strategy in the MBMS User Service Description (see clause 5.2.2).

3. - 3n. The data of the MBMS user service are transmitted to all listening MBMS UEs. Several MBMS Bearer services may be used to transmit the MBMS user service data. MBMS user service data may be integrity and/or confidentiality protected. In case MBMS user service data are integrity and/or confidentiality protected, MBMS traffic keys are delivered simultaneously on the same or a different MBMS bearer.

4. - 4n. The MBMS Delivery function uses the MBMS Session Stop procedure to trigger the GGSN, possibly through the Gmb Proxy function to release all MBMS Bearer Service for this User Service. A unique identifier for the MBMS Bearer service to be deactivated (i.e. the TMGI) is passed on as a parameter.

5. In case associated delivery procedures are allowed or requested for an MBMS User Service, the MBMS UE sends an associated-delivery procedure request to the associated -delivery function. The BM-SC may authenticate the user. See 3GPP TS 33.246 [20]. The MBMS UE may need to wait a random time before it starts the associated delivery procedure according to clause 9.

## 5.5 MBMS Protocols

Figure 9 illustrates the protocol stack used by MBMS User services. The grey-shaded protocols and functions are outside of the scope of the present document. MBMS security functions and the usage of HTTP-digest and SRTP are defined in 3GPP TS 33.246 [20].

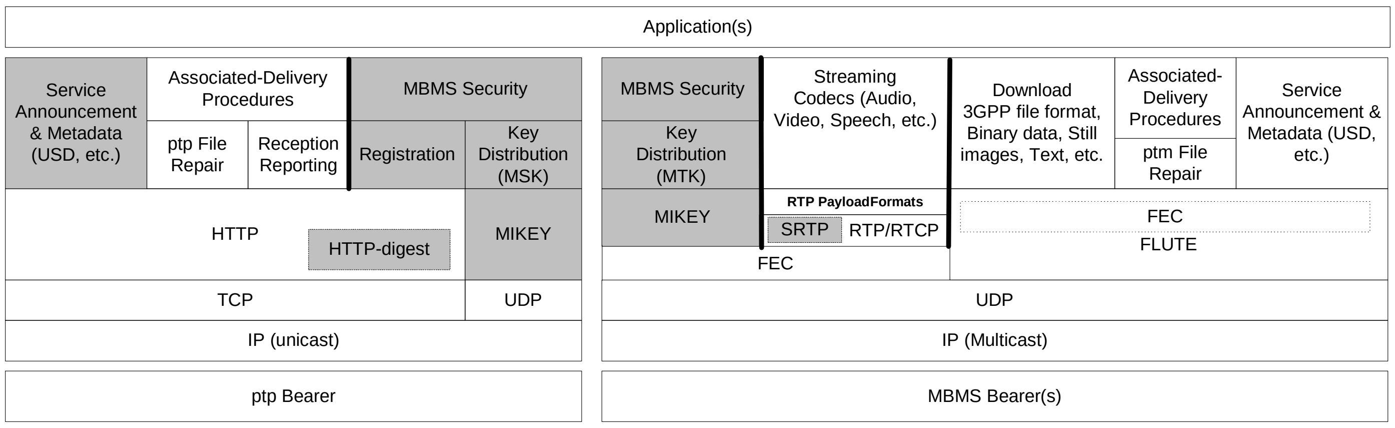


**Figure 9: Protocol stack view of the MBMS User Services**

# 6 Introduction on Delivery Methods

Two delivery methods are defined in the present document - the download delivery method and the streaming delivery method. MBMS delivery methods make use of MBMS bearers for content delivery but may also use the associated procedures defined in clause 9.

Use of MBMS bearers by the download delivery method is described in clause 7. The File Repair Procedure and the Reception Reporting Procedure (described in clause 9) may be used by the download delivery method.

Use of MBMS bearers by the streaming delivery method is described in clause 8.

# 7 Download Delivery Method

## 7.1 Introduction

MBMS download delivery method uses the FLUTE protocol (RFC 3926 [9]) when delivering content over MBMS bearers. Usage of FLUTE protocol is described in this clause.

FLUTE is built on top of the Asynchronous Layered Coding (ALC) protocol instantiation (RFC 3450 [10]). ALC combines the Layered Coding Transport (LCT) building block [11], a congestion control building block and the Forward Error Correction (FEC) building block (RFC 3452 [12]) to provide congestion controlled reliable asynchronous delivery of content to an unlimited number of concurrent receivers from a single sender. As mentioned in (RFC 3450 [10]), congestion control is not appropriate in the type of environment that MBMS download delivery is provided, and thus congestion control is not used for MBMS download delivery. See figure 10 for an illustration of FLUTE building block structure. FLUTE is carried over UDP/IP, and is independent of the IP version and the underlying link layers used.

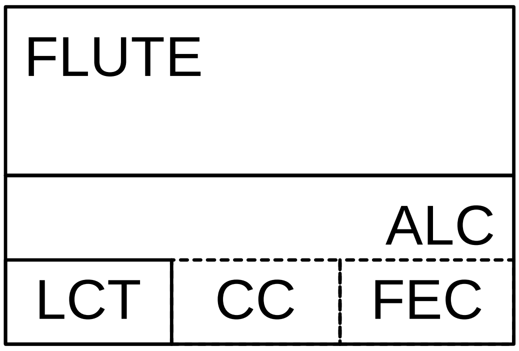


**Figure 10: Building block structure of FLUTE**

ALC uses the LCT building block to provide in-band session management functionality. The LCT building block has several specified and under-specified fields that are inherited and further specified by ALC. ALC uses the FEC building block to provide reliability. The FEC building block allows the choice of an appropriate FEC code to be used within ALC, including using the no-code FEC code that simply sends the original data using no FEC coding. ALC is under-specified and generally transports binary objects of finite or indeterminate length. FLUTE is a fully-specified protocol to transport files (any kind of discrete binary object), and uses special purpose objects - the File Description Table (FDT) Instances - to provide a running index of files and their essential reception parameters in-band of a FLUTE session.

## 7.2 FLUTE usage for MBMS download

The purpose of download is to deliver content in files. In the context of MBMS download, a file contains any type of MBMS data (e.g. 3GPP file (Audio/Video), Binary data, Still images, Text, Service Announcement metadata).

In the present document the term "file" is used for all objects carried by FLUTE (with the exception of the FDT Instances).

UE applications for MBMS user services built upon the download delivery method have three general approaches to getting files from the FLUTE receiver for a joined session:

- **Promiscuous:** Instruct FLUTE to promiscuously receive all files available. Promiscuous reception can be suitable for single purpose sessions (generally with limited number and/or size of files) although uncertainty over the quality and content of files makes this approach generally undesirable.

- **One-copy:** Instruct FLUTE to receive a copy of one or more specific files (identified by the fileURI) - and potentially leaving the session following reception of one copy of all the specified files. Specifying the download file ensures that the UE has an upper bound to the quantity of files downloaded. One-copy reception requires prior knowledge of the file identifiers (fileURIs).

- **Keep-updated:** Instruct FLUTE to receive one or more specific files and continue to receive any updates to those files. As with one-copy, the keep-updated approach bounds the quantity of files downloaded and requires prior knowledge of the file identifiers.

NOTE: The present document does not prevent or endorse changing download reception approach, and any related file list, during the life of the download session. Discovery of session content lists (including file lists) out-of-band of the delivery method sessions is beyond the scope of the present document.

MBMS clients and servers supporting MBMS download shall implement the FLUTE specification (RFC 3926 [9]), as well as ALC (RFC 3450 [10]) and LCT (RFC 3451 [11]) features that FLUTE inherits. In addition, several optional and extended aspects of FLUTE ,as described in the following clauses, shall be supported.

### 7.2.1 Fragmentation of Files

Fragmentation of files shall be provided by a blocking algorithm (which calculates source blocks from source files) and a symbol encoding algorithm (which calculates encoding symbols from source blocks).

### 7.2.2 Symbol Encoding Algorithm

The "Compact No-Code FEC scheme" - RFC 3452 [12] (FEC Encoding ID 0, also known as "Null-FEC") shall be supported.

The 'MBMS FEC scheme' is described in clause 7.2.12.

A UE that supports MBMS User Services shall support a decoder for the 'MBMS FEC scheme'.

If a UE that supports MBMS User Services receives a mathematically sufficient set of encoding symbols generated according to the encoder specification in Annex B for reconstruction of a source block then the decoder shall recover the entire source block. Note that the example decoder described in annex B fulfils this requirement.

### 7.2.3 Blocking Algorithm

In the case of the Compact No-Code FEC scheme [12] (FEC Encoding ID 0), then the "Algorithm for Computing Source Block Structure" described within the FLUTE specification (RFC 3926 [9]) shall be used.

In the case of MBMS forward error correction, then the algorithm defined in Annex B shall be used.

The values of $N$, $Z$, $T$ and $A$ shall be set such that the sub-block size is less than 256KB.

### 7.2.4 Congestion Control

For simplicity of congestion control, FLUTE channelization shall be provided by a single FLUTE channel with single rate transport.

### 7.2.5 Content Encoding of Files for Transport

Files may be content encoded for transport, as described in [9], in the Download delivery method using the generic GZip algorithm RFC 1952 [42]. UEs shall support GZip content decoding of FLUTE files (GZIP RFC 1952 [42], clause 9).

### 7.2.6 Transport File Grouping

Files downloaded as part of a multiple-file delivery are generally related to one another. Examples includes web pages, software packages, and the referencing metadata envelopes and their metadata fragments. FLUTE clients analyse the XML-encoded FDT Instances as they are received, identify each requested file, associate it with FLUTE packets (using the TOI) and discover the relevant in-band download configuration parameters of each file.

An additional "group" field in the FLUTE FDT instance and file elements enables logical grouping of related files. A FLUTE receiver should download all the files belonging to all groups where one or more of the files of those groups have been requested. However, a UE may instruct its FLUTE receiver to ignore grouping to deal with special circumstances, such as low storage availability.

The group names are allocated by the FLUTE sender and each specific group name shall group the corresponding files together as one group, including files describes in the same and other FDT Instances, for a session.

Group field usage in FDT Instances is shown in the FDT XML schema (clause 7.2.9). Each file element of an FDT Instance may be labelled with zero, one or more group names. Each FDT Instance element may be labelled with zero, one or more group names which are inherited by all files described in that FDT Instance.

### 7.2.7 Signalling of Parameters with Basic ALC/FLUTE Headers

FLUTE and ALC mandatory header fields shall be as specified in [9, 10] with the following additional specializations:

- The length of the CCI (Congestion Control Identifier) field shall be 32 bits and it is assigned a value of zero (C=0).
- The Transmission Session Identifier (TSI) field shall be of length 16 bits (S=0, H=1, 16 bits).
- The Transport Object Identifier (TOI) field should be of length 16 bits (O=0, H=1).
- Only Transport Object Identifier (TOI) 0 (zero) shall be used for FDT Instances.
- The following features may be used for signalling the end of session and end of object transmission to the receiver:
    - The Close Session flag (A) for indicating the end of a session.
    - The Close Object flag (B) for indicating the end of an object.

In FLUTE the following applies:

- The T flag shall indicate the use of the optional "Sender Current Time (SCT)" field (when T=1).
- The R flag shall indicate the use of the optional "Expected Residual Time (ERT)" field (when R=1).
- The LCT header length (HDR_LEN) shall be set to the total length of the LCT header in units of 32-bit words.
- For "Compact No-Code FEC scheme" [12], the FEC Payload ID shall be set according to RFC 3695 [13] such that a 16 bit SBN (Source Block Number) and then the 16 bit ESI (Encoding Symbol ID) are given.
- For 'MBMS FEC scheme', the FEC Payload ID shall be set according to Section 7.2.10 below.

### 7.2.8 Signalling of Parameters with FLUTE Extension Headers

FLUTE extension header fields EXT_FDT, EXT_FTI , EXT_CENC [9] shall be used as follows:

- EXT_FTI shall be included in every FLUTE packet carrying symbols belonging to any FDT Instance.
- FLUTE packets carrying symbols of files (not FDT Instances) shall not include an EXT_FTI.
- FDT Instances shall not be content encoded and therefore EXT_CENC shall not be used.

In FLUTE the following applies:

- EXT_FDT is in every FLUTE packet carrying symbols belonging to any FDT Instance.
- FLUTE packets carrying symbols of files (not FDT instances) do not include the EXT_FDT.

### 7.2.9 Signalling of Parameters with FDT Instances

The FLUTE FDT Instance schema (RFC 3926 [9]) shall be used. In addition, the following applies to both the session level information and all files of a FLUTE session.

The inclusion of these FDT Instance data elements is mandatory according to the FLUTE specification:

- Content-Location (URI of a file).
- TOI (Transport Object Identifier of a file instance).
- Expires (expiry data for the FDT Instance).

Additionally, the inclusion of these FDT Instance data elements is mandatory:

- Content-Length (source file length in bytes).
- Content-Type (content MIME type).
- FEC Encoding ID.

Other FEC Object Transmission Information specified by the FEC scheme in use :

NOTE: The FEC Object Transmission Information elements used are dependent on the FEC scheme, as indicated by the FEC Encoding ID.

- FEC-OTI-Maximum-Source-Block-Length.
- FEC-OTI-Encoding-Symbol-Length.
- FEC-OTI-Max-Number-of-Encoding-Symbols.
- FEC-OTI-Scheme-Specific-Info.

NOTE 1: RFC 3926 [9] describes which part or parts of an FDT Instance may be used to provide these data elements.

These optional FDT Instance data elements may or may not be included for FLUTE in MBMS:

- Complete (the signalling that an FDT Instance provides a complete, and subsequently unmodifiable, set of file parameters for a FLUTE session may or may not be performed according to this method).
- Content-Encoding.

NOTE 2: The values for each of the above data elements are calculated or discovered by the FLUTE sender.

The FEC-OTI-Scheme-Specific-Info FDT Instance data element contains information specific to the FEC scheme indicated by the FEC Encoding ID encoded using base64.

## 7.2.10 FDT Schema

The following XML Schema shall be use for the FDT Instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:fl="http://www.example.com/flute"
        elementFormDefault:xs="qualified"
        targetNamespace:xs="http://www.example.com/flute">
    <xs:element name="FDT-Instance">
        <xs:complexType>
        <xs:sequence>
            <xs:element name="File" maxOccurs="unbounded">
                <xs:complexType>

                <xs:sequence>
                    <xs:element name="Group" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>

                <xs:sequence>
                    <xs:element name="MBMS-Session-Identity" type="xs:unsignedByte"
                                minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>

                <xs:any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
                <xs:attribute name="Content-Location" type="xs:anyURI" use="required"/>
                <xs:attribute name="TOI" type="xs:positiveInteger" use="required"/>
                <xs:attribute name="Content-Length" type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="Transfer-Length" type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="Content-Type" type="xs:string" use="optional"/>
                <xs:attribute name="Content-Encoding" type="xs:string" use="optional"/>
                <xs:attribute name="Content-MD5" type="xs:base64Binary" use="optional"/>
                <xs:attribute name="FEC-OTI-FEC-Encoding-ID" type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="FEC-OTI-FEC-Instance-ID" type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length"
                                    type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="FEC-OTI-Encoding-Symbol-Length"
                                    type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="FEC-OTI-Max-Number-of-Encoding-Symbols"
                                    type="xs:unsignedLong" use="optional"/>
                <xs:attribute name="FEC-OTI-Scheme-Specific-Info"
                                    type="xs:base64Binary" use="optional"/>

                <xs:anyAttribute processContents="skip"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>

        <xs:sequence>
            <xs:element name="Group" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>

        <xs:sequence>
            <xs:element name="MBMS-Session-Identity-Expiry"
                        type="MBMS-Session-Identity-Expiry-Type"
                        minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>

        <xs:any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>

        <xs:attribute name="Expires" type="xs:string" use="required"/>
        <xs:attribute name="Complete" type="xs:boolean" use="optional"/>
        <xs:attribute name="Content-Type" type="xs:string" use="optional"/>
        <xs:attribute name="Content-Encoding" type="xs:string" use="optional"/>
        <xs:attribute name="FEC-OTI-FEC-Encoding-ID" type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="FEC-OTI-FEC-Instance-ID" type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length"
                                type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="FEC-OTI-Encoding-Symbol-Length"
                                type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="FEC-OTI-Max-Number-of-Encoding-Symbols"
                                type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="FEC-OTI-Scheme-Specific-Info"
                                    type="xs: base64Binary" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
        </xs:complexType>
    </xs:element>

    <xs:complexType name="MBMS-Session-Identity-Expiry-Type">
        <xs:attribute name="id" type="xs:unsignedByte" use="required"/>
        <xs:attribute name="value" type="xs:string" use="required"/>
    </xs:complexType>

</xs:schema>
```

### 7.2.11 MBMS Session Identify

The *MBMS-Session-Identity* element associates the file to the identity of the MBMS session. If the file will be part of several MBMS transmission sessions, then a list of MBMS session identities is defined.

The *MBMS-Session-Identity-Expiry* element associates an expiration time with a MBMS session identity value. Similar to the FLUTE FDT expiration time, the MBMS session identity expiration time (*value* attribute) is expressed within the FDT Instance payload as a 32 bit data field. The value of the data field represents the 32 most significant bits of a 64 bit Network Time Protocol (NTP) [78] time value. These 32 bits provide an unsigned integer representing the time in seconds relative to 0 hours 1 January 1900.

## 7.2.12 FEC Scheme definition

### 7.2.12.1 General

This clause defines an FEC encoding scheme for the MBMS forward error correction code defined in Annex B for the download delivery method. This scheme is identified by FEC Encoding ID [TBA]. The FEC Payload ID format and FEC Object Transmission Information format are as defined in the following clauses.

### 7.2.12.2 FEC payload ID

The FEC Payload ID shall be a 4 octet field defined as follows:

| | |
|---|---|
| Source Block Number (SBN) | Encoding Symbol ID (ESI) |

Source Block Number (SBN), (16 bits): An integer identifier for the source block that the encoding symbols within the packet relate to.

Encoding Symbol ID (ESI), (16 bits): An integer identifier for the encoding symbols within the packet.

The interpretation of the Source Block Number and Encoding Symbol Identifier is defined in Annex B.

### 7.2.12.3 FEC Object Transmission Information

The FEC Object Transmission information shall consist of:

- The FEC Encoding ID
- The Transfer Length (*F*)
- The parameters *T*, *Z*, *N* and *A* defined in Annex B.

When EXT-FTI is used to communicate the Object Transmission Information, the FEC Encoding ID and Transfer Length shall be coded according to FLUTE [9]. The other parameters shall be encoded in the FEC Encoding ID specific portion of the EXT_FTI field as shown in Figure 10a below.

| | | |
|---|---|---|
| General EXT_FTI format | Encoding Symbol Length (*T*) | |
| Number of Source Blocks (*Z*) | Number of Sub-Blocks (*N*) | Symbol Alignment Parameter (*A*) |

**Figure 10a: FEC Encoding ID-specific EXT_FTI format**

The parameters *T* and *Z* are 16 bit unsigned integers, *N* and *A* are 8 bit unsigned integers.

When the FDT is used to deliver the FEC Object Transmission Information, then the FEC Encoding ID, Transfer Length (F) and Encoding Symbol Length (*T*) shall be encoded using the Transfer-Length, FEC-OTI-Encoding-ID and FEC-OTI-Encoding-Symbol-Length elements defined in Section 7.2.10. The remaining parameters *Z*, *N*, *A*, shall be encoded as a 4 byte field within the FEC-OTI-Scheme-Specific-Info field, according to the format specified in Figure 10a above, excepting the Encoding Symbol Length field.

## 7.3 SDP for Download Delivery Method

### 7.3.1 Introduction

RFC 3926 [9] describes required and optional parameters for FLUTE session and media descriptors. This clause specifies SDP for FLUTE session that is used for the MBMS download and service announcement sessions. The formal specification of the parameters is given in ABNF (RFC 2234 [23]).

### 7.3.2 SDP Parameters for MBMS download session

The semantics of a Session Description of an MBMS download session includes the following parameters:

- The sender IP address.
- The number of channels in the session.
- The destination IP address and port number for each channel in the session per media.
- The Transport Session Identifier (TSI) of the session.
- The start time and end time of the session.
- The protocol ID (i.e. FLUTE/UDP).
- Media type(s) and fmt-list.
- Data rate using existing SDP bandwidth modifiers.
- Mode of MBMS bearer per media.
- FEC capabilities and related parameters.
- Service-language(s) per media.

This list includes the parameters required by FLUTE - RFC 3926 [9]

These shall be expressed in SDP (RFC 2327 [14], draft-ietf-mmusic-sdp-srcfilter [15], RFC 3266 [16]) syntax according to the following clauses.

#### 7.3.2.1 Sender IP address

There shall be exactly one IP sender address per MBMS download session, and thus there shall be exactly one IP source address per complete MBMS download session SDP description. The IP source address shall be defined according to the source-filter attribute ("a=source-filter:") (RFC 2327 [14] and draft-ietf-mmusic-sdp-srcfilter [15]) for both IPv4 and IPv6 sources, with the following exceptions:

1. Exactly one source address may be specified by this attribute such that exclusive-mode shall not be used and inclusive-mode shall use exactly one source address in the <src-list>.
2. There shall be exactly one source-filter attribute per complete MBMS download session SDP description, and this shall be in the session part of the session description (i.e. not per media).
3. The * value shall be used for the <dest-address> subfield, even when the MBMS download session employs only a single LCT (multicast) channel.

#### 7.3.2.2 Number of channels

Only one FLUTE channel is allowed per FLUTE session in the present document and thus there is no further need for a descriptor of the number of channels.

#### 7.3.2.3 Destination IP address and port number for channels

The FLUTE channel shall be described by the media-level channel descriptor. These channel parameters shall be per channel:

- IP destination address.
- Destination port number.

The IP destination address shall be defined according to the "connection data" field ("c=") of SDP (RFC 2327 [14] and draft-ietf-mmusic-sdp-srcfilter [15]). The destination port number shall be defined according to the <port> sub-field of the media announcement field ("m=") of SDP (RFC 2327 [14] and draft-ietf-mmusic-sdp-srcfilter [15]).

The presence of a FLUTE session on a certain channel shall be indicated by using the "*m*-line" in the SDP description as shown in the following example:

*m=application 12345 FLUTE/UDP 0*

*c=IN IP6 FF1E:03AD::7F2E:172A:1E24/1*

In the above SDP attributes, the *m*-line indicates the media used and the *c*-line indicates the corresponding channel. Thus, in the above example, the *m*-line indicates that the media is transported on a channel that uses FLUTE over UDP. Further, the *c*-line indicates the channel address, which, in this case, is an IPv6 address.

#### 7.3.2.4 Transport Session Identifier (TSI) of the session

The combination of the TSI and the IP source address identifies the FLUTE session. Each TSI shall uniquely identify a FLUTE session for a given IP source address during the time that the session is active, and also for a large time before and after the active session time (this is also an LCT requirement - RFC 3451 [11]).

The TSI shall be defined according the SDP descriptor given below. There shall be exactly one occurrence of this descriptor in a complete FLUTE SDP session description and it shall appear at session level.

The syntax in ABNF is given below:

- sdp-flute-tsi-line = "a" "=" "flute-tsi" ":" integer CRLF.
- integer = as defined in RFC 2327 [14].

#### 7.3.2.5 Multiple objects transport indication

RFC 3626 [9] requires the use of the Transport Object Indentifier (TOI) header field (with one exception for packets with no payload when the A flag is used). The transport of a single FLUTE file requires that multiple TOIs are used (TOI 0 for FDT Instances). Thus, there is no further need to indicate to receivers that the session carries packets for more than one object and no SDP attribute (or other FLUTE out of band information) is needed for this.

#### 7.3.2.6 Session Timing Parameters

A MBMS download session start and end times shall be defined according to the SDP timing field ("t=") (RFC 2327 [14] and draft-ietf-mmusic-sdp-srcfilter [15]).

#### 7.3.2.7 Mode of MBMS bearer per media

A new MBMS bearer mode declaration attribute is defined which results in, e.g.:

- a=mbms-mode:broadcast 1234

Where any media for the MBMS multicast mode are described, the MBMS bearer mode declaration attribute shall not be used at session level. The MBMS bearer mode declaration attribute shall not be used at media level for media flows on the MBMS multicast mode. The MBMS bearer mode declaration attribute shall be used to described MBMS broadcast mode media.

The MBMS bearer mode declaration attribute may be session-level (where all media are broadcast (and so becomes the default for all media); and media-level to specify differences between media.

- mbms-bearer-mode-declaration-line = "a=mbms-mode:" "broadcast" SP tmgi CRLF:
  - tmgi = 1*DIGIT

The Temporary Mobile Group Identity (tmgi) information element is defined in RFC 3267 [33]. Only octets 3 to 5 or octets 3 to 8 are encoded in tmgi attribute as a decimal number. Octet 3 is the most significant octet.

### 7.3.2.8 FEC capabilities and related parameters

A new FEC-declaration attribute is defined which results in, e.g.:

- a=FEC-declaration:0 encoding-id=128; instance-id=0

This attribute may be used on both session-level and media-level. Multiple instances are allowed to specify several different FEC declarations. The attribute is used on session level to define FEC declarations used by multiple media components. Each media component references an FEC declaration using the 'a=FEC' attribute. On media level it is used to define FEC declarations which are only valid for a single media component. If FEC declarations on both session and media level use the same reference number (fec-ref) then the media level declaration takes precedence for that media component.

This attribute is optional to use for the download delivery method as the information will be available elsewhere (e.g. FLUTE FDT Instances). If this attribute is not used, and no other FEC-OTI information is signalled to the UE by other means, the UE may assume that support for FEC id 0 is sufficient capability to enter the session.

A new FEC-declaration reference attribute is defined which results in, e.g.:

- a=FEC:0

This is a media-level only attribute, used as a short hand to reference one of one or more FEC-declarations.

The syntax for the attributes in ABNF - RFC 2234 [23] is:

- sdp-fec-declaration-line = "a=FEC-declaration:" fec-ref SP fec-enc-id ";" [SP fec-inst-id] CRLF
- fec-ref = 1*3DIGIT (value is the SDP-internal identifier for FEC-declaration).
- fec-enc-id = "encoding-id=" enc-id
- end-id = 1*DIGIT (value is the FEC Encoding ID used).
- fec-inst-id = "instance-id=" inst-id
- inst-id = 1*DIGIT (value is the FEC Instance ID used).
- sdp-fec-line = "a=FEC:" fec-ref CRLF

### 7.3.2.9 Service-language(s) per media

The existing SDP attribute "a=lang" is used to label the language of any language-specific media. The values are taken from RFC 3066 [73] which in turn takes language and (optionally) country tags from ISO 639 [74] and ISO 3661 [75] (e.g. "a=lang:EN-US"). These are the same tags used in the User Service Description XML.

### 7.3.2.10 Bandwidth Specification

The maximum bit-rate required by this FLUTE session shall be specified using the "AS" bandwidth modifier RFC 2327 [14] on media level. The Application Specific (AS) bandwidth for a FLUTE session shall be the largest sum of the sizes of all packets transmitted during any one second long period of the session, expressed as kilobits. The size of the packet shall be the complete packet, i.e. IP, UDP and FLUTE headers, and the data payload.

### 7.3.3 SDP Examples for FLUTE Session

Here is a full example of SDP description describing a FLUTE session:

```
v=0
o=user123 2890844526 2890842807 IN IP6 2201:056D::112E:144A:1E24
s=File delivery session example
i=More information
t=2873397496 2873404696
a=mbms-mode:broadcast 1234
a=FEC-declaration:0 encoding-id=128; instance-id=0
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=flute-tsi:3

m=application 12345 FLUTE/UDP 0
c=IN IP6 FF1E:03AD::7F2E:172A:1E24/1
b=64
a=lang:EN
a=FEC:0
```

# 8 Streaming delivery method

## 8.1 Introduction

The purpose of the MBMS streaming delivery method is to deliver continuous multimedia data (i.e. speech, audio and video) over an MBMS bearer. This delivery method complements the download delivery method which consists of the delivery of files. The streaming delivery method is particularly useful for multicast and broadcast of scheduled streaming content.

## 8.2 Transport protocol

RTP is the transport protocol for MBMS streaming delivery. RTP provides means for sending real-time or streaming data over UDP and is already used for the transport of PSS in 3GPP. RTP provides RTCP for feedback about the transmission quality. The transmission of RTCP packets in the downlink (sender reports) is allowed. In this version of the specification, RTCP RR shall be turned off by SDP RR bandwidth modifiers. Note that in the context of MBMS detection of link aliveness is not necessary.

### 8.2.1 RTP payload formats for media

The RTP payload formats and corresponding MIME types are aligned with those defined in PSS Rel-6 3GPP TS 26.234 [47] as much as possible. For RTP/UDP/IP transport of continuous media the following RTP payload formats shall be used:

- AMR narrow-band speech codec (see clause 10.2) RTP payload format according to RFC 3267 [33]. A MBMS client is not required to support multi-channel sessions.
- AMR wideband speech codec (see clause 10.2) RTP payload format according to RFC 3267 [33]. A MBMS client is not required to support multi-channel sessions.
- Extended AMR-WB codec (see clause 10.3) RTP payload format according to [34].
- Enhanced aacPlus codec (see clause 10.3): RTP payload format and MIME types according to RFC 3640 [41], namely the Low Bit-Rate AAC or the High Bit-Rate AAC modes.

- H.264 (AVC) video codec (see clause 10.4) RTP payload format according to RFC 3987 [35]. An MBMS client supporting H.264 (AVC) is required to support all three packetization modes: single NAL unit mode, non-interleaved mode and interleaved mode. For the interleaved packetization mode, an MBMS client shall support streams for which the value of the "sprop-deint-buf-req" MIME parameter is less than or equal to MaxCPB * 1000 / 8, inclusive, in which "MaxCPB" is the value for VCL parameters of the H.264 (AVC) profile and level in use, as specified in [TBD].

### 8.2.2 FEC mechanism for RTP

The 'MBMS FEC scheme' is described in clause 8.2.2.8.

A UE that supports MBMS User Services shall support a decoder for the 'MBMS FEC scheme'.

This section defines a generic mechanism for applying Forward Error Correction to streaming media. The mechanism consists of three components:

(i) construction of an FEC source block from the source media packets belonging to one or several UDP packet flows related to a particular segment of the stream(s) (in time). The UDP flows is include RTP, RTCP, SRTP and MIKEY packets.

(ii) modification of source packets to indicate the position of the source data from the source packet within the source block

(iii)definition of repair packets, sent over UDP, which can be used by the FEC decoder to reconstruct missing portions of the source block.

The mechanism does not place any restrictions on the source data which can be protected together, except that the source data is carried over UDP. The data may be from several different UDP flows that are protected jointly.

A receiver supporting the streaming delivery method shall support the packet format for FEC source packets and may also support the packet format for FEC repair packets.

At the sender, the mechanism begins by processing original UDP packets to create:

(i) a stored copy of the original packets in the form of a source block; and

(ii) FEC source packets for transmission to the receiver.

After constructing the source block from the original UDP payloads to be protected and their flow identity (based on destination IP address and UDP port), the FEC encoder generates the desired amount of FEC protection data, i.e. encoding symbols. These encoding symbols are then sent using the FEC repair packet format to the receiver. The FEC repair packets are sent to a UDP destination port different from any of the original UDP packets' destination port(s) as indicated by the signaling.

The receiver recovers the original packets directly from the FEC source packets and buffers them at least the min-buffer-time to allow time for the FEC repair. The receiver uses the FEC source packets to construct a (potentially incomplete) copy of the source block, using the Source FEC Payload ID in each packet to determine where in the source block the packet shall be placed. In indication of the UPD flow (i.e. destination IP address and UDP port) the packet is part of is included in the source block with the UDP payload.

If any FEC source packets have been lost, but sufficient FEC source and FEC repair packets have been received, FEC decoding can be performed to recover the FEC source block. The original packets UDP payload and UDP flow identity can then be extracted from the source block and provided to the upper layer. If not enough FEC source and repair packets were received, only the original packets that were received as FEC source packets will be available. The rest of the original packets are lost.

If a UE that supports MBMS User Services receives a mathematically sufficient set of encoding symbols generated according to the encoder specification in Annex B for reconstruction of a source block then the decoder shall recover the entire source block. Note that the example decoder described in Annex B fulfils this requirement.

Note that the receiver must be able to buffer all the original packets and allow time for the FEC repair packets to arrive and FEC decoding to be performed before media playout begins. The min-buffer-time parameter specified in clause 8.3.1.9 helps the receiver to determine a sufficient duration for initial start-up delay.

The Source and Repair FEC payload IDs are used to associate the FEC source packets and FEC repair packets, respectively, to a source block. The Source and Repair FEC payload ID formats are part of the definition of the FEC scheme. Each FEC scheme is identified by an FEC Encoding ID and, in the case of underspecified FEC schemes, FEC Instance ID, values. One FEC scheme for the streaming delivery method is specified in clause 8.2.2.6. Any FEC schemes using the packet formats defined in the present document shall be systematic FEC codes and may use different FEC payload ID formats for FEC source packets and FEC repair packets.

The protocol architecture is illustrated in figure 11.

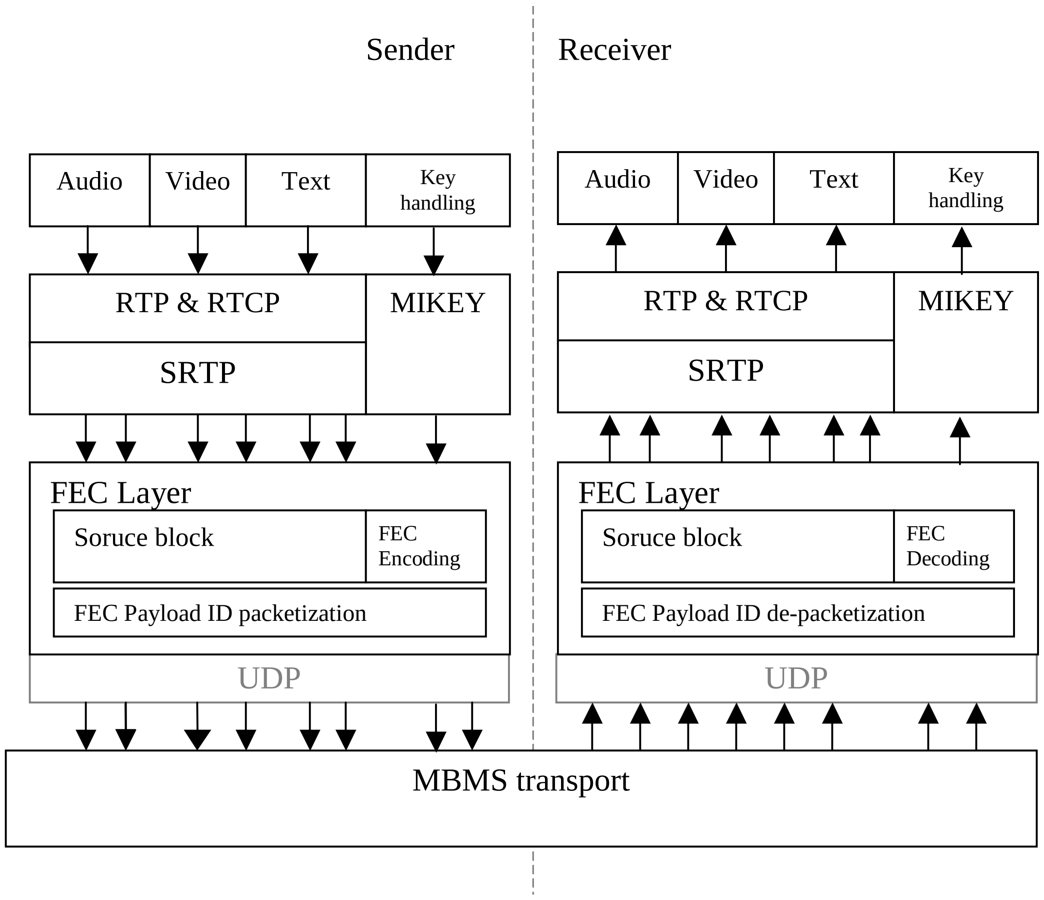


**Figure 11: FEC mechanism for the streaming delivery method interaction diagram**

Figure 11 depicts how one or more out of several possible packet flows of different types (Audio, video, text RTP and RTCP flows, MIKEY flow) are sent to the FEC layer for protection. The source packets are modified to carry the FEC payload ID and a new flow with repair data is generated. The receiver takes the source and repair packets and buffers them to perform, if necessary, the FEC decoding. After appropriate buffering received and recovered source packets are forwarded to the higher layers. The arrows in the figure indicate distinct data flows.

### 8.2.2.1 Sending Terminal Operation (Informative)

It is assumed that the sender has constructed or received original data packets for the session. These may be RTP, RTCP, MIKEY or other UDP packets. The following procedures are based on the UDP payload and the identity of the UDP flow.

In order to FEC protect a sequence of original data packets, the sender constructs a source block as specified in clause 8.2.2.6 to which the FEC algorithm is to be applied, and includes the original source packet data within FEC source packets. The following operations describe a possible way to generate compliant FEC source packet and FEC repair packet streams:

1. Each original packet is placed in the source block. In doing so, the Source FEC Payload ID information to be included in the FEC payload ID of the FEC source packet can be determined. In the source block the identity of the packet"s flow is marked using the Flow ID. See clause 8.2.2.5 and 8.2.2.7 for details.
2. The FEC source packet is constructed according to clause 8.2.2.3. The identity of the original flow is maintained by the source packet through the use of the destination UDP port number and destination IP address, which has been advertised (for example using SDP), as carrying FEC source packets generated from an original stream of a particular protocol (e.g. RTP, RTCP, SRTP, MIKEY etc.). See Clause 8.2.2.11.
3. The FEC source packet generated is sent according to UDP procedures defined in RFC 3267 [33].

When a source block is complete, the FEC encoder generates encoding symbols and places these symbols into FEC repair packets, to be conveyed to the receivers. These repair packets are sent using normal UDP procedures to a unique destination port to separate it from any of the source packet flows.

#### 8.2.2.2 Receiving Terminal Operation (Informative)

The following describes a possible receiver algorithm, when receiving an FEC source or repair packet:

1. If a FEC source packet is received (as indicated by the UDP port on which was received):
   a. The original source packet is reconstructed by removing the Source FEC Payload ID. The resulting packet is buffered to allow time for the FEC repair.
   b. The resulting packet is placed into the source block according to the information in the Source FEC Payload ID and the source block format described in clause 8.2.2.5. The UDP port the packet was received on is used to determine the Flow ID written into the source block.
2. If an FEC repair packet is received (as indicated by the UDP port), the contained encoding symbols are placed into an FEC encoding block according to the Repair FEC Payload ID.
3. If at least one source packet is missing, then FEC decoding may be desirable. The FEC decoder determines if the encoding block constructed in steps 1 and 2 contains enough symbols from the source and repair packets for decoding and, if so, performs the decoding operation.
4. Any missing source packets that were reconstructed during the decoding operation are then buffered as normal received packets (see step 1a above).

Note that the above procedure may result in that not all original packets are recovered, and they must simply be marked as being lost.

Obviously, buffering and packet re-ordering arerequired to insert any reconstructed packets in the appropriate place in the packet sequence if that is necessary according to the used higher layer protocol (RTP, RTCP or MIKEY). To allow receivers to determine the minimal start-up buffering requirement for FEC decoding, the min-buffer-time parameter indicates a minimum initial buffering time that is sufficient regardless of the position of the stream in which the reception starts.

### 8.2.2.3 (Void)

### 8.2.2.4 Packet format for FEC source packets

The packet format for FEC source packets shall be used to encapsulate an original UDP packet.. As depicted in figure 12, it consists of the original UDP packet, followed by the Source FEC payload ID.

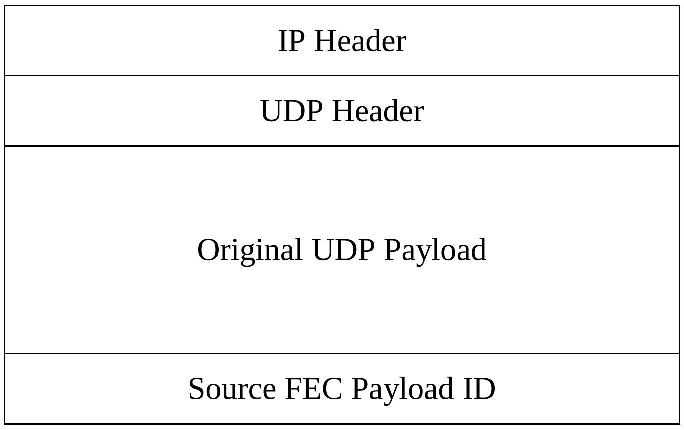


**Figure 12: Structure of the FEC packet format for FEC source packets**

The destination IP address and UDP port shall be set as indicated in the session control signalling. This ensures that the receiver can determine which protocols and FEC Payload ID formats are used for this flow. The remaining fields in the IP and UDP headers shall be set according to their specifications.

The UDP payload shall consist of the original UDP Payload followed by the Source FEC Payload ID.

The Source FEC Payload ID consists of information required for the operation of the FEC algorithm. Its construction is specified in section 8.2.2.7.

The FEC Source packets over IP and UDP are indicated to be used for a flow by using one of the SDP protocol identifiers 'UDP/MBMS-FEC/RTP/AVP', 'UDP/MBMS-FEC/RTP/SAVP' depending on the upper layer protocol RTP/AVP or RTP/SAVP respectively. If MIKEY is FEC protected and encapsulated in source packets, then it is indicated in the security description using the *fecProtection* element and the destination IP address.

### 8.2.2.5 Packet Format for Repair packets

The packet format for FEC repair packets carries, as its payload, encoding symbols generated by the FEC encoding process. The format of a FEC repair packet is depicted in figure 13. The UDP payload consists of the Repair FEC Payload ID, and one or more encoding symbols.

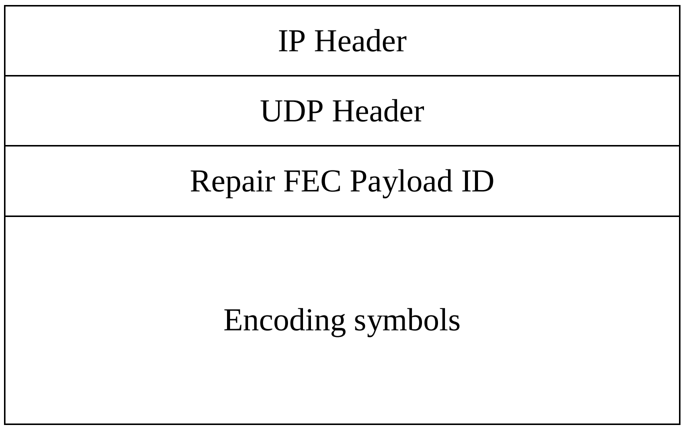


**Figure 13: RTP payload structure for repair RTP packets**

The repair packet sent over IP and UDP is indicated in the SDP using the protocol identifier 'UDP/MBMS-REPAIR'.

### 8.2.2.6 Structure of the FEC source block

This clause defines the layout of the FEC source block.

The FEC source block shall contain at least one complete UDP packet payload (i.e. excluding the IP and UDP headers), and three octets indicating the UPD flow from which the packet was taken and the length of the UDP packet. Note: this implies that no source UDP packet be larger than the length of the FEC source block minus 3.

Let

| | |
|---|---|
| $n$ | be the number of UDP packets in the source block. $n$ is determined dynamically during the source block construction process. |
| $R_i$ | denote the octets of the UDP payload of the *i*th UDP packet to be added to the source block. |
| $l_i$ | be the length of $R_i$ in octets. |
| $L_i$ | denote two octets representing the value of $l_i$ in network byte order (high order octet first). |
| $f_i$ | denote an integer 'flow ID' identifying the UPD flow from which the $i^{th}$ packet was taken |
| $F_i$ | denote a single octet representing the value of $f_i$ |
| $S_i$ | be the smallest integer such that $s_iT >= (l_i+3)$. |
| $P_i$ | denote $s_iT$-( $l_n$+3) zero octets. Note: $P_i$ are padding octets to align the start of each UDP packet with the start of a symbol. |
| $T$ | be the source symbol size in bytes. |

Then, the source block is constructed by concatenating $F_i$ ,$L_i$, $R_i$, $P_i$ for $i$ = 1, 2, ... $n$. and the source block size, S = sum {$s_iT$, $i$=1, …, $n$}.

A UDP flow is uniquely defined by an IP source and destination address and UDP source and destination port value. The assignment of Flow ID values to UDP flows is described in Section 8.2.2.11 and 8.3.1.10.

### 8.2.2.7 FEC block Construction algorithm and example (informative)

When the original UDP packet is placed into the source block, the value of the UDP flow identifier,F, followed by the value if the UDP payload length, L, are first written as a single byte and two-byte value in network byte order (i.e. with high order byte first) respectively into the first available bytes in the source block, followed by the UDP packet payload itself (i.e.not including the IP/UDP headers). Following this, if the next available byte is not the first byte of a new symbol, then padding bytes up to the next symbol boundary shall be included using the value 0 in each byte. As long as any source UDP packets remain to be placed, the procedure is repeated starting each UPD flow identifier at the start of the next encoding symbol.

An example of forming a source block is given in figure 14 below. In this example, three UDP packets of lengths 25, 51 and 102 have been placed into a source block with symbol size T = 16 bytes. The first two packets are from UDP flow 0 and the third from UDP flow 1. Each entry in Figure 14 is a byte and the rows correspond to the source symbols and are numbered from 0 to 12. $B_{i,j}$ denotes the (j+1)th byte of the (i+1)th RTP packet.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 25 | | $B_{0,0}$ | $B_{0,1}$ | $B_{0,2}$ | $B_{0,3}$ | $B_{0,4}$ | $B_{0,5}$ | $B_{0,6}$ | $B_{0,7}$ | $B_{0,8}$ | $B_{0,9}$ | $B_{0,10}$ | $B_{0,11}$ | $B_{0,12}$ |
| $B_{0,13}$ | $B_{0,14}$ | $B_{0,15}$ | $B_{0,16}$ | $B_{0,17}$ | $B_{0,18}$ | $B_{0,19}$ | $B_{0,20}$ | $B_{0,21}$ | $B_{0,22}$ | $B_{0,23}$ | $B_{0,24}$ | $B_{0,25}$ | 0 | 0 | 0 |
| 0 | 51 | | $B_{1,0}$ | $B_{1,1}$ | $B_{1,2}$ | $B_{1,3}$ | $B_{1,4}$ | $B_{1,5}$ | $B_{1,6}$ | $B_{1,7}$ | $B_{1,8}$ | $B_{1,9}$ | $B_{1,10}$ | $B_{1,11}$ | $B_{1,12}$ |
| $B_{1,13}$ | $B_{1,14}$ | $B_{1,15}$ | $B_{1,16}$ | $B_{1,17}$ | $B_{1,18}$ | $B_{1,19}$ | $B_{1,20}$ | $B_{1,21}$ | $B_{1,22}$ | $B_{1,23}$ | $B_{1,24}$ | $B_{1,25}$ | $B_{1,26}$ | $B_{1,27}$ | $B_{1,28}$ |
| $B_{1,29}$ | $B_{1,30}$ | $B_{1,31}$ | $B_{1,32}$ | $B_{1,33}$ | $B_{1,34}$ | $B_{1,35}$ | $B_{1,36}$ | $B_{1,37}$ | $B_{1,38}$ | $B_{1,39}$ | $B_{1,40}$ | $B_{1,41}$ | $B_{1,42}$ | $B_{1,43}$ | $B_{1,44}$ |
| $B_{1,45}$ | $B_{1,46}$ | $B_{1,47}$ | $B_{1,48}$ | $B_{1,49}$ | $B_{1,50}$ | $B_{1,51}$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 102 | | $B_{2,0}$ | $B_{2,1}$ | $B_{2,2}$ | $B_{2,3}$ | $B_{2,4}$ | $B_{2,5}$ | $B_{2,6}$ | $B_{2,7}$ | $B_{2,8}$ | $B_{2,9}$ | $B_{2,10}$ | $B_{2,11}$ | $B_{2,12}$ |
| $B_{2,13}$ | $B_{2,14}$ | $B_{2,15}$ | $B_{2,16}$ | $B_{2,17}$ | $B_{2,18}$ | $B_{2,19}$ | $B_{2,20}$ | $B_{2,21}$ | $B_{2,22}$ | $B_{2,23}$ | $B_{2,24}$ | $B_{2,25}$ | $B_{2,26}$ | $B_{2,27}$ | $B_{2,28}$ |
| $B_{2,29}$ | $B_{2,30}$ | $B_{2,31}$ | $B_{2,32}$ | $B_{2,33}$ | $B_{2,34}$ | $B_{2,35}$ | $B_{2,36}$ | $B_{2,37}$ | $B_{2,38}$ | $B_{2,39}$ | $B_{2,40}$ | $B_{2,41}$ | $B_{2,42}$ | $B_{2,43}$ | $B_{2,44}$ |
| $B_{2,45}$ | $B_{2,46}$ | $B_{2,47}$ | $B_{2,48}$ | $B_{2,49}$ | $B_{2,50}$ | $B_{2,51}$ | $B_{2,52}$ | $B_{2,53}$ | $B_{2,54}$ | $B_{2,55}$ | $B_{2,56}$ | $B_{2,57}$ | $B_{2,58}$ | $B_{2,59}$ | $B_{2,60}$ |
| $B_{2,61}$ | $B_{2,62}$ | $B_{2,63}$ | $B_{2,64}$ | $B_{2,65}$ | $B_{2,66}$ | $B_{2,67}$ | $B_{2,68}$ | $B_{2,69}$ | $B_{2,70}$ | $B_{2,71}$ | $B_{2,72}$ | $B_{2,73}$ | $B_{2,74}$ | $B_{2,75}$ | $B_{2,76}$ |
| $B_{2,77}$ | $B_{2,78}$ | $B_{2,79}$ | $B_{2,80}$ | $B_{2,81}$ | $B_{2,82}$ | $B_{2,83}$ | $B_{2,84}$ | $B_{2,85}$ | $B_{2,86}$ | $B_{2,87}$ | $B_{2,88}$ | $B_{2,89}$ | $B_{2,90}$ | $B_{2,91}$ | $B_{2,92}$ |
| $B_{2,93}$ | $B_{2,94}$ | $B_{2,95}$ | $B_{2,96}$ | $B_{2,97}$ | $B_{2,98}$ | $B_{2,99}$ | $B_{2,100}$ | $B_{2,101}$ | $B_{2,102}$ | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14: Source block consisting of 3 source RTP packets of lengths 26, 52 and 103 bytes.**

### 8.2.2.8 MBMS FEC scheme definition

This clause defines a FEC encoding scheme for MBMS forward error correction as defined in Annex B for the streaming delivery method. This scheme is identified by the FEC encoding ID [TBA]. It utilized the method for forming FEC source block as defined in clause 8.2.1.3. It defines two different FEC Payload ID formats, one for FEC source packets and another for FEC repair packets.

### 8.2.2.9 Source FEC Payload ID

The Source FEC payload ID is composed as follows:

| | |
|---|---|
| Source Block Number (SBN) | Encoding Symbol ID (ESI) |

**Source Block Number (SBN), (16 bits):** An integer identifier for the source block that the source data within the packet relates to.

**Encoding Symbol ID (ESI), (16 bits):** The starting symbol index of the source packet in the source block.

**Figure 15: Source FEC Payload ID**

The interpretation of the Source Block Number and Encoding Symbol Identifier is defined in Annex B.

### 8.2.2.10 Repair FEC payload ID

The structure of the Repair FEC Payload ID is as follows:

| | |
|---|---|
| Source Block Number (SBN) | Encoding Symbol ID (ESI) |
| Source Block Length (SBL) | |

**Source Block Number (SBN), (16 bits):** An integer identifier for the source block that the repair symbols within the packet relate to.
**Encoding Symbol ID (ESI), (16 bits):** integer identifier for the encoding symbols within the packet.
**Source Block Length (SBL), (16 bits):** The number of source symbols in the source block.
The interpretation of the Source Block Number, Encoding Symbol Identifier and Source Block Length is defined in Annex B.

**Figure 16: Repair FEC Payload ID**

### 8.2.2.10a FEC Object Transmission information

The FEC Object Transmission information shall consist of:

- the FEC Encoding ID
- the maximum source block length, in symbols
- the symbol size, in bytes

The symbol size and maximum source block length shall be encoded into a 4 octet field defined as follows:

| | |
|---|---|
| Symbol Size (*T*) | Maximum Source Block Length |

Symbol Size (*T*) (16 bits): The size of an encoding symbol, in bytes,

Maximum Source Block Length (16 bits): The maximum length of a source block, in symbols.

The interpretation of *T* is defined in Annex B.

The Source Block Length signalled within the Repair FEC Payload ID of any packet of a stream shall not exceed the Maximum Source Block Length signalled within the FEC Object Transmission Information for the stream.

The FEC Object Transmission Information shall be communicated as described in Section 8.2.2.14.

### 8.2.2.11 Hypothetical FEC Decoder

This clause specifies the hypothetical FEC decoder and its use to check packet stream and MBMS receiver conformance.

The hypothetical FEC decoder uses the packet stream, the transmission time of each packet, the initial buffering delay, and the SDP for the stream as inputs. The packet stream from the beginning of the FEC source block until the end of the stream shall comply with the hypothetical reference decoder as specified below when the initial buffer delay equals to the value of the min-buffer-time parameter.

The maximum buffer size for MBMS streaming is 1 Mbytes. The default hypothetical FEC decoding buffer size is equal to 1 Mbytes.

For the packet stream, the buffer occupancy level of the hypothetical FEC decoding buffer shall not exceed the value of the buf-size parameter, when it is present in the SDP, or the default FEC decoding buffer size, when the buf-size parameter is not present in the SDP. The output of the hypothetical FEC decoder shall comply with the RTP payload and decoding specifications of the media format.

The hypothetical FEC decoder operates as follows:

1) The hypothetical FEC decoding buffer is initially empty.

2) Each FEC source packet and FEC repair packet, starting from the first packet in transmission order, is inserted to a FEC source block at its transmission time. The FEC source block generation is done as specified in clause 8.2.2.5. The FEC source block resides in the hypothetical FEC decoding buffer.

3) When both the last FEC source packet and the last FEC repair packet of an FEC source block are transmitted, any elements of the FEC source block that are not original UDP packets (e.g. FEC repair packets and potential padding bytes) are removed from the hypothetical FEC decoding buffer.

4) Original UDP packets are not removed from the hypothetical FEC decoding buffer before the signaled initial buffering delay has expired. Then, the first original UDP packet in sequence number order is output and removed from the hypothetical FEC decoding buffer immediately. Each succeeding original UDP packet is output and removed when the following conditions are true:

   i. The following time (in seconds) since the removal of the previous packet has elapsed:

      8 × (size of the previous original UDP packet *including* UDP/IP header in bytes) / (1 000 × (value of "b=AS" SDP attribute for the stream))

   ii. All the packets in the same FEC source block as the original UDP packet have been transmitted.

An MBMS client shall be capable of receiving a packet stream that complies with the hypothetical FEC decoder. Furthermore, in the case of RTP packets, when an MBMS client complies with the requirements for the media decoding of the packet stream, it shall be able to de-packetize and decode the packet stream and output decoded data at the correct rate specified by the RTP timestamps of the received packet stream.

#### 8.2.2.12 FEC encoding procedures

FEC encoding shall be performed using the MBMS forward error correction code defined in Annex B.

#### 8.2.2.13 Signalling

The signalling for streaming FEC consists of several components:

- If several user services are bundled together they are indicated as a sequence of services in the User Service Bundle Description. See section 5.2.2.4
- A separate SDP describing the FEC repair stream and all the flow Ids. Referenced from the User Service Bundle Description. See section 5.2.2.6 and 8.2.2.13
- SDP protocol identifiers and attributes to indicate the usage of the source packet format, how the FEC payload ID is configured and other FEC parameters such as minimal buffering delay, for the RTP/RTCP streams. See section 8.2.2.12
- Security description extensions to indicate usage of FEC source packet format, and the FEC parameters. See section 5.2.2.3 and 8.2.2.12.

The user service description contains either a single service or several bundled services. All of the streaming delivery methods and security descriptions that are present in within the *bundleDescription* element must be considered when configuring the FEC operations. This includes RTP, RTCP and MIKEY flows. A receiver intending to perform FEC decoding to cover for packet losses shall receive all the flows that are indicated to be sent as FEC source packets, even if the flows are in a service currently not played out. A receiver intending to use FEC shall also receive the FEC repair stream as described by the FEC Repair Stream Description. The delivery method"s session description, and the security description both carry the FEC source packet configuration information: FEC encoding ID, FEC instance ID, and FEC OTI information. The FEC repair packet stream is configured using the similar methods as for the source packets, with the addition of the Flow ID information and buffer delay parameter.

### 8.2.2.13a SDP for FEC source packet streams

To indicate the presence of the FEC layer between IP/UDP and, RTP or SRTP a SDP protocol identifier is used. Instead of the normal RTP/AVP and RTP/SAVP protocol identifiers, UDP/MBMS-FEC/RTP/AVP" and "UDP/MBMS-FEC/RTP/SAVP" are defined respectively. Both these protocol identifiers shall use the FMT space rules that are used for RTP/AVP and RTP/SAVP respectively, i.e. payload types used in the RTP session is listed. The protocol identifiers are defined in Appendix C1.

The FEC parameters, FEC encoding ID, FEC instance ID and FEC-OTI-Extension information are signalled using the mechanism defined in 8.3.1.9. The 'a=FEC' SDP attribute shall be used to indicate the single definition that is used for each media component.

For MIKEY messages the service protection description is used to indicate when FEC source packet shall be used, see section 5.2.2.3. The FEC parameter used is also defined in the service protection description. As all MIKEY packets from all user services arrive on the same port, the receiver must use the destination address to separate FEC protected packets from not FEC protected packets. This requires that all MIKEY packets sent to a specific destination address are either FEC protected or not. Note that it is not possible to mix protected and non-protected packets within a single stream as there is no mechanism to determine whether they are protected or not.

### 8.2.2.14 SDP for FEC repair packet streams

The repair packet stream is indicated in SDP using a media block with the protocol identifier 'UDP/MBMS-REPAIR'. The media type shall be 'application'. The FEC parameters, FEC encoding ID, FEC instance ID, FEC-OTI-Extension information and repair parameters (min-buffer-time) are signalled using the mechanisms defined in 8.3.1.9. A single FEC declaration shall be indicated using the 'a=FEC' attribute.

The mapping of the FEC source block flow ID (see 8.2.2.5) to the destination IP address and UDP port are done using the SDP attribute 'a=mbms-flowid' defined in section 8.3.1.10.

### 8.2.2.15 Signalling example for FEC

This section contains a complete signalling example for a session using FEC with a Service description, a SDP for the streaming delivery method, a SDP for the FEC repair stream, and a security description.

The top element is the security description that

```
<?xml version="1.0" encoding="UTF-8"?>
<bundleDescription
    xmlns="www.example.com/3gppUserServiceDescription"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    fecDescriptionURI="http://www.example.com/3gpp/mbms/session1-fec.sdp">
    <userServiceDescription
        userServiceId="urn:3gpp:0010120123hotdog">
            <deliveryMethod
                sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"
                protectionDescriptionURI="http://www.example.com/3gpp/mbms/sec-descript"/>
    </userServiceDescription>
</bundleDescription>
```

The security description has the URI: http://www.example.com/3gpp/mbms/sec-descript

```
<?xml version="1.0" encoding="UTF-8"?>
<securityDescription
    xmlns="www.example.com/3gppSecurityDescription"
    xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    confidentialityProtection="true"
    integrityProtection="true"
    uiccKeyManagement="true">
    <keyManagement
        waitTime="5"
        maxBackOff="10">
        <serverURI =http://register.operator.umts/ />
        <serverURI ="http:// register2.operator.umts/" />
    </keyManagement>
    <keyId identity="<someMSKidA>" mediaFlow=FF1E:03AD::7F2E:172A:1E24:4002/>
    <keyId identity="<someMSKidB>" mediaFlow=FF1E:03AD::7F2E:172A:1E24:4004/>
    <fecProtection
        fecEncodingId="130"
```

```
            fecInstanceId="0"
            fecOtiExtension="1SCxWEMNe397m24SwgyRhg=="/>
    </securityDescription>
```

An example of how the SDP http://www.example.com/3gpp/mbms/session1.sdp could look for a session containing two media streams that are FEC protected. In this example we have assumed an audiovisual stream, using 56 kbps for video and 12 kbps for audio. In addition another 300 bits/second of RTCP packets from the source is used for the each of the sessions. Hence, the total media session bandwidth is 56+12+0.3+0.3 = 68.6 kbps.

```
v=0
o=ghost 2890844526 2890842807 IN IP4 2001:210:1:2:240:96FF:FE25:8EC9
s=3GPP MBMS Streaming SDP Example
i=Example of MBMS streaming SDP file
u=http://www.infoserver.example.com/ae600
e=ghost@mailserver.example.com
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
t=3034423619 3042462419
b=AS:62
b=TIAS: 60500
a=maxprate: 25
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=FEC-declaration:0 encoding-id=130; instance-id=0
a=FEC-OTI-extension:0 1SCxWEMNe397m24SwgyRhg==

m=video 4002 UDP/MBMS-FEC/RTP/AVP 96
b=TIAS:55000
b=RR:0
b=RS:300
a=rtpmap:96 H263-2000/90000
a=fmtp:96 profile=3;level=10
a=framesize:96 176-144
a=FEC:0
a=maxprate:15
m=audio 4004 UDP/MBMS-FEC/RTP/AVP 98
b=TIAS: 11500
b=RR:0
b=RS:300
a=rtpmap:98 AMR/8000
a=fmtp:98 octet-align=1
a=FEC:0
a=maxprate:10
```

The FEC stream used to protect the above RTP sessions and a MIKEY key stream has the below SDP (http://www.example.com/3gpp/mbms/session1-fec.sdp):

```
v=0
o=ghost 2890844526 2890842807 IN IP6 2001:210:1:2:240:96FF:FE25:8EC9
s=3GPP MBMS Streaming FEC SDP Example
i=Example of MBMS streaming SDP file
u=http://www.infoserver.example.com/ae600
e=ghost@mailserver.example.com
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
t=3034423619 3042462419
b=AS:15
a=FEC-declaration:0 encoding-id=131; instance-id=0
a=FEC-OTI-extension:0 1SCxWEMNe397m24SwgyRhg==
a=mbms-repair: 0 min-buffer-time=2600
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
m=application 4006 UDP/MBMS-REPAIR *
b=AS:15
a=FEC:0
a=mbms-apid: 1=FF1E:03AD::7F2E:172A:1E24/4002, 2=FF1E:03AD::7F2E:172A:1E24/4003,
3=FF1E:03AD::7F2E:172A:1E24/4004, 4=FF1E:03AD::7F2E:172A:1E24/4005, 5=FF1E:03AD::7F2E:172A:1E24/2269
```

## 8.3 Session description

SDP is provided to the MBMS client via a discovery/announcement procedure to describe the streaming delivery session. The SDP describes one or more RTP session part of the MBMS streaming session. The SDP shall be a correctly formed SDP according to RFC 2327 [14].

### 8.3.1 SDP Parameters for MBMS streaming session

The semantics of a Session Description of an MBMS streaming session shall include the parameters:

- The sender IP address.
- The number of media in the session.
- The destination IP address and port number for each and all of the RTP sessions in the MBMS streaming session.
- The start time and end time of the session.
- The protocol ID (i.e. RTP/AVP).
- Media type(s) and fmt-list.
- Data rate using existing SDP bandwidth modifiers.
- Mode of MBMS bearer per media.
- FEC configuration and related parameters.
- Service-language(s) per media.
- QoE Metrics (defined in clause 8.3.1.1).

#### 8.3.1.1 Sender IP address

There shall be exactly one IP source address per media description within the SDP. The IP source address shall be defined according to the source-filter attribute ("a=source-filter:") [15] for both IPv4 and IPv6 sources, with the following exceptions:

1. Exactly one source address may be specified by this attribute such that exclusive-mode shall not be used and inclusive-mode shall use exactly one source address in the <src-list>.
2. There shall be exactly one source-filter attribute per complete MBMS streaming session SDP description, and this shall be in the session part of the session description (i.e. not per media).
3. The * value shall be used for the <dest-address> subfield.

#### 8.3.1.2 Destination IP address and port number for channels

Each RTP session part of a MBMS streaming session is defined by two parameters:

- IP destination address.
- Destination port number(s).

The IP destination address shall be defined according to the "connection data" field ("c=") of RFC 2327 [14]. The destination port number shall be defined according to the <port> sub-field of the media announcement field ("m=") of RFC 2327 [14]. Multiple ports using "/" notation shall not be used. The RTCP port, if used, shall be RTP port +1.

#### 8.3.1.3 Media Description

The media description line shall be used as defined in RFC 2327 [14] for RTP. The <media> part indicates the type of media, audio, video, or text. The usage of RTP and any applicable RTP profile shall be indicated by using the <proto> field of the '*m*-line'. The one or more payload types that are being used in this RTP session are enumerated in the <fmt> part. Each payload type is declared using the "a=rtpmap" attribute according to RFC 2327 [14] and use the "a=fmtp" line when required to describe the payload format parameters.

### 8.3.1.4 Session Timing Parameters

A MBMS streaming session start and end times shall be defined according to the SDP timing field ("t=") - RFC 2327 [14].

### 8.3.1.5 Mode of MBMS bearer per media

The MBMS bearer mode declaration attribute defined in clause 7.3.2.7 may be used.

### 8.3.1.6 Service-language(s) per media

The existing SDP attribute "a=lang" is used to label the language of any language-specific media. The values are taken from RFC 3066 [73] which in turn takes language and (optionally) country tags from ISO 639 [74] and ISO 3661 [75] (e.g. "a=lang:EN-US"). These are the same tags used in the User Service Description XML.

### 8.3.1.7 Bandwidth specification

The bit-rate required by the MBMS streaming session and its media components shall be specified using both the "AS" bandwidth modifier and the "TIAS" bandwidth modifier combined with "a=maxprate" [31] on media level in the SDP. On session level the "TIAS" bandwidth modifier combined with "a=maxprate" may be used. Where the session level expresses the aggregated peak bit-rate, which may be lower than the sum of the individual media streams.

The bandwidth required for RTCP is specified by the "RR" and "RS" bandwidth modifiers (3GPP TS 26244 [32]) on media level for each RTP session. The "RR" modifier shall be included and set to 0 to specify that RTCP receiver reports are not used. The bandwidth used for RTCP sender reports shall be specified using the "RS" bandwidth modifier.

### 8.3.1.8 FEC Parameters

The FEC encoding ID and instance ID is provided using the "a=FEC-declaration" attribute defined in clause 7.3.2.8. Any OTI information for that FEC encoding ID and instance ID is provided with below defined FEC OTI attribute.

The FEC OTI attribute must be immediately preceded by the "a=FEC-declaration" attribute (and so can be session-level and media-level). The fec-ref maps the oti-extension to the FEC-declaration OTI it extends. The purpose of the oti-extension is to define FEC code specific OTI required for RTP receiver FEC payload configuration, exact contents are FEC code specific and need to be specified by each FEC code using this attribute.

The syntax for the attributes in RFC 2234 [23] is:

- sdp-fec-oti-extension-line = "a=FEC-OTI-extension:" fec-ref SP oti-extension CRLF
- fec-ref = 1*3DIGIT (the SDP-internal identifier for the associated FEC-declaration).
- oti-extension = base64
- base64 = *base64-unit [base64-pad]
- base64-unit = 4base64-char
- base64-pad = 2base64-char "==" / 3base64-char "="
- base64-char = ALPHA / DIGIT / "+" / "/"

To provide the FEC repair packets with additional, non FEC specific parameters, a session and media level SDP attribute is defined.

sdp-fec-parameter-line = 'a=mbms-repair: 0*1SP fec-ref  SP parameter-list CRLF

parameter-list = parameter-spec *(1*SP parameter-spec)

parameter-spec = name '=' value;

name = 1*(ALPHA / DIGIT)

value = 1*(safe) ; safe defined in RFC 2327

Currently one FEC non code-specific parameter is defined:

**min-buffer-time**: This FEC buffering parameter specifies the minimum receiver buffer time (delay) needed to ensure that FEC repair has time to happen regardless of the FEC source block of the stream from which the reception starts. The value is in milliseconds and represents the wallclock time between the reception of the first FEC source or repair packet of a FEC source block, whichever is earlier in transmission order, and the wallclock time when media decoding can safely start.

The parameters name and value is defined in ABNF as follows:

Min-buffer-time-parameter-name = 'min-buffer-time'

Min-buffer-time-parameter-value = 1*8DIGIT ;Wallclock time in milliseconds.

The FEC declaration and FEC OTI information utilized in a specific source or repair packet is indicated using the FEC-ref number in the a=fec lines as described in clause 8.2.2.12 and 8.2.2.13.

#### 8.3.1.9 FEC Flow ID attribute

To indicate the mapping between destination IP address and UDP port number and FEC source block flow IDs, the 'a=mbms-flowid' SDP attribute is defined. Each flowID that may be used in source block within the bundled sessions shall be included. It is a media level attribute that shall be present in any SDP media block using the 'UDP/MBMS-REPAIR' protocol identifier.

The syntax for the attributes in ABNF [23] is:

Sdp-mbms-flowid-attr = "a=mbms-flowid:" *WSP flow-id-spec *("," *WSP flow-id-spec) CRLF

flow-id-spec = flowID "=" address-spec "/" port-spec

address-spec = multicast-address ; As defined by RFC 3266

port-spec  = 1*5DIGIT

### 8.3.2 SDP Example for Streaming Session

Here is a full example of SDP description describing a FLUTE session:

```
v=0
o=ghost 2890844526 2890842807 IN IP4 192.168.10.10
s=3GPP MBMS Streaming SDP Example
i=Example of MBMS streaming SDP file
u=http://www.infoserver.example.com/ae600
e=ghost@mailserver.example.com
c=IN IP6 FF1E:03AD::7F2E:172A:1E24
t=3034423619 3042462419
b=AS:77
a=mbms-mode:broadcast 1234
a=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9
a=FEC-declaration:0 encoding-id=130; instance-id=0
a=FEC-OTI-extension:0 1SCxWEMNe397m24SwgyRhg==
m=video 4002 UDP/MBMS-FEC/RTP/AVP 96
b=TIAS:62000
b=RR:0
b=RS:600
a=maxprate:17
a=rtpmap:96 H264/90000
a=fmtp:96 profile-level-id=42A01E; packetization-mode=1; sprop-parameter-sets=Z0IACpZTBYmI,aMljiA==

m=audio 4004 UDP/MBMS-FEC/RTP/AVP 98
b=TIAS:15120
b=RR:0
b=RS:600
a=maxprate:10
a=rtpmap:98 AMR/8000
a=fmtp:98 octet-align=1
a=FEC:0
```

#### 8.3.2.1 SDP Description for QoE Metrics

Similar as in 3GPP TS 26.234 [47], an SDP attribute for QoE, which can be used either at session or media level, is defined below in RFC 2234 [23] based on RFC 2327 [14]:

- QoE-Metrics-line = "a" "=" "3GPP-QoE-Metrics:" att-measure-spec *("," att-measure-spec)) CRLF
- att-measure-spec = Metrics ";" Sending-rate [";" Measure-Range] *([";" Parameter-Ext])
- Metrics = "metrics" "=" "{"Metrics-Name *("," Metrics-Name) " }"
- Metrics-Name = 1*((0x21..0x2b) / (0x2d..0x3a) / (0x3c..0x7a) / 0x7c / 0x7e) ;VCHAR except ";", ",", "{" or "}"
- Sending-Rate = "rate" "=" 1*DIGIT / "End"
- Measure-Range = "range" "=" Ranges-Specifier
- Parameter-Ext = (1*DIGIT ["." 1*DIGIT]) / (1*((0x21..0x2b) / (0x2d..0x3a) / (0x3c..0x7a) / 0x7c / 0x7e))
- Ranges-Specifier = as defined in RFC 2326 [76].

An MBMS server uses this attribute to indicate that QoE metrics are supported and shall be used if also supported by the MBMS client. When present at session level, it shall only contain metrics that apply to the complete session. When present at media level, it shall only contain metrics that are applicable to individual media.

The "Metrics" field contains the list of names that describes the metrics/measurements that are required to be reported in a MBMS session (see clause 8.4). The names that are not included in the "Metrics" field shall not be reported during the session.

In this version of the specification, the "Sending-Rate" shall be set to the value "End", which indicates that only one report is sent at the end of the MBMS session.

The optional "Measure-Range" field, if used, shall define the time range in the stream for which the QoE metrics will be reported. There shall be only one range per measurement specification. The range format shall be any of the formats allowed by the media. If the "Measure-Range" field is not present, the corresponding (media or session level) range attribute in SDP shall be used. If SDP information is not present, the metrics range shall be the whole session duration.

## 8.4 Quality of Experience

### 8.4.1 General

The MBMS Quality of Experience (QoE) metrics feature is optional for both MBMS streaming server and MBMS client, and shall not disturb the MBMS service. An MBMS Server that supports the QoE metrics feature shall activate the gathering of client QoE metrics with SDP as described in clause 8.3.1.1 and via the reception reporting procedure as described in clause 9.4. An MBMS client supporting the feature shall perform the quality measurements in accordance to the measurement definitions, aggregate them into client QoE metrics and report the metrics to the MBMS server using the content reception reporting procedure. The way the QoE metrics are processed and made available is out of the scope of the present document.

### 8.4.2 QoE Metrics

An MBMS client should measure the metrics at the transport layer after FEC decoding (if FEC is used), but may also do it at the application layer for better accuracy.

The reporting period for the metrics is the whole streaming duration. This duration may be less than the session duration, because of late joiners or early leavers. The reporting period shall not include any voluntary event that impacts the actual play, such as pause, or any buffering or freezes/gaps caused by them.

The following metrics shall be derived by the MBMS client implementing QoE. All the metrics defined below are only applicable to at least one of audio, video, speech and timed text media types, and are not applicable to other media types such as synthetic audio, still images, bitmap graphics, vector graphics, and text. Any unknown metrics shall be ignored by the client and not included in any QoE report. Among the QoE metrics, corruption duration, successive loss of RTP packets, frame-rate deviation and jitter duration are of media level, whereas initial buffering duration and rebuffering duration are of session level.

### 8.4.2.1 Corruption duration metric

Corruption duration, M, is the time period from the NPT time of the last good frame before the corruption, to the NPT time of the first subsequent good frame or the end of the reporting period (whichever is sooner). A corrupted frame may either be an entirely lost frame, or a media frame that has quality degradation and the decoded frame is not the same as in error-free decoding. A good frame is a "completely received" frame X that, either:

- it is a refresh frame (does not reference any previously decoded frames AND where none of the subsequently decoded frames reference any frames decoded prior to X); or
- does not reference any previously decoded frames; or
- references previously decoded "good frames".

"Completely received" means that all the bits are received and no bit error has occurred.

Corruption duration, M, in milliseconds can be calculated as below:

a) M can be derived by the client using the codec layer, in which case the codec layer signals the decoding of a good frame to the client. A good frame could also be derived by error tracking methods, but decoding quality evaluation methods shall not be used.

b) In the absence of information from the codec layer, M should be derived from the NPT time of the last frame before the corruption and N, where N is optionally signalled from MBMS streaming server (via SDP) to the MBMS client and represents the maximum duration between two subsequent refresh frames in milliseconds.

c) In the absence of information from the codec layer and if N is not signalled, then M defaults to ∞ (for video) or to one frame duration (for audio), or the end of the reporting period (whichever is sooner).

The optional parameter N as defined in point b is used with the "Corruption_Duration" parameter. Another optional parameter T is defined to indicate whether the client uses error tracking or not. The value of T shall be set by the client via reception reporting (clause 9.5.2) as on or off. The syntax for N to be included in the "att-measure-spec" (clause 8.3.1.1) is as follows:

- N = "N" "=" 1*DIGIT

In MBMS reception reporting will be done only once at the end of streaming, hence all the occurred corruption durations are summed up over the period of the stream as the value *TotalCorruptionDuration*. The unit of this metrics is expressed in milliseconds. The number of individual corruption events over the stream duration are summed up in the value *NumberOfCorruptionEvents.* These two values are reported by the MBMS client as part of the reception report (clauses 9.4.6 and 9.5.2).

### 8.4.2.2 Rebuffering duration metric

Rebuffering is defined as any stall in playback time due to any involuntary event at the client side.

The syntax for the metric "Rebuffering_Duration" for the QoE-Feedback header is as defined in clause 8.3.1.1.

Rebuffering starts at the NPT time of the last played frame before the occurrence of the rebuffering.

In MBMS reception reporting will be done only once at the end of streaming, hence all the occurred rebuffering durations are summed up over the period of the stream as the value *TotalRebufferinfDuration*. The unit of this metrics is expressed in seconds, and can be a fractional value. The number of individual rebuffering events over the stream duration are summed up in the value *NumberOfRebufferingEvents.* These two values are reported by the MBMS client as part of the reception report (clauses 9.4.6 and 9.5.2).

#### 8.4.2.3 Initial buffering duration metric

Initial buffering duration is the time from receiving the first RTP packet until playing starts.

The syntax for the "Initial_Buffering_Duration" is as defined in clause 8.3.1.1.

If the reporting period is shorter than the "Initial_Buffering_Duration" then the MBMS client should send this parameter for the reporting period as long as it observes it. The metric value indicates the initial buffering duration where the unit of this metrics is expressed in seconds, and can be a fractional value. There can be only one measure and it can only take one value. "Initial_Buffering_Duration" is a session level parameter.This value is reported by the MBMS client as part of the reception report (clauses 9.4.6 and 9.5.2).

#### 8.4.2.4 Successive loss of RTP packets

The metric "Successive_Loss" indicates the number of RTP packets lost in succession (excluding FEC packets) per media channel.

The syntax for the metrics "Successive_Loss" is as defined in clause 8.3.1.1.

In MBMS reception reporting will be done only once at the end of streaming, hence all the number of successively lost RTP packets are summed up over the period of the stream as the value *TotalNumberofSuccessivePacketLoss*. The unit of this metric is expressed as an integer equal to or larger than 1. The number of individual events over the stream duration are summed up in the value *NumberOfSuccessiveLossEvents.* These two values are reported by the MBMS client as part of the reception report (clauses 9.4.6 and 9.5.2).

#### 8.4.2.5 Frame rate deviation

Frame rate deviation indicates the playback frame rate information. Frame rate deviation happens when the actual playback frame rate during a reporting period is deviated from a pre-defined value.

The actual playback frame rate is equal to the number of frames played during the reporting period divided by the time duration, in seconds, of the reporting period.

The parameter FR that denotes the pre-defined frame rate value is used with the "Framerate_Deviation" parameter in the "3GPP-QoE-Metrics" attribute. The value of FR shall be set by the server. The syntax for FR to be included in the "att-measure-spec" (clause 8.3.1.1) is as follows:

- FR = "FR" "=" 1*DIGIT "." 1*DIGIT

The syntax for the metric "Framerate_Deviation" is defined in clause 8.3.1.1.

For the Metrics-Name "Framerate_Deviation", the value field indicates the frame rate deviation value that is equal to the pre-defined frame rate minus the actual playback frame rate. This metric is expressed in frames per second, and can be a fractional value, and can be negative. This value is reported by the MBMS client as part of the reception report (clauses 9.4.6 and 9.5.2).

#### 8.4.2.6 Jitter duration

Jitter happens when the absolute difference between the actual playback time and the expected playback time is larger than a pre-defined value, which is 100 milliseconds. The expected time of a frame is equal to the actual playback time of the last played frame plus the difference between the NPT time of the frame and the NPT time of the last played frame.

The syntax for the metric "Jitter_Duration" is defined in clause 8.3.1.1.

In MBMS reception reporting will be done only once at the end of streaming, hence all the Jitter_Durations are summed up over the period of the stream as the value *TotalJitterDuration*. The unit of this metrics is expressed in seconds, and can be a fractional value. The number of individual events over the stream duration are summed up in the value *NumberOfJitterEvents.* These two values are reported by the MBMS client as part of the reception report (clauses 9.4.6 and 9.5.2).

### 8.4.3 Example metrics initiation with SDP

This following example shows the syntax of the SDP attribute for QoE metrics. The session level QoE metrics description (Initial buffering duration and rebufferings) are to be monitored and reported only once at the end of the session. Also video specific description of metrics (corruptions) are to be monitored and reported at the end from the beginning of the stream until the time 40s. Finally, audio specific description of metrics (corruptions) is to be monitored and reported at the end from the beginning until the end of the stream.

SDP example:

```
v=0
o=- 3268077682 433392265 IN IP4 63.108.142.6
s=QoE Enabled Session Description Example
e=support@foo.com
c=IN IP4 0.0.0.0
t=0 0
a=range:npt=0-83.660000
a=3GPP-QoE-Metrics:{Initial_Buffering_Duration,Rebuffering_Duration};rate=End
a=control:*
m=video 0 RTP/AVP 96
b=AS:28
a=3GPP-QoE-Metrics:{Corruption_Duration};rate=End;range:npt=0-40
a=control:trackID=3
a=rtpmap:96 MP4V-ES/1000
a=range:npt=0-83.666000
a=fmtp:96profile-level-id=8;config=000001b008000001b50900012000
m=audio 0 RTP/AVP 98
b=AS:13
a=3GPP-QoE-Metrics:{Corruption_Duration};rate=End
a=control:trackID=5
a=rtpmap:98 AMR/8000
a=range:npt=0-83.660000
a=fmtp:98 octet-align=1
a=maxptime:200
```

# 9 Associated delivery procedures

## 9.1 Introduction

Associated delivery procedures describe general procedures, which start before, during or after the MBMS data transmission phase. They provide auxiliary features to MBMS user services in addition, and in association with, MBMS delivery methods and their sessions. Those procedures that shall only be permitted after the MBMS Data transmission phase may also be described as post-delivery procedures.

To enable future backwards compatibility beyond 3GPP MBMS release 6 specifications, clause 9specifies generic and extensible techniques for a potentially wide range of associated delivery procedures.

Clauses 9.3 and 9.4 the associated delivery procedures that are included in release 6, and are initiated only after an MBMS data transmission phase.

The present document describes these associated delivery procedures:

- File repair, for post-delivery repair of files initially delivered as part of an MBMS download session.
- Content reception reporting of files delivered to an MBMS UE.

These procedures are enabled by establishing a point-to-point connection; and using the MBMS session parameters, received during User Service Discovery/Announcement, to communicate the context (e.g. file and session in question) to the network and the MBMS sender infrastructure. To avoid network congestion in the uplink and downlink directions, and also to protect servers against overload situations, the associated delivery procedures from different MBMS UEs shall be distributed over time and resources (network elements).

An instance of an "associated procedure description" is an XML file that describes the configuration parameters of one or more associated delivery procedures.

MBMS Download receivers shall support the file repair procedure as defined in clause 9.3.

MBMS Download receivers shall support the reception reporting procedure as defined in clause 9.4.

MBMS Streaming receivers shall support reception reporting procedures (StaR and StaR-all report types) as defined in clause 9.4.

## 9.2 Associated Procedure Description

An associated procedure description instance (configuration file) for the associated delivery procedures may be delivered to the MBMS clients:

- during a User Service Discovery / Announcement prior to the MBMS Download delivery session along with the session description (out-of-band of that session); or
- in-band within a MBMS Download delivery session.

The most recently delivered configuration file (i.e. the one with the highest version number - as given from the envelope, see clause 5.2.3.3) shall take priority, such that configuration parameters received prior to, and out-of-band of, the download session they apply to are regarded as "initial defaults", and configuration parameters received during, and in-band with the download session, overwrite the earlier received parameters. Thus, a method to update parameters dynamically on a short time-scale is provided but, as would be desirable where dynamics are minimal, is not mandatory.

During the User Service Discovery / Announcement Procedure, the associated procedure description instance is clearly identified using a URI, to enable UE cross-referencing of in and out-of-band configuration files.

The MIME application type "application/mbms-associated-procedure-parameter" identifies associated delivery procedure description instances (configuration files).

In XML, each associated delivery procedure entry shall be configured using an "`associatedProcedureDescription`" element. All configuration parameters of one associated delivery procedure are contained as attributes of an "`associatedProcedureDescription`" element. The elements (e.g. "`postFileRepair`" and "`postReceptionReport`") of an "`associatedProcedureDescription`" element identify which associated procedure(s) to configure.. The associated delivery procedure description is specified formally as an XML schema in clause 9.5.1.

## 9.3 File Repair Procedure

### 9.3.1 Introduction

The purpose of the File Repair Procedure is to repair lost or corrupted file fragments from the MBMS download data transmission. When in multicast/broadcast environment, scalability becomes an important issue as the number of MBMS clients grows. Three problems must generally be avoided:

- Feedback implosion due to a large number of MBMS clients requesting simultaneous file repairs. This would congest the uplink network channel.
- Downlink network channel congestion to transport the repair data, as a consequence of the simultaneous clients requests.
- File repair server overload, caused again by the incoming and outgoing traffic due to the clients' requests arriving at the server, and the server responses to serve these repair requests.

The three problems are interrelated and must be addressed at the same time, in order to guarantee a scalable and efficient solution for MBMS file repair.

The principle to protect network resources is to spread the file repair request load in time and across multiple servers.

The MBMS client:

1. Identifies the end of transmission of files or sessions.
2. Identifies the missing data from an MBMS download.
3. Calculates a random *back-off time* and selects a file repair server randomly out of a list.

4. Sends a *repair request* message to the selected file repair server at the calculated time.

When a MBMS download session of repair data is configured in the associated delivery descriptions, a MBMS client should wait for repair data in the defined MBMS download session on its MBMS bearer - except where the UE is prevented from doing so due to limited simultaneous context activation capability.

Then the file repair server:

1. Responds with a *repair response* message either containing the requested data, redirecting the client to an MBMS download session, redirecting the client to another server, or alternatively, describing an error case.

The BM-SC may also send the repair data on a MBMS bearer (possibly the same MBMS bearer as the original download) as a function of the repair process.

The random distribution, in time, of *repair request* messages enhances system scalability to the total number of such messages the system can handle without failure.

### 9.3.2 Identification of End of Transmission for MBMS Download Delivery

FLUTE File Delivery Table (FDT) Instances include an "expires" attribute, which defines the expiration time of the FDT instance. The sender must use an expiry time relative to its sender current time. The Sender Current Time header field shall be present in all FLUTE packets containing data of an FDT Instance. According to RFC 3926 [9], "the receiver SHOULD NOT use a received FDT Instance to interpret packets received beyond the expiration time of the FDT Instance".

The MBMS UE determines the end-of-transmission for the MBMS download delivery based on the expiration time of the FDT instance and any end-of-object (B-flag) and end-of-session (A-flag, and SDP end time) information available.

When a particular file (URI) is present in several FDT Instances with different TOI values, then the expiration time of the FDT Instances, which contain the highest TOI value of that file determines the end of transmission time for that file. A UE shall only determine transmission completeness for a file for the most up-to-date instance of the file (the file instance/version with the highest/most-up-to-date TOI) - and shall not use FDT Instance expiry time to determine transmission completeness for any other (TOI) instances of a file (fileURI).

NOTE 1: The intention of this clause is to just start the Associated Delivery Procedure back-off timer for the latest version of a file.

When a particular file (URI) is present in more than one FDT Instance with the same TOI value, then the end of transmission time is defined by the expiration time of the last FDT Instance to expire.

If an FDT Instance is received describing the file after this time (giving an FDT Instance expiry time in the future and the same or a higher TOI value) the UE shall determine that the transmission of the file is incomplete - i.e. that more packets may arrive by the MBMS download session for that file, 'forgetting' its previous file transmission complete determination.

NOTE 2: This effectively resets and stops any running timers already initiated for an associated delivery procedure for that file.

If the MBMS UE receives an end-of-object packet (with FLUTE header B flag set true) the MBMS UE shall determine that the transmission of that object is complete, and shall interpret that as file transmission complete if no, more recent, TOIs are described for the same file (URI) in any received and unexpired FDT Instance(s).

If the MBMS UE determines that the download session is complete (as specified in clause 9.4.2) then it shall interpret this also that all the transmissions of all files (and TOIs) described by all FDT Instances, received from that session, are complete.

### 9.3.3 Identification of Missing Data from an MBMS Download

The session description and (in-band) the MBMS download delivery protocol, FLUTE, provide the client with sufficient information to determine the source block and encoding symbol structure of each file. From this a client is able to determine which source symbols should have been transmitted but have not been received. The client is also able to determine the number of symbols it has received for each source block of each file, and thus the number of further symbols required to decode the block.

Thus, an MBMS client is able to identify any source symbols lost in transmission, and the number (and ESI values where appropriate) of required source and/or repair symbols that would complete the reconstruction of a source block (of a file).

Where the MBMS FEC scheme is used, the MBMS client shall consider already received repair symbols when making the determination of the further symbols required. In this case, the client should either:

- - identify a minimal set of specific symbols that, combined with the already received symbols, allow the MBMS FEC decoder to recover the file, or
- - identify a number, *r*, of symbols such that reception of *r* previously unreceived symbols will allow the MBMS FEC decoder to recover the file.

### 9.3.4 Back-off Timing the Procedure Initiation Messaging for Scalability

This clause describes a *back-off mode* for MBMS download to provide information on when a receiver, that did not correctly receive some data from the MBMS sender during a transmission session, can start a request for a repair session. In the following it is specified how the information and method a MBMS client uses to calculate a time (*back-off time*), instance of the back-off mode, to send a file repair message to the MBMS server.

The back-off mode is represented by a *back-off unit*, a *back-off value,* and a *back-off window*. The two latter parameters describe the back-off time used by the MBMS client.

*The back-off unit* (in the time dimension) defaults to *seconds* and it is not signalled.

The *back-off time* shall be given by an *offset time* (describing the back-off value) and a *random time period* (describing the back-off window) as described in the following clauses.

An MBMS client shall generate random or pseudo-random time dispersion of *repair requests* to be sent from the receiver (MBMS client) to the sender (MBMS server). In this way, the repair request is delayed by a pre-determined (random) amount of time.

The back-off timing of *repair request* messages (i.e. delaying the sending of *repair requests* at the receiver) enhances system scalability to the total number of such messages the system can handle without failure.

#### 9.3.4.1 Offset time

The *OffsetTime* refers to the repair request suppression time to wait before requesting repair, or in other words, it is the time that a MBMS client shall wait after the end of the MBMS data transmission to start the file repair procedure. An associated procedure description instance shall specify the wait time (expressed in *back-off unit*) using the "offset-time" attribute.

#### 9.3.4.2 Random Time Period

The *Random Time Period* refers to the time window length over which a MBMS client shall calculate a *random time* for the initiation of the file repair procedure. The method provides for statistically uniform distribution over a relevant period of time. An associated procedure description instance shall specify the wait time (expressed in *back-off unit*) using the "random-time-period" attribute.

The MBMS client shall calculate a uniformly distributed *Random Time* out of the interval between 0 and *Random Time Period*.

#### 9.3.4.3 Back-off Time

The sending of the file *repair request* message shall start at *Back-off Time = offset-time + Random Time*, and this calculated time shall be a relative time after the MBMS data transmission. The MBMS client shall not start sending the repair request message before this calculated time has elapsed after the initial transmission ends.

#### 9.3.4.4 Reset of the Back-off Timer

The reception of an updated (higher version number) associatedDeliveryProcedureDescription and/or an updated sessionDescription shall overwrite the timer parameters used in the back-off algorithm. Except in the case that the offset-time, random-time-period and session end time parameters are identical to the earlier version; the back-off time shall be recalculated. For currently running timers this requires a reset.

### 9.3.5 File Repair Server Selection

#### 9.3.5.1 List of Server URIs

A list of file repair servers is provided by a list of server URIs as attributes of the Associated Delivery procedure description. These attributes and elements specify URIs of the file repair servers. Server URIs may also be given as IP addresses, which may be used to avoid a requirement for DNS messaging. The file repair server URIs of a single associated delivery procedure description shall be of the same type, e.g. all IP addresses of the same version, or all domain names. The number of URIs is determined by the number of "serverURI" elements, each of which shall be a child-element of the "procedure" element. The "serverURI" element provides the references to the file repair server via the "xs:anyURI" value. At least one "serverURI" element shall be present.

#### 9.3.5.2 Selection from the Server URI List

The MBMS client randomly selects one of the server URIs from the list, with uniform distribution.

### 9.3.6 File Repair Request Message

Once missing file data is identified, the MBMS client sends one or more messages to a file repair server requesting transmission of data that allows recovery of missing file data. All file repair requests and repair responses for a particular MBMS transmission shall take place in a single TCP session using the HTTP protocol (RFC 2616 [18]). The repair request is routed to the file repair server IP address resolved from the selected "serverURI".

The timing of the opening of the TCP connection to the server, and the first repair request, of a particular MBMS client is randomized over a time window as described in clause 9.3.2. If there is more than one repair request to be made these are sent immediately after the first.

When a MBMS client identifies symbols in repair requests these shall be source symbols, and should include all the missing source symbols of the relevant source block. Note, these represent information for the file repair server and the BM-SC may use these and/or redundant symbols in providing the necessary repair data.

#### 9.3.6.1 File Repair Request Message Format

After the MBMS download session, the receiver identifies a set of FLUTE encoding symbols which allows recovery of the missing file data and requests for their transmission in a file repair session. Specific encoding symbols are uniquely identified by the combination (URI, SBN, ESI).

The file repair request shall include the URI of the file for which it is requesting the repair data. URI is required to uniquely identify the file (resource) and is found from the download delivery method (the FLUTE FDT Instances describe file URIs). The (SBN, ESI) pair uniquely identifies an encoding symbol. For completely missed files, a Repair Request may give only the URI of the file.

The client makes a file repair request using the HTTP (RFC 2616 [18]) request method GET. If specific symbols are requested, the (SBN, ESI) of requested encoding symbols are URL-encoded (RFC 1738 [19]) and included in the HTTP GET request. If a number of previously unreceived symbols are requested for a specific Source Block, then the SBN is provided along with the ESI of the symbol which is subsequent in the symbol sequence to the latest received symbol for that source block and the number of symbols requested.

For example, assume that in a FLUTE session a 3gp file with URI = www.example.com/news/latest.3gp was delivered to an MBMS client. After the FLUTE session, the MBMS client recognized that it did not receive two packets with SBN = 5, ESI = 12 and SBN=20, ESI = 27. Then the HTTP GET request is as follows:

**GET** www.example.com/news/latest.3gp?mbms-rel6-FLUTE-repair&SBN=5;ESI=12&SBN=20;ESI=27 **HTTP/1.1**

A file repair session shall be used to recover the missing file data from a single MBMS download session only. If more than one file were downloaded in a particular MBMS download session, and, if the MBMS client needs repair data for more than one file received in that session, the MBMS client shall send separate HTTP GET requests for each file.

An HTTP client implementation might limit the length of the URL to a finite value, for example 256 bytes. In the case that the length of the URL-encoded (SBN, ESI) data exceeds this limit, the MBMS client shall distribute the URL-encoded data into multiple HTTP GET requests.

In any case, all the HTTP GETs of a single file repair session shall be performed within a single TCP session and they shall be performed immediately one after the other.

In the following, we give the details of the syntax used for the above request method in ABNF.

In this case an HTTP GET with a normal query shall be used to request the missing data.

The general HTTP URI syntax is as follows RFC 2616 [18]:

- http_URL = "http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]

Where, for MBMS File Repair Request:

- query = application *( "&" sbn_info)
- application = "mbms-rel6-flute-repair"
- sbn_info = "SBN=" sbn_range
- sbn_range = ( sbnA [ "-" sbnZ ] ) / ( sbnA [ ";" esi_info] )
- esi_info = ( "ESI=" ((esi_range *( "," esi_range ) ) ) / (esiA '+' number_symbols)
- esi_range = esiA [ "-" esiZ ]
- sbnA = 1*DIGIT; the SBN, or the first of a range of SBNs
- sbnZ = 1*DIGIT; the last SBN of a range of SBNs
- esiA = 1*DIGIT; the ESI, or the first of a range of ESIs
- esiZ = 1*DIGIT; the last ESI of a range of ESIs

Thus, the following symbols adopt a special meaning for MBMS FLUTE: ? - + , ; & =

One example of a query on encoding symbol 34 of source block 12 of a music file "number1.aac" is:

- http://www.operator.com/greatmusic/number1.aac?mbms-rel6-flute-repair&SBN=12;ESI=34

For messaging efficiency, the formal definition enables several contiguous and non-contiguous ranges to be expressed, as well as a number of symbols with ESIs of a given value or above in a single query:

- A symbol of a source block (like in the above example).
- A range of symbols for a certain source block (e.g. ...&SBN=12;ESI=23-28).
- A number of symbols with ESIs of a given value or above (e.g. …&SBN=12;ESI=120+10).
- A list of symbols for a certain source block (e.g. ...&SBN=12;ESI=23,26,28).
- All symbols of a source block (e.g. ...&SBN=12).

- All symbols of a range of source blocks (e.g. ...&SBN=12-19).
- non-contiguous ranges (e.g.1. ...&SBN=12;ESI=34&SBN=20;ESI=23 also, e.g. 2. ...&SBN=12-19&SBN=28;ESI=23-59&SBN=30;ESI=101).

## 9.3.7 File Repair Response Message

Once the MBMS file repair server has assembled a set of encoding symbols that contain sufficient data to allow the UE to reconstruct the file data from a particular file repair request, the MBMS file repair server sends one message to the UE. Each file repair response occurs in the same TCP and HTTP session as the repair request that initiated it.

An MBMS client shall be prepared for any of these 4 response scenarios:

- The server returns a repair response message where a set of encoding symbols forms an HTTP payload as specified below.
- The server redirects the client to a broadcast/multicast delivery (an MBMS download session).
- The server redirects the client to another file repair server (if a server is functioning correctly but is temporarily overloaded).
- An HTTP error code is returned (note that clause 9.3.6 describes the case of no server response).

For (reasonably) uniformly distributed random data losses, immediate point-to-point HTTP delivery of the repair data will generally be suitable for all clients. However, broadcast/multicast delivery of the requested data may be desirable in some cases:

- A repeat MBMS download (all or part of the files from a download session) is already scheduled and the BM-SC prefers to handle repairs after that repeat MBMS download.
- Many UEs request download data (over a short period of time) indicating that broadcast/multicast delivery of the repaired data would be desirable.

In this case a redirect to the broadcast/multicast repair session for UEs that have made a repair request would be advantageous.

### 9.3.7.1 File Repair Response Messages Codes

In the case that the file repair server receives a correctly formatted repair request which it is able to understand and properly respond to with the appropriate repair data, the file repair server shall attempt to serve that request without an error case.

For a direct point-to-point HTTP response with the requested data, the file response message shall report a 200 OK status code and the file repair response message shall consist of HTTP header and file repair response payload (HTTP payload), as defined in clause 9.3.5.2. If the client receives a 200 OK response with fewer than all the quantity of requested symbols it shall assume that the file repair server wishes the missing symbols to be requested again (due to its choice or inability to deliver those symbols with this HTTP response).

For a redirect case the file repair server uses the HTTP response status code 302 (Found - Redirection) to indicate to the UE that the resource (file repair data) is temporarily available via a different URI. The temporary URI is given by the Location field in the HTTP response. In the case of a redirect to another file repair server, this temporary URI shall be the URL of that repair server.

In the case of a redirect to a broadcast/multicast delivery, the temporary URI shall be the URI of the Session Description (SDP file) of the broadcast/multicast (repair) session as described in clause 9.3.5.3.Other HTTP status codes (RFC 2616 [18]) shall be used to support other cases. Other cases may include server errors, client errors (in the file repair request message) and server overload.

### 9.3.7.2 File Repair Response Message Format for HTTP Carriage of Repair Data

The file repair response message consists of HTTP header and file repair response payload (HTTP payload).

The HTTP header shall provide:

- HTTP status code, set to 200 OK.
- Content type of the HTTP payload (see below).
- Content transfer encoding, set to binary.

The Content-Type shall be set to "application/simpleSymbolContainer", which denotes that the message body is a simple container of encoding symbols as described below.

This header is as follows:

- HTTP/1.1 200 OK
- Content-Type: application/simpleSymbolContainer
- Content-Transfer-Encoding: binary

NOTE: Other HTTP headers (RFC 2616 [18]) may also be used but are not mandated by this mechanism.

Encoding symbols are included in the response in groups. Each group is preceded by an indication of the number of symbols within the group and an FEC Payload ID coded according to the FEC scheme used for the original file delivery session. The FEC Payload ID identifies all the symbols in the group in the same way that the FEC Payload ID of an FEC source or repair packet identifies all the symbols in the packet. The file repair response payload is constructed by including each FEC Payload ID and Encoding Symbol group one after another (these are already byte aligned). The order of these pairs in the repair response payload may be in order of increasing SBN, and then increasing ESI, value; however no particular order is mandated.

A single HTTP repair response message shall contain, at the most, the same number of symbols as requested by the respective HTTP repair request message.

The UE and file repair server already have sufficient information to calculate the length of each encoding symbol and each FEC Payload ID. All encoding symbols are the same length; with the possible exception of the last source encoding symbol in the repair response. All FEC Payload IDs are the same length for one file repair request-response as a single FEC Scheme is used for a single file.

**Figure 17: deleted**

Figure 18 illustrates the complete file repair response message format (box sizes are not indicative of the relative lengths of the labelled entities).

| HTTP Header | | |
|---|---|---|
| Length Indicator | FEC Payload ID | Encoding Symbols |
| Length Indicator | FEC Payload ID | Encoding Symbols |
| | | |
| Length Indicator | FEC Payload ID | Encoding Symbols |

**Length Indicator** (2 bytes): indicates the number of encoding symbols in the group (in network byte order, i.e. high order byte first)

**FEC Payload ID**: indicates which encoding symbols are included in the group. The format and interpretation of the FEC Payload ID are dependent on the FEC Scheme in use.

**Encoding Symbols**: contain the encoding symbols. All the symbols shall be the same length.

**Figure 18: File Repair Response Message Format**

### 9.3.7.3 File Repair Response for Broadcast/Multicast of Repair Data

Details of how a file repair server decides, or is instructed, to use broadcast/multicast repair instead of point-to-point over HTTP are implementation specific and beyond the scope of the present document.

Prior to the decision to use broadcast/multicast repair, each repair response shall be provided by HTTP according to clause 9.3.5.2.

The file repair server uses the HTTP response status code 302 (Found - Redirection) to indicate to the UE that the resource (file repair data) is temporarily available via a different URI. The temporary URI is given by the Location field in the HTTP response and is the URI of the Session Description (SDP file) of the broadcast/multicast repair session.

Where feasible, it is recommended that the same download session that delivered the original data use used for the broadcast/multicast repair. If this conflicts with the session end time limit of the Session Description then a new version of the Session Description shall be sent with an updated (extended) session end time. This shall be sent in-band of that download session.

In some cases this may not be feasible and an different (possibly new) download session may be defined for the repair.

The SDP file for broadcast/multicast repair session may be carried as payload (entity-body) in the HTTP response - which is especially useful if the broadcast/multicast repair session is a new (or recently end time modified) FLUTE download session and other means of service announcement prior to this were not feasible.

The delivery method's associatedDeliveryProcedureDescription may be updated and the new version transmitted in-band with the download session so that currently active client back-off timers are reset, thus minimizing additional client requests until after the broadcast/multicast repair session. The server shall be prepared for additional requests in any case as successful reception of the updated associatedDeliveryProcedureDescription can not be assured in all cases.

The existence of a broadcast/multicast file repair session is signalled by the inclusion of the optional *bmFileRepair* procedure in the updated Associated Delivery procedure description. This is signalled by the *bmFileRepair* element with a single "sessionDescriptionURI" attribute of type "xs:anyURI" which specifies the URI of the broadcast/multicast file repair session's session description.

In the cases where the same IP addressing is used for the broadcast/multicast repair session as the original download session, the UE simply shall not leave the group. Otherwise, the UE shall join to the MBMS bearer for the repair session as it would for any MBMS session.

A broadcast/multicast file repair session behaves just as an MBMS download session, and the determination of end of files and session, and use of further associated delivery procedures uses the same techniques as specified for the MBMS download delivery method.

## 9.3.8 Server Not Responding Error Case

In the error case where a UE determines that the its selected file repair server is not responding it shall return to the serverURI list of repair servers and uniformly randomly select another server from the list, excluding any servers it has determined are not responding. All the repair requests message(s) from that UE shall then be immediately sent to the newly selected file repair server.

If all of the repair servers from the serverURI list are determined to be not responding, the UE may attempt an HTTP GET to retrieve a, potentially new, instance of the session's Associated Procedure Description; otherwise UE behaviour in this case is unspecified.

A UE determines that a file repair server is not responding if any of these conditions apply:

1. The UE is unable to establish a TCP connection to the server.

2. The server does not respond to any of the HTTP repair requests that have been sent by the UE (it is possible that second and subsequent repair requests are sent before the first repair request is determined to be not-responded-to).

3. The server returns an unrecognized message (not a recognizable HTTP response).

The server returns an HTTP server error status code (in the range 500 to 505).

## 9.4 The Reception Reporting Procedure

Following successful reception of content whether through point-to-multipoint MBMS bearers only or using both point-to-multipoint and point-to-point bearers, a reception reporting procedure can be initiated by the MBMS Receiver (UE) to the BM-SC.

For MBMS Download Delivery method, the reception reporting procedure is used to report the complete reception of one or more files. For MBMS Streaming Delivery method, the reception reporting procedure is used to report statistics on the stream.

If the BM-SC provided parameters requiring reception reporting confirmation then the MBMS Receiver shall confirm the content reception.

If reception reporting is requested for statistical purposes the BM-SC may specify the percentage subset of MBMS receivers it would like to perform reception reporting.

Transport errors can prevent an MBMS Receiver from deterministically discovering whether the reception reporting associated delivery procedure is described for a session, and even if this is successful whether a sample percentage is described. An MBMS Receiver shall behave according to the information it has even when it is aware that this may be incomplete.

The MBMS Receiver:

1. Identifies the complete reception of a content item (e.g. a file). See clauses 9.4.1and 9.4.2.

2. Determines the need to report reception. See clause 9.4.3.

3. Selects a time (Request time) at which a reception report request will be sent and selects a server from a list - both randomly and uniformly distributed. See clause 9.4.4 and 9.4.5.

4. Sends a *reception report request* message to the selected server at the selected time. See clause 9.4.6.

Then the server:

1. Responds with a *reception report response* message either containing the requested data, or alternatively, describing an error case. See clause 9.4.7.

### 9.4.1 Identifying Complete File Reception from MBMS Download

A file is determined to be completely downloaded when it is fully received and reconstructed by MBMS reception with FEC decoding (if FEC is actually used) and/or a subsequent File Repair Procedure (clause 9.3). The purpose of determining file download completeness is to determine when it is feasible for a UE to compile the RAck reception report for that file.

### 9.4.2 Identifying Complete MBMS Delivery Session Reception

Delivery sessions (download and streaming) are considered complete when the "time to" value of the session description (from "t=" in SDP) is reached. Where the end time is unbounded (time to = 0) then this parameter is not used for identifying completed sessions.

Delivery sessions are also considered complete when the UE decides to exit the session - where no further data from that session will be received. In this case the UE may or may not deactivate the MBMS bearer(s).

For MBMS download sessions, FLUTE provides a "Close session flag" (see clause 7.2.6) which, when used, indicates to the UE that the session is complete.

## 9.4.3 Determining Whether a Reception Report Is Required

Upon full reception of a content item or when a session is complete, the MBMS Receiver must determine whether a reception report is required. An Associated Delivery Procedure Description indicates the parameters of a reception reporting procedure (which is transported using the same methods as the ones that describe File Repair).

A delivery method may associate zero or one associated delivery procedure descriptions with an MBMS delivery session. Where an associated delivery procedure description is associated with a session, and the description includes a *postReceptionReport* element, the UE shall initiate a reception reporting procedure. Reception reporting behaviour depends on the parameters given in the description as explained below.

The Reception Reporting Procedure is initiated if:

a. A *postReceptionReport* element is present in the associated procedure description instance.

One of the following will determine the UE behaviour:

b. *reportType* is set to RAck (Reception Acknowledgement). Only successful file reception is reported without reception details.

c. *reportType* is set to StaR (Statistical Reporting for successful reception). Successful file reception is reported (as with RAck) with reception details for statistical analysis in the network.

d. *reportType* is set to StaR-all (Statistical Reporting for all content reception). The same as StaR with the addition that failed reception is also reported. StaR-all is relevant to both streaming and download delivery.

The *reportType* attribute is optional and behaviour shall default to RAck when it is not present.

The *samplePercentage* attribute can be used to set a percentage sample of receivers which should report reception. This can be useful for statistical data analysis of large populations while increasing scalability due to reduced total uplink signalling. The *samplePercentage* takes on a value between 0 and 100, including the use of decimals. It is recommended that no more than 3 digits follow a decimal point (e.g. 67.323 is sufficient precision).

The *samplePercentage* attribute is optional and behaviour shall default to 100 (%) when it is not present. The *samplePercentage* attribute may be used with StaR and StaR-all, but shall not be used with RAck.

When the *samplePercentage* is not present or its value is 100 each UE which entered the associated session shall send a reception report. If the *samplePercentage* were provided for reportType StaR and StaR-all and the value is less than 100, the UE generates a random number which is uniformly distributed in the range of 0 to100. The UE sends the reception report when the generated random number is of a lower value than the *samplePercentage* value.

## 9.4.4 Request Time Selection

The MBMS receiver selects a time at which it is to issue a delivery confirmation request.

Back-off timing is used to spread the load of delivery confirmation requests and responses over time.

Back-off timing is performed according to the procedure described in clause 9.3.2.3. The *offsetTime* and *randomTimePeriod* used for delivery confirmation may have different values from those used for file-repair and are signalled separately in the reception reporting description of the associated delivery procedure description instance.

In general, reception reporting procedures may be less time critical than file repair procedures. Thus, if a postFileRepair timer may expire earlier than a postReceptionReport, radio and signalling resources may be saved by using the file repair point-to-point PDP context (and radio bearer) activate period also for reception reporting (to remove the delay and signalling of multiple activations and deactivations over time)

The default behaviour is that a UE shall stop its postReceptionReport timers which are active when a postFileRepair timer expires, which results in the successful initiation of point-to-point communications between UE and BM-SC.

In some circumstances, the system bottleneck may be in the server handling of reception reporting. In this case the *forceTimeIndependence* attribute may be used and set to true. (false is the default case and would be a redundant use of this optional attribute). When *forceTimeIndependence* is true the UE shall not use file repair point-to-point connections to send reception reporting messages. Instead it will allow the timers to expire and initiate point-to-point connections dedicated to reception report messaging.

For StaR and StaR-all, session completeness - according to clause 9.4.2 - shall determine the back-off timer initialization time.

For RAck, the complete download session - according to clause 9.4.2 - as well as completing any associated file repair delivery procedure shall determine the back-off timer initialization time. RAcks shall be only sent for completely received files according to clause 9.4.1.

### 9.4.5 Reception Report Server Selection

Reception report server selection is performed according to the procedure described in clause 9.3.3.2.

### 9.4.6 Reception Report Message

Once the need for reception reporting has been established, the MBMS receiver sends one or more Reception Report messages to the BM-SC. All Reception Report requests and responses for a particular MBMS transmission should take place in a single TCP session using the HTTP protocol (RFC 2616 [18]).

The Reception Report request shall include the URI of the file for which delivery is being confirmed. URI is required to uniquely identify the file (resource).

The client shall make a Reception Report request using the HTTP (RFC 2616 [18]) POST request carrying XML formatted metadata for each reported received content (file). An HTTP session shall be used to confirm the successful delivery of a single file. If more than one file were downloaded in a particular MBMS download multiple descriptions shall be added in a single POST request.

Each Reception Report is formatted in XML according the following XML schema (clause 9.5.2). An informative example of a single reception report XML object is also given (clause 9.5.2.2).

Multipart MIME (multipart/mixed) may be used to aggregate several small XML files of reception reports to a larger object.

For Reception Acknowledgement (RAck) a receptionAcknowledgement element shall provide the relevant data.

For Statistical Reporting (StaR) a statisticalReporting element shall provide the relevant data.

For both RAck and StaR/StaR-all (mandatory):

- For download, one or more *fileURI* elements shall specify the list of files which are reported.

For only StaR/StaR-all (all optional):

- Each *fileURI* element has an optional *receptionSuccess* status code attribute which defaults to "true" ("1") when not used. This attribute shall be used for StaR-all reports. This attribute shall not be used for StaR reports.
- Each QoE Metrics element has eleven attributes as defined in clause 9.5.2 that correspond to the QoE metrics listed in clause 8.4.2. Individual metrics, both at session and at media level can be selected via SDP as described in clause 8.3.1.1.
- The *sessionID* attribute identified the delivery session. This is of the format source_IP_address + ":" + FLUTE_TSI/RTP_source_port.
- The *sessionType* attribute defines the basic delivery method session type used = "download" || "streaming" || "mixed".
- The *serviceId* attribute is value and format is taken from the respective userServiceDescription serviceID definition.
- The *clientId* attribute is unique identifier for the receiver. [format is FFS].

- The *serverURI* attribute value and format is taken from the respective associatedDeliveryProcedureDescription serverURI which was selected by the UE for the current report. This attribute expresses the reception report server to which the reception report is addressed.

#### 9.4.7 Reception Report Response Message

An HTTP response is used as the Reception Report response message.

The HTTP header shall use a status code of 200 OK to signal successful processing of a Reception Report. Other status codes may be used in error cases as defined in RFC 2616 [18].

## 9.5 XML-Schema for Associated Delivery Procedures

### 9.5.1 Generic Associated Delivery Procedure Description

Below is the formal XML syntax of associated delivery procedure description instances.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="associatedProcedureDescription" type="associatedProcedureType"/>
    <xs:complexType name="associatedProcedureType">
        <xs:sequence>
            <xs:element name="postFileRepair" type="basicProcedureType" minOccurs="0"
            maxOccurs="1"/>
            <xs:element name="bmFileRepair" type=" bmFileRepairType" minOccurs="0" maxOccurs="1"/>
            <xs:element name="postReceptionReport" type="reportProcedureType" minOccurs="0"
            maxOccurs="1"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="basicProcedureType">
        <xs:sequence>
            <xs:element name="serverURI" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="offsetTime" type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="maxBackOff" type="xs:unsignedLong" use="required"/>
    </xs:complexType>

    <xs:complexType name="bmFileRepairType">
        <xs:attribute name="sessionDescriptionURI" type="xs:anyURI " use="required"/>
    </xs:complexType>
    <xs:complexType name="reportProcedureType">
        <xs:simpleContent>
            <xs:extension base="basicProcedureType">
                <xs:attribute name="samplePercentage" type="xs:decimal" default="100"
use="optional"/>
                <xs:attribute name="forceTimeIndependence" type="xs:boolean" default="false"
use="optional"/>
                <xs:attribute name="reportType" type="xs:string" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>

"reportType" value = "rack" || "star" || "star-all"
```

#### 9.5.2 Example Associated Delivery Procedure Description Instance

Below is an example of an associated delivery procedure description for reception reporting.

```
<?xml version="1.0" encoding="UTF-8"?>
<associatedProcedureDescription xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.example.com/mbms-associated-descrition.xsd">
    <postFileRepair
            offsetTime="5"
            maxBackOff="10">
        <serverURI>http://mbmsrepair.operator.umts/"</serverURI>
        <serverURI>http://mbmsrepair1.operator.umts/"</serverURI>
        <serverURI>http://mbmsrepair2.operator.umts/"</serverURI>
    </postFileRepair>
    <bmFileRepair sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>
    <postReceptionReport
```

```
            offsetTime="5"
            maxBackOff="10"
            reportType="star-all"
            samplePercentage="100"
            forceTimeIndependence="0">
        <serverURI>"http://mbmsrepair.operator.umts/"</serverURI>
    </postReceptionReport>
</associatedProcedureDescription>
```

### 9.5.3 XML Syntax for a Reception Report Request

Below is the formal XML syntax of reception report request instances.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="receptionReport">
        <xs:choice>
        <xs:element name="receptionAcknowledgement" type="rackType"/>
        <xs:element name="statisticalReport" type="starType"/>
        </xs:choice>
    </xs:element>
<xs:complexType name="rackType">
    <xs:sequence>
        <xs:element name="fileURI" type="xs:anyURI"
            minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="starType">
    <xs:simpleContent>
        <xs:element name="fileURI" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded">
            <xs:attribute name="receptionSuccess" type="xs:boolean" use="optional"/>
        </xs:element>
        <xs:element name="qoeMetrics" type="qoeMetricsType" minOccurs="0"/>
        <xs:attribute name="sessionId" type="xs:string" use="optional"/>
        <xs:attribute name="sessionType" type="xs:string" use="optional"/>
        <xs:attribute name="serviceId" type="xs:string" use="optional"/>
        <xs:attribute name="clientId" type="xs:string" use="optional"/>
        <xs:attribute name="serverURI" type="xs:anyURI" use="optional"/>
    </xs:simpleContent>
</xs:complexType>
    <xs:complexType name="qoeMetricsType">
        <xs:simpleContent>
            <xs:attribute name="totalCorruptionDuration" type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="numberOfCorruptionEvents" type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="t" type="xs:boolean" use="optional"/>
            <xs:attribute name="totalRebufferingDuration" type="xs:double" use="optional"/>
            <xs:attribute name="numberOfRebufferingEvents" type="xs:unsignedLong" use="optional"/>
            <xs:attribute name="initialBufferingDuration" type="xs:double" use="optional"/>
            <xs:attribute name="totalNumberofSuccessivePacketLoss" type="xs:unsignedLong"
            use="optional"/>
            <xs:attribute name="numberOfSuccessiveLossEvents" type="xs:unsignedLong"
            use="optional"/>
            <xs:attribute name="framerateDeviation" type="xs:double" use="optional"/>
            <xs:attribute name="totalJitterDuration" type="xs:double" use="optional"/>
            <xs:attribute name="numberOfJitterEvents" type="xs:unsignedLong" use="optional"/>
        </xs:simpleContent>
        </xs:complexType>
    </xs:complexType>
</xs:schema>
```

#### 9.5.3.1 Use of Specific Values

"sessionType" value = {"download", "streaming", "mixed"}

#### 9.5.3.2 Example XML for the Reception Report Request

```
<?xml version="1.0" encoding="UTF-8"?>
<receptionReport
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.example.com/mbmsReceptionReport.xsd">
    <receptionAcknowledgement>
        <fileURI>"http://www.example.com/mbms-files/file1.3gp"</fileURI>
        <fileURI>"http://www.example.com/mbms-files/file2.3gp"</fileURI>
        <fileURI>"http://www.example.com/mbms-files/file4.3gp"</fileURI>
    </receptionAcknowledgement>
</receptionReport>
```

# 10 Media codecs and formats

## 10.1 General

The set of media decoders that are supported by the MBMS Client to support a particular media type are defined below. Speech, Audio, Video and Timed Text media decoders are relevant for both MBMS Download and Streaming delivery. Other media decoders are only relevant for MBMS Download delivery.

## 10.2 Speech

If speech is supported, the AMR decoder shall be supported for narrow-band speech 3GPP TS 26.071 [48], 3GPP TS 26.090 [49], 3GPP TS 26.073 [50] and 3GPP TS 26.107 [51]. The AMR wideband speech decoder, 3GPP TS 26.171 [52], 3GPP TS 26.190 [53], 3GPP TS 26.173 [54] and 3GPP TS 26.204 [55], shall be supported when wideband speech working at 16 kHz sampling frequency is supported.

## 10.3 Audio

If audio is supported, then the following two audio decoders should be supported:

- Enhanced aacPlus 3GPP TS 26.401 [28], 3GPP TS 26.410 [29] and 3GPP TS 26.411 [30].
- Extended AMR-WB 3GPP TS 26.290 [24], 3GPP TS 26.304 [25] and 3GPP TS 26.273 [26].

Specifically, based on the audio codec selection test results, Extended AMR-WB is strong for the scenarios marked with blue, Enhanced aacPlus is strong for the scenarios marked with orange, and both are strong for the scenarios marked with green colour in table 1.

**Table 1**

| Content type<br>Bit rate | Music | Speech over Music | Speech between Music | Speech |
|---|---|---|---|---|
| 14 kbps mono | 🟩 | 🟩 | 🟦 | 🟦 |
| 18 kbps stereo | 🟧 | 🟧 | 🟦 | 🟦 |
| 24 kbps stereo | 🟧 | 🟧 | 🟦 | 🟦 |
| 24 kbps mono | 🟧 | 🟧 | 🟧 | 🟦 |
| 32 kbps stereo | 🟧 | 🟧 | 🟧 | 🟧 |
| 48 kbps stereo | 🟧 | 🟧 | 🟧 | 🟧 |

## 10.4 Synthetic audio

If synthetic audio is supported, the Scalable Polyphony MIDI (SP-MIDI) content format defined in Scalable Polyphony MIDI Specification [56] and the device requirements defined in Scalable Polyphony MIDI Device 5-to-24 Note Profile for 3GPP [57] should be supported.

SP-MIDI content is delivered in the structure specified in Standard MIDI Files 1.0 [58], either in format 0 or format 1.

In addition the Mobile DLS instrument format defined in [59] and the Mobile XMF content format defined in [60] should be supported.

A PSS client supporting Mobile DLS shall meet the minimum device requirements defined in [59] in section 1.3 and the requirements for the common part of the synthesizer voice as defined in ISO/IEC 10646-1 [70] in section 1.2.1.2. If Mobile DLS is supported, wavetables encoded with the G.711 A-law codec (wFormatTag value 0x0006, as defined in [59]) shall also be supported. The optional group of processing blocks as defined in [59] may be supported. Mobile DLS resources are delivered either in the file format defined in ISO/IEC 10646-1 [70], or within Mobile XMF as defined in

[60]. For Mobile DLS files delivered outside of Mobile XMF, the loading application should unload Mobile DLS instruments so that the sound bank required by the SP-MIDI profile [57] is not persistently altered by temporary loadings of Mobile DLS files.

Content that pairs Mobile DLS and SP-MIDI resources is delivered in the structure specified in Mobile XMF [60]. As defined in [60], a Mobile XMF file shall contain one SP-MIDI SMF file and no more than one Mobile DLS file. PSS clients supporting Mobile XMF must not support any other resource types in the Mobile XMF file. Media handling behaviours for the SP-MIDI SMF and Mobile DLS resources contained within Mobile XMF are defined in [60].

## 10.5 Video

If video is supported, H.264 (AVC) Baseline Profile Level 1b decoder (ITU-T Recommendation H.264 [43] and ISO/IEC 14496-10/FDAM1: "AVC Fidelity Range Extensions" [44]) with constraint_set1_flag=1 and without requirements on output timing conformance (annex C of ITU-T Recommendation H.264 [43]) should be supported.

Note that MBMS does not offer dynamic negotiation of media codecs. To ensure the maximum level of interoperability, H.264 (AVC) is the only video decoder recommended for MBMS. However, it is to be noted that ITU-T Recommendation H.263 profile 0 level 45 decoder (ITU-T Recommendation H.263 [45] and H.263 annex X [46]) shall be supported for PSS (3GPP TS 26.234 [47]) and hence may be used for MBMS User Service.

When H.264 (AVC) is in use in the MBMS streaming delivery method, it is recommended to transmit H.264 (AVC) parameter sets within the SDP description of a stream (using sprop-parameter-sets MIME/SDP parameter - ISO/IEC 14496-10/FDAM1: "AVC Fidelity Range Extensions" [44]), and it is not recommended to transmit parameter sets within the RTP stream. Moreover, it is not recommended to reuse any parameter set identifier value that appeared previously in the SDP description or in the RTP stream. However, if a sequence parameter set is taken into use or updated within the RTP stream, it shall be contained at least in each IDR access unit and each access unit including a recovery point SEI message in which the sequence parameter set is used in the decoding process. If a picture parameter set is taken into use or updated within the RTP stream, it shall be contained at the latest in the first such access unit in each entry sequence that uses the picture parameter set in the decoding process, in which an entry sequence is defined as the access units between an IDR access unit or an access unit containing a recovery point SEI message, inclusive, and the next access unit, exclusive, in decoding order, which is either an IDR access unit or contains a recovery point SEI message.

There are no requirements on output timing conformance (annex C of ITU-T Recommendation H.264 [43]) for MBMS clients.

The H.264 (AVC) decoder in an MBMS client shall start decoding immediately when it receives data (even if the stream does not start with an IDR access unit) or alternatively no later than it receives the next IDR access unit or the next recovery point SEI message, whichever is earlier in decoding order. Note that when the interleaved packetization mode of H.264 (AVC) is in use, de-interleaving is done normally before starting the decoding process. The decoding process for a stream not starting with an IDR access unit shall be the same as for a valid H.264 (AVC) bitstream. However, the client shall be aware that such a stream may contain references to pictures not available in the decoded picture buffer.

## 10.6 Still images

If still images are supported, ISO/IEC JPEG [61] together with JFIF [62] decoders shall be supported. The support for ISO/IEC JPEG only applies to the following two modes:

- baseline DCT, non-differential, Huffman coding, as defined in table B.1, symbol 'SOF0' in 3GPP TS 26.273 [26];
- progressive DCT, non-differential, Huffman coding, as defined in table B.1, symbol 'SOF2' 3GPP TS 26.273 [26].

## 10.7 Bitmap graphics

If bitmap graphics is supported, the following bitmap graphics decoders should be supported:

- GIF87a, [63];

- GIF89a, [64];
- PNG, [65].

## 10.8 Vector graphics

If vector graphics is supported, SVG Tiny 1.2 [66], [67] and ECMAScript [68] shall be supported.

NOTE 1: The compression format for SVG content is GZIP [42], in accordance with the SVG specification [66].

NOTE 2 Content creators of SVG Tiny 1.2 are strongly recommended to follow the content creation guidelines provided in annex L of 3GPP TS 26.234 [47].

NOTE 3: If SVG Tiny 1.2 will not be published within a reasonable timeframe, the decision to adopt SVG Tiny 1.2 in favour of SVG Tiny 1.1 may be reconsidered.

## 10.9 Text

The text decoder is intended to enable formatted text in a SMIL presentation.

If text is supported, a MBMS client shall support

- text formatted according to XHTML Mobile Profile [69];
- rendering a SMIL presentation where text is referenced with the SMIL 2.0 "text" element together with the SMIL 2.0 "src" attribute.

If text is supported, the following character coding formats shall be supported:

- UTF-8, [71];
- UCS-2, [70].

NOTE: Since both SMIL and XHTML are XML based languages it would be possible to define a SMIL plus XHTML profile. In contrast to the presently defined SMIL Language Profile that only contain SMIL modules, such a profile would also contain XHTML modules. No combined SMIL and XHTML profile is specified for MBMS. Rendering of such documents is out of the scope of the present document.

## 10.10 Timed text

If timed text is supported, MBMS clients shall support 3GPP TS 26.245 [72]. Timed text may be transported over RTP or downloaded contained in 3GP files using Basic profile.

NOTE: When a MBMS client supports timed text it needs to be able to receive and parse 3GP files containing the text streams. This does not imply a requirement on MBMS clients to be able to render other continuous media types contained in 3GP files, e.g. AMR, if such media types are included in a presentation together with timed text. Audio and video are instead streamed to the client using RTP.

## 10.11 3GPP file format

An MBMS client shall support the Basic profile and the Extended presentation profile of the 3GPP file format 3GPP TS 26.244 [32].

# Annex A (normative): FLUTE Support Requirements

This clause provides a table representation of the requirement levels for different features in FLUTE. Table A.1 includes requirements for an MBMS client and an MBMS server for FLUTE support as well as the requirements for a FLUTE client and a FLUTE server according to the FLUTE protocol (RFC 3926 [9]). The terms used in table A.1 are described underneath.

**Table A.1: Overview of the FLUTE support requirements in MBMS servers and clients**

| | **FLUTE Client support requirement as per [9]** | **MBMS FLUTE Client support requirement as per present document** | **FLUTE Server use requirement as per [9]** | **MBMS FLUTE Server use requirement as per present document** |
|---|---|---|---|---|
| FLUTE Blocking Algorithm | Required | Required | Strongly recommended | Required |
| Symbol Encoding Algorithm | Compact No-Code algorithm required.<br><br>Other FEC building blocks are undefined optional plug-ins. | Compact No-Code algorithm required.<br><br>MBMS Forward Error Correction required | Compact No-Code algorithm is the default option.<br><br>Other FEC building blocks are undefined optional plug-ins. | Compact No-Code algorithm is the default option.<br><br>MBMS Forward Error Correction. |
| Congestion Control Building Block (CCBB) / Algorithm | Congestion Control building blocks undefined. | Single channel support required | Single channel without additional CCBB given for the controlled network scenario. | Single channel support required |
| Content Encoding for FDT Instances | Optional | Required | Optional | Optional |
| A flag active (header) | Required | Required | Optional | Not recommended to use |
| B flag active (header) | Required | Required | Optional | Not recommended to use |
| T flag active and SCT field (header) | Optional | Required | Optional | Required for FDT packets |
| R flag active and ERT field (header) | Optional | Optional | Optional | Optional |
| Content-Location attribute (FDT) | Required | Required | Required | Required |
| TOI (FDT) | Required | Required | Required | Required |
| FDT Expires attribute (FDT) | Required | Required | Required | Required |
| Complete attribute (FDT) | Required | Required | Optional | Optional |
| FEC-OTI-Maximum-Source-Block-Length | Required | Required | Required | Required |
| FEC-OTI-Encoding-Symbol-Length | Required | Required | Required | Required |
| FEC-OTI-Max-Number-of-Encoding-Symbols. | Required | Required | Required | Required |
| FEC-OTI-FEC-Instance-ID | Required | [TBD] | Required | [TBD] |
| FEC-OTI-Scheme-Specific-Info | n/a | Required | n/a | Required if MBMS FEC used |

The following are descriptions of the above terms:

- **Blocking algorithm:** The blocking algorithms is used for the fragmentation of files. It calculates the source blocks from the source files.

- **Symbol Encoding algorithm:** The symbol encoding algorithm is used for the fragmentation of files. It calculates encoding symbols from source blocks for Compact No-Code FEC. It may also be used for other FEC schemes.
- **Congestion Control Building Block:** A building block used to limit congestion by using congestion feedback, rate regulation and receiver controls (RFC 3048 [17]).
- **Content Encoding for FDT Instances:** FDT Instance may be content encoded for more efficient transport, e.g. using ZLIB.
- **A flag:** The Close Session flag for indicating the end of a session to the receiver in the ALC/LCT header.
- **B flag:** The Close Object flag is for indicating the end of an object to the receiver in the ALC/LCT header.
- **T flag:** The T flag is used to indicate the use of the optional "Sender Current Time (SCT)" field (when T=1) in the ALC/LCT header.
- **R flag:** The R flag is used to indicate the use of the optional "Expected Residual Time (ERT) field in the ALC/LCT header.
- **Content Location attribute:** This attribute provides a URI for the location where a certain piece of content (or file) being transmitted in a FLUTE session is located.
- **Transport Object Identifier (TOI):** The TOI uniquely identifies the object within the session from which the data in the packet was generated.
- **FDT Expires attribute:** Indicates to the receiver the time until which the information in the FDT is valid.
- **Complete attribute:** This may be used to signal that the given FDT Instance is the last FDT Instance to be expected on this file delivery session.
- **FEC-OTI-Maximum-Source-Block-Length:** This parameter indicates the maximum number of source symbols per source block.
- **FEC-OTI-Encoding-Symbol-Length:** This parameter indicates the length of the Encoding Symbol in bytes.
- **FEC-OTI-Max-Number-of-Encoding-Symbols:** This parameter indicates the maximum number of Encoding Symbols that can be generated for a source block.
- **FEC-OTI-FEC-Instance-ID:** This field is used to indicate the FEC Instance ID, if a FEC scheme is used.
- FEC-OTI-Scheme-Specific-Info: Carries Object Transmission Information which is specific to the FEC scheme in use.

# Annex B (normative): FEC encoder and decoder specification

This Annex specifies the systematic Raptor forward error correction code and its application to MBMS [7]. Raptor is a fountain code, i.e., as many encoding symbols as needed can be generated by the encoder on-the-fly from the source symbols of a block. The decoder is able to recover the source block from any set of encoding symbols only slightly more in number than the number of source symbols.

The code described in this document is a Systematic code, that is, the original source symbols are sent unmodified from sender to receiver, as well as a number of repair symbols.

## B.1 Definitions, Symbols and Abbreviations

### B.1.1 Definitions

For the purposes of this Annex, the following terms and definitions apply.

**Source block**: a block of *K* source symbols which are considered together for Raptor encoding purposes.

**Source symbol**: the smallest unit of data used during the encoding process. All source symbols within a source block have the same size.

**Encoding symbol**: a symbol that is included in a data packet. The encoding symbols consist of the source symbols and the repair symbols. Repair symbols generated from a source block have the same size as the source symbols of that source block.

**Systematic code:** a code in which the source symbols are included as part of the encoding symbols sent for a source block.

**Repair symbol**: the encoding symbols sent for a source block that are not the source symbols. The repair symbols are generated based on the source symbols.

**Intermediate symbols:** symbols generated from the source symbols using an inverse encoding process . The repair symbols are then generated directly from the intermediate symbols. The encoding symbols do not include the intermediate symbols, i.e., intermediate symbols are not included in data packets.

**Symbol**: a unit of data. The size, in bytes, of a symbol is known as the symbol size.

**Encoding symbol group**: a group of encoding symbols that are sent together, i.e., within the same packet whose relationship to the source symbols can be derived from a single Encoding Symbol ID.

**Encoding Symbol ID**: information that defines the relationship between the symbols of an encoding symbol group and the source symbols.

**Encoding packet:** data packets that contain encoding symbols

**Sub-block**: a source block is sometime broken into sub-blocks, each of which is sufficiently small to be decoded in working memory. For a source block consisting of *K* source symbols, each sub-block consists of *K* sub-symbols, each symbol of the source block being composed of one sub-symbol from each sub-block.

**Sub-symbol**: part of a symbol. Each source symbol is composed of as many sub-symbols as there are sub-blocks in the source block.

**Source packet:** data packets that contain source symbols.

**Repair packet:** data packets that contain repair symbols.

## B.1.2 Symbols

*i, j, x, h, a, b, d, v, m* represent positive integers

| | |
|---|---|
| ceil(*x*) | denotes the smallest positive integer which is greater than or equal to *x* |
| choose(*i*,*j*) | denotes the number of ways *j* objects can be chosen from among *i* objects without repetition |
| floor(*x*) | denotes the largest positive integer which is less than or equal to *x* |
| *i* % *j* | denotes *i* modulo *j* |
| *X* ^ *Y* | denotes, for equal-length bit strings *X* and *Y,* the bitwise exclusive-or of *X* and *Y* |
| *A* | denote a symbol alignment parameter. Symbol and sub-symbol sizes are restricted to be multiples of *A*. |
| $\mathbf{A}^T$ | denotes the transposed matrix of matrix **A** |
| $\mathbf{A}^{-1}$ | denotes the inverse matrix of matrix **A** |
| *K* | denotes the number of symbols in a single source block |
| $K_{MAX}$ | denotes the maximum number of source symbols that can be in a single source block. Set to 8192. |
| *L* | denotes the number of pre-coding symbols for a single source block |
| *S* | denotes the number of LDPC symbols for a single source block |
| *H* | denotes the number of Half symbols for a single source block |
| **C** | denotes an array of intermediate symbols, *C*[0], *C*[1], *C*[2],…, *C*[*L*-1] |
| **C**" | denotes an array of source symbols, *C"*[0], *C"*[1], *C"*[2],…, *C*"[*K*-1] |
| *X* | a non-negative integer value |
| $V_0$, $V_1$ | two arrays of 4-byte integers, $V_0[0]$, $V_0[1]$,…, $V_0[255]$ and $V_1[0]$, $V_1[1]$,…, $V_1[255]$ |
| Rand[*X*, *i*, *m*] | a pseudo-random number generator |
| Deg[*v*] | a degree generator |
| LTEnc[*K*, **C** ,(*d*, *a*, *b*)] | a LT encoding symbol generator |
| Trip[*K*, *X*] | a triple generator function |
| *G* | the number of symbols within an encoding symbol group |
| *N* | the number of sub-blocks within a source block |
| *T* | the symbol size in bytes. If the source block is partitioned into sub-blocks, then $T = T''\cdot N$. |
| *T"* | the sub-symbol size, in bytes. If the source block is not partitioned into sub-blocks then *T"* is not relevant. |
| *F* | the file size, for file download, in bytes |
| *I* | the sub-block size in bytes |
| *P* | for file download, the payload size of each packet, in bytes, that is used in the recommended derivation of the file download transport parameters. For streaming, the payload size of each repair packet, in bytes, that is used in the recommended derivation of the streaming transport parameters. |
| *Q* | $Q = 65521$, i.e., *Q* is the largest prime smaller than $2^{16}$ |
| *Z* | the number of source blocks, for file download |
| *J(K)* | the systematic index associated with *K* |

| | |
|---|---|
| $\mathbf{G}$ | denotes any generator matrix |
| $\mathbf{I}_S$ | denotes the $S$x$S$ identity matrix |
| $\mathbf{0}_{SxH}$ | denotes the $S$x$H$ zero matrix |

### B.1.3 Abbreviations

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

| | |
|---|---|
| ESI | Encoding Symbol ID |
| LDPC | Low Density Parity Check |
| LT | Luby Transform |
| SBN | Source Block Number |
| SBL | Source Block Length (in units of symbols) |

## B.2 Overview

The Raptor forward error correction code can be applied to both the MBMS file delivery and MBMS streaming applications described in the main body of this document. Raptor code aspects which are specific to each of these applications are discussed in Sections B.3 and B.4 of this document.

The principle component of the systematic Raptor code is the basic encoder described in Section B.5. First, it is described how to derive values for a set of intermediate symbols from the original source symbols such that knowledge of the intermediate symbols is sufficient to reconstruct the source symbols. Secondly, the encoder produces repair symbols which are each the exclusive OR of a number of the intermediate symbols. The encoding symbols are the combination of the source and repair symbols. The repair symbols are produced in such a way that the intermediate symbols and therefore also the source symbols can be recovered from any sufficiently large set of encoding symbols.

This document defines the systematic Raptor code encoder. A number of possible decoding algorithms are possible. An efficient decoding algorithm is provided in Section B.8.

The construction of the intermediate and repair symbols is based in part on a pseudo-random number generator described in Section B.5. This generator is based on a fixed set of 512 random numbers which must be available to both sender and receiver. These are provided in Section B.7.

Finally, the construction of the intermediate symbols from the source symbols is governed by a "systematic index", values of which are provided in Section B.6 for source block sizes from 4 source symbols to $K_{MAX}$ = 8192 source symbols.

## B.3 File download

### B.3.1 Source block construction

#### B.3.1.1 General

In order to apply the Raptor encoder to a source file, the file may be broken into $Z \geq 1$ blocks, known as *source blocks*. The Raptor encoder is applied independently to each source block. Each source block is identified by a unique integer Source Block Number (SBN), where the first source block has SBN zero, the second has SBN one, etc. Each source block is divided into a number, $K$, of *source symbols* of size $T$ bytes each. Each source symbol is identified by a unique integer Encoding Symbol Identifier (ESI), where the first source symbol of a source block has ESI zero, the second has ESI one, etc.

Each source block with $K$ source symbols is divided into $N \geq 1$ sub-blocks, which are small enough to be decoded in the working memory. Each sub-block is divided into $K$ sub-symbols of size $T''$.

Note that the value of $K$ is not necessarily the same for each source block of a file and the value of $T''$ may not necessarily be the same for each sub-block of a source block. However, the symbol size $T$ is the same for all source blocks of a file and the number of symbols, $K$ is the same for every sub-block of a source block. Exact partitioning of the file into source blocks and sub-blocks is described in B.3.1.2 below.

Figure B.3.1.1.-1 shows an example source block placed into a two dimensional array, where each entry is a $T''$-byte sub-symbol, each row is a sub-block and each column is a source symbol. In this example, the value of $T''$ is the same for every sub-block. The number shown in each sub-symbol entry indicates their original order within the source block. For example, the sub-symbol numbered $K$ contains bytes $T''\cdot K$ through $T''\cdot(K+1)-1$ of the source block. Then, source symbol $i$ is the concatenation of the $i$th sub-symbol from each of the sub-blocks, which corresponds to the sub-symbols of the source block numbered $i$, $K+i$, $2\cdot K+i$,…, $(N-1)\cdot K+i$.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | … | … | … | … | $K$-1 |
| $K$ | $K$+1 | $K$+2 | … | … | … | … | 2·$K$-1 |
| 2·$K$ | 2·$K$+1 | 2·$K$+2 | … | … | … | … | 3·$K$-1 |
| … | … | … | … | … | … | … | |
| ($N$-1)·$K$ | … | … | … | … | … | … | $N$·$K$-1 |

**Figure B.3.1.1-1 – Source symbols from sub-symbols– the 3 highlighted columns show source symbols 0, 2 and *K*-1**

## B.3.1.2 Source block and sub-block partitioning

The construction of source blocks and sub-blocks is determined based on five input parameters, $F$, $A$, $T$, $Z$ and $N$ and a function Partition[]. The five input parameters are defined as follows:

- $F$ the size of the file, in bytes
- $A$ a symbol alignment parameter, in bytes
- $T$ the symbol size, in bytes, which must be a multiple of $A$
- $Z$ the number of source blocks
- $N$ the number of sub-blocks in each source block

These parameters shall be set so that ceil(ceil($F/T$)/$Z$) $\leq K_{MAX}$. Recommendations for derivation of these parameters are provided in Section B.3.4.

The function Partition[] takes a pair of integers ($I$, $J$) as input and derives four integers ($I_L$, $I_S$, $J_L$, $J_S$) as output. Specifically, the value of Partition[$I$, $J$] is a sequence of four integers ($I_L$, $I_S$, $J_L$, $J_S$), where $I_L$ = ceil($I/J$), $I_S$ = floor($I/J$), $J_L = I - I_S \cdot J$ and $J_S = J - J_L$. Partition[] derives parameters for partitioning a block of size $I$ into $J$ approximately equal sized blocks. Specifically, $J_L$ blocks of length $I_L$ and $J_S$ blocks of length $I_S$.

The source file shall be partitioned into source blocks and sub-blocks as follows:

Let,

$K_t$ = ceil($F/T$)

($K_L$, $K_S$, $Z_L$, $Z_S$) = Partition[$K_t$, $Z$ ]

($T_L$, $T_S$, $N_L$, $N_S$) = Partition[$T/A$, $N$ ]

Then, the file shall be partitioned into $Z = Z_L + Z_S$ contiguous source blocks, the first $Z_L$ source blocks each having length $K_L\cdot T$ bytes and the remaining $Z_S$ source blocks each having $K_S\cdot T$ bytes.

If $K_t \cdot T > F$ then for encoding purposes, the last symbol shall be padded at the end with $K_t \cdot T - F$ zero bytes.

Next, each source block shall be divided into $N = N_L + N_S$ contiguous sub-blocks, the first $N_L$ sub-blocks each consisting of $K$ contiguous sub-symbols of size of $T_L \cdot A$ and the remaining $N_S$ sub-blocks each consisting of $K$ contiguous sub-symbols of size of $T_S \cdot A$. The symbol alignment parameter $A$ ensures that sub-symbols are always a multiple of $A$ bytes.

Finally, the *m*th symbol of a source block consists of the concatenation of the *m*th sub-symbol from each of the $N$ sub-blocks.

## B.3.2 Encoding packet construction

### B.3.2.1 General

Each encoding packet contains the following information:

- Source Block Number (SBN)
- Encoding Symbol ID (ESI)
- encoding symbol(s)

Each source block is encoded independently of the others. Source blocks are numbered consecutively from zero.

Encoding Symbol ID values from 0 to $K$-1 identify the source symbols. Encoding Symbol IDs from $K$ onwards identify repair symbols.

### B.3.2.2 Encoding packet construction

Each encoding packet either consists entirely of source symbols (source packet) or entirely of repair symbols (repair packet). A packet may contain any number of symbols from the same source block. In the case that the last symbol in the packet includes padding bytes added for FEC encoding purposes then these bytes need not be included in the packet. Otherwise, only whole symbols shall be included.

The Encoding Symbol ID, $X$, carried in each source packet is the Encoding Symbol ID of the first source symbol carried in that packet. The subsequent source symbols in the packet have Encoding Symbol IDs, $X$+1 to $X$+$G$-1, in sequential order, where $G$ is the number of symbols in the packet.

Similarly, the Encoding Symbol ID, $X$, placed into a repair packet is the Encoding Symbol ID of the first repair symbol in the repair packet and the subsequent repair symbols in the packet have Encoding Symbol IDs $X$+1 to $X$+$G$-1 in sequential order, where $G$ is the number of symbols in the packet.

Note that it is not necessary for the receiver to know the total number of repair packets. The $G$ repair symbol triples ($d$[0], $a$[0], $b$[0]),…, ($d$[$G$-1], $a$[$G$-1], $b$[$G$-1]) for the repair symbols placed into a repair packet with ESI $X$ are computed using the Triple generator defined in B.5.3.4 as follows:

```
For each $i = 0, …, G-1$
    $(d[i], a[i], b[i]) = \text{Trip}[K,X+i]$
```

The $G$ repair symbols to be placed in repair packet with ESI $X$ are calculated based on the repair symbol triples as described in Section B.5.3 using the intermediate symbols **C** and the LT encoder LTenc[$K$, **C**, ($d$[$i$], $a$[$i$], $b$[$i$])].

## B.3.3 Transport

This section describes the information exchange between the Raptor encoder/decoder and any transport protocol making use of Raptor forward error correction for file delivery.

The Raptor encoder and decoder for file delivery require the following information from the transport protocol:

- The file size, $F$, in bytes

- The symbol alignment parameter, $A$
- The symbol size, $T$, in bytes, which must be a multiple of $A$
- The number of source blocks, $Z$
- The number of sub-blocks in each source block, $N$

The Raptor encoder for file delivery additionally requires:

- the file to be encoded, $F$ bytes

The Raptor encoder supplies the transport protocol with encoding packet information consisting, for each packet, of:

- Source Block Number (SBN)
- Encoding Symbol ID (ESI)
- encoding symbol(s)

The transport protocol shall communicate this information transparently to the Raptor decoder.

Suitable transport protocols based on FLUTE/ALC and HTTP are defined in this specification.

## B.3.4 Example parameters

### B.3.4.1 Parameter derivation algorithm

This section provides recommendations for the derivation of the four transport parameters, $A$, $T$, $Z$ and $N$. This recommendation is based on the following input parameters:

- $F$ the file size, in bytes
- $W$ a target on the sub-block size, in bytes
- $P$ the maximum packet payload size, in bytes, which is assumed to be a multiple of $A$
- $A$ the symbol alignment factor, in bytes
- $K_{MAX}$ the maximum number of source symbols per source block.
- $K_{MIN}$ a minimum target on the number of symbols per source block
- $G_{MAX}$ a maximum target number of symbols per packet

Based on the above inputs, the transport parameters $T$, $Z$ and $N$ are calculated as follows:

Let,

$G = \min\{\mathrm{ceil}(P \cdot K_{MIN}/F), P/A, G_{MAX}\}$ - the approximate number of symbols per packet

$T = \mathrm{floor}(P/(A \cdot G)) \cdot A$

$K_t = \mathrm{ceil}(F/T)$ - the total number of symbols in the file

$Z = \mathrm{ceil}(K_t / K_{MAX})$

$N = \min\{\mathrm{ceil}(\mathrm{ceil}(K_t/Z) \cdot T/W), T/A\}$

The values of $G$ and $N$ derived above should be considered as lower bounds. It may be advantageous to increase these values, for example to the nearest power of two. In particular, the above algorithm does not guarantee that the symbol size, $T$, divides the maximum packet size, $P$, and so it may not be possible to use the packets of size exactly $P$. If, instead, $G$ is chosen to be a value which divides $P/A$, then the symbol size, $T$, will be a divisor of $P$ and packets of size $P$ can be used.

Recommended settings for the input parameters, $W$, $A$, $K_{MIN}$ and $G_{MAX}$ are as follows:

$W$ = 256 KB $A$ = 4 $K_{MIN}$ = 1024 $G_{MAX}$ = 10

### B.3.4.2 Examples

The above algorithm leads to transport parameters as shown in Table B.3.4.2-1 below, assuming the recommended values for $W$, $A$, $K_{MIN}$ and $G_{MAX}$ and $P$ = 512:

**Table B.3.4.2-1**

| File size $F$ | $G$ | Symbol size $T$ | $G*T$ | $K_t$ | Source blocks $Z$ | Sub-blocks $N$ | $K_L$ | $K_S$ | $T_L \cdot A$ | $T_S \cdot A$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 100 KB | 6 | 84 | 504 | 1,220 | 1 | 1 | 1,220 | 1,220 | N/A | N/A |
| 100 KB | 8 | 64 | 512 | 1,600 | 1 | 1 | 1,600 | 1,600 | N/A | N/A |
| 300 KB | 2 | 256 | 512 | 1,200 | 1 | 2 | 1,200 | 1,200 | 128 | 128 |
| 1,000 KB | 1 | 512 | 512 | 2,000 | 1 | 5 | 2,000 | 2,000 | 104 | 100 |
| 3,000 KB | 1 | 512 | 512 | 6,000 | 1 | 12 | 6,000 | 6,000 | 44 | 40 |
| 10,000 KB | 1 | 512 | 512 | 20,000 | 3 | 14 | 6,666 | 6,667 | 40 | 36 |

## B.4 Streaming

### B.4.1 Source block construction

A source block is constructed by the transport protocol, for example as defined in this document, making use of the Systematic Raptor Forward Error Correction code. The symbol size, $T$, to be used for source block construction and the repair symbol construction are provided by the transport protocol. The parameter $T$ shall be set so that the number of source symbols in any source block is at most $K_{MAX}$.

Recommended parameters are presented in section B.4.4.

### B.4.2 Encoding packet construction

As described in B.4.3., each repair packet contains the following information:

- Source Block Number (SBN)
- Encoding Symbol ID (ESI)
- Source Block Length (SBL)
- repair symbol(s)

The number of repair symbols contained within a repair packet is computed from the packet length. The ESI values placed into the repair packets and the repair symbol triples used to generate the repair symbols are computed as described in Section B.3.2.2..

## B.4.3 Transport

This section describes the information exchange between the Raptor encoder/decoder and any transport protocol making use of Raptor forward error correction for streaming.

The Raptor encoder for streaming requires the following information from the transport protocol for each source block:

- The symbol size, $T$, in bytes
- The number of symbols in the source block, $K$
- The Source Block Number (SBN)
- The source symbols to be encoded, $K \cdot T$ bytes

The Raptor encoder supplies the transport protocol with encoding packet information consisting, for each repair packet, of:

- Source Block Number (SBN)
- Encoding Symbol ID (ESI)
- Source Block Length (SBL)
- repair symbol(s)

The transport protocol shall communicate this information transparently to the Raptor decoder.

A suitable transport protocol is defined in this specification.

## B.4.4 Example parameters

### B.4.4.1 Parameter derivation algorithm

This section provides recommendations for the derivation of the transport parameter $T$. This recommendation is based on the following input parameters:

- $B$ the maximum source block size, in bytes
- $P$ the maximum repair packet payload size, in bytes, which is a multiple of $A$
- $A$ the symbol alignment factor, in bytes
- $K_{MAX}$ the maximum number of source symbols per source block.
- $K_{MIN}$ a minimum target on the number of symbols per source block
- $G_{MAX}$ a maximum target number of symbols per repair packet

A requirement on these inputs is that $\mathrm{ceil}(B/P) \leq K_{MAX}$. Based on the above inputs, the transport parameter $T$ is calculated as follows:

Let,

$G = \min\{\mathrm{ceil}(P \cdot K_{MIN}/B), P/A, G_{MAX}\}$ - the approximate number of symbols per packet

$T = \mathrm{floor}(P/(A \cdot G)) \cdot A$

The value of $T$ derived above should be considered as a guide to the actual value of $T$ used. It may be advantageous to ensure that $T$ divides into $P$, or it may be advantageous to set the value of $T$ smaller to minimize wastage when full size repair symbols are used to recover partial source symbols at the end of lost source packets (as long as the maximum number of source symbols in a source block does not exceed $K_{MAX}$). Furthermore, the choice of $T$ may depend on the source packet size distribution, e.g., if all source packets are the same size then it is advantageous to choose $T$ so that the actual payload size of a repair packet $P''$, where $P''$ is a multiple of $T$, is equal to (or as few bytes as possible larger than) the number of bytes each source packet occupies in the source block.

Recommended settings for the input parameters, $A$, $K_{MIN}$ and $G_{MAX}$ are as follows:

$A = 4$ $K_{MIN} = 1024$ $G_{MAX} = 10$

### B.4.4.2 Examples

The above algorithm leads to transport parameters as shown in Table B.4.4.2-1 below, assuming the recommended values for $A$, $K_{MIN}$ and $G_{MAX}$ and $P = 512$:

**Table B.4.4.2-1**

| Max source block size $B$ | $G$ | Symbol size $T$ | $G \cdot T$ |
|---|---|---|---|
| 40 KB | 10 | 48 | 480 |
| 160 KB | 4 | 128 | 512 |
| 640 KB | 1 | 512 | 512 |

## B.5 Systematic Raptor encoder

### B.5.1 Encoding overview

The systematic Raptor encoder is used to generate *repair symbols* from a source block that consists of $K$ *source symbols*.

Symbols are the fundamental data units of the encoding and decoding process. For each source block (sub-block) all symbols (sub-symbols) are the same size. The atomic operation performed on symbols (sub-symbols) for both encoding and decoding is the exclusive-or operation.

Let $C''[0],\ldots, C''[K-1]$ denote the $K$ source symbols.

Let $C[0],\ldots, C[L-1]$ denote $L$ intermediate symbols.

The first step of encoding is to generate a number, $L > K$, of intermediate symbols from the $K$ source symbols. In this step, $K$ source triples $(d[0], a[0], b[0]), \ldots, (d[K-1], a[K-1], b[K-1])$ are generated using the Trip[] generator as described in Section B.5.4.4. The $K$ source triples are associated with the $K$ source symbols and are then used to determine the $L$ intermediate symbols $C[0],\ldots, C[L-1]$ from the source symbols using an inverse encoding process. This process can be can be realized by a Raptor decoding process.

Certain 'pre-coding relationships' must hold within the $L$ intermediate symbols. Section B.5.2 describes these relationships and how the intermediate symbols are generated from the source symbols.

Once the intermediate symbols have been generated, repair symbols are produced and one or more repair symbols are placed as a group into a single data packet. Each repair symbol group is associated with an Encoding Symbol ID (ESI) and a number, $G$, of encoding symbols. The ESI is used to generate a triple of three integers, $(d, a, b)$ <u>for each repair symbol,</u> again using the Trip[] generator as described in Section B.5.4.4. This is done as described in Sections B.3 and B.4 using the generators described in Section B.5.4 . Then, each $(d,a,b)$-triple is used to generate the corresponding repair symbol from the intermediate symbols using the LTEnc[$K$, $C[0],\ldots, C[L-1]$, $(d,a,b)$] generator described in Section B.5.4.3.

## B.5.2 First encoding step: Intermediate symbol Generation

### B.5.2.1 General

The first encoding step is a pre-coding step to generate the $L$ intermediate symbols $C[0], \ldots, C[L-1]$ from the source symbols $C''[0], \ldots, C''[K-1]$. The intermediate symbols are uniquely defined by two sets of constraints:

1. The intermediate symbols are related to the source symbols by a set of *source symbol triples.* The generation of the source symbol triples is defined in Section B.5.2.2 using the the Trip[] generator as described in Section B.5.4.4.

2. A set of pre-coding relationships hold within the intermediate symbols themselves. These are defined in Section B.5.2.3.

The generation of the $L$ intermediate symbols is then defined in Section 5.2.4.

### B.5.2.2 Source symbol triples

Each of the $K$ source symbols is associated with a triple $(d[i], a[i], b[i])$ for $0 \leq i < K$. The source symbol triples are determined using the Triple generator defined in Section B.5.4.4 as:

```
For each $i$, $0 \leq i < K$
    $(d[i], a[i], b[i]) = \mathrm{Trip}[K, i]$
```

### B.5.2.3 Pre-coding relationships

The pre-coding relationships amongst the $L$ intermediate symbols are defined by expressing the last $L$-$K$ intermediate symbols in terms of the first $K$ intermediate symbols.

The last $L$-$K$ intermediate symbols $C[K],\ldots,C[L-1]$ consist of $S$ *LDPC symbols* and $H$ *Half symbols* The values of $S$ and $H$ are determined from $K$ as described below. Then $L= K+S+H$.

Let

$X$ be the smallest positive integer such that $X \cdot (X-1) = 2 \cdot K$.

$S$ be the smallest prime integer such that $S \geq \mathrm{ceil}(0.01 \cdot K) + X$

$H$ be the smallest integer such that $\mathrm{choose}(H,\mathrm{ceil}(H/2)) \geq K + S$

$H'' = \mathrm{ceil}(H/2)L = K+S+H$

$C[0],\ldots, C[K-1]$ denote the first $K$ intermediate symbols

$C[K],\ldots, C[K+S-1]$ denote the $S$ LDPC symbols, initialised to zero

$C[K+S],\ldots, C[L-1]$ denote the $H$ Half symbols, initialised to zero

The $S$ LDPC symbols are defined to be the values of $C[K],\ldots, C[K+S-1]$ at the end of the following process:

```
For $i = 0,\ldots,K-1$ do
    $a = 1 + (\mathrm{floor}(i/S) \% (S-1))$
    $b = i \% S$
    $C[K + b] = C[K + b] \wedge C[i]$
    $b = (b + a) \% S$
    $C[K + b] = C[K + b] \wedge C[i]$
    $b = (b + a) \% S$
```

$C[K + b] = C[K + b] \wedge C[i]$

The *H* Half symbols are defined as follows:

Let

$g[i] = i \wedge (\text{floor}(i/2))$ for all positive integers $i$

Note: $g[i]$ is the Gray sequence, in which each element differs from the previous one in a single bit position

$g[j,k]$ denote the $j$th element, $j$=0, 1, 2, …, of the subsequence of $g[i]$ whose elements have exactly $k$ non-zero bits in their binary representation

Then, the Half symbols are defined as the values of $C[K+S]$,…, $C[L-1]$ after the following process:

```
For $h = 0,\dots,H-1$ do
    For $j = 0,\dots,K+S-1$ do
        If bit $h$ of $g[j,H'']$ is equal to 1 then $C[h+K+S] = C[h+K+S] \wedge C[j]$.
```

## B.5.2.4 Intermediate symbols

### B.5.2.4.1 Definition

Given the $K$ source symbols $C''[0]$, $C''[1]$,…, $C''[K-1]$ the $L$ intermediate symbols $C[0]$, $C[1]$,…, $C[L-1]$ are the uniquely defined symbol values that satisfy the following conditions:

1. The $K$ source symbols $C''[0]$, $C''[1]$,…, $C''[K-1]$ satisfy the $K$ constraints

$$C''[i] \equiv \text{LTEnc}[K, (C[0],\dots, C[L-1]), (d[i], a[i], b[i])], \text{ for all } i, 0 \le i < K.$$

2. The $L$ intermediate symbols $C[0]$, $C[1]$,…, $C[L-1]$ satisfy the pre-coding relationships defined in B.5.2.3.

### B.5.2.4.2 Example method for calculation of intermediate symbols

This subsection describes a possible method for calculation of the $L$ intermediate symbols $C[0]$, $C[1]$,…, $C[L-1]$ satisfying the constraints in B.5.2.4.1

The generator matrix **G** for a code which generates $N$ output symbols from $K$ input symbols is an $N$x$K$ matrix over $GF(2)$, where each row corresponds to one of the output symbols and each column to one of the input symbols and where the $i$th output symbol is equal to the sum of those input symbols whose column contains a non-zero entry in row $i$.

Then, the $L$ intermediate symbols can be calculated as follows:

Let

**C** denote the column vector of the $L$ intermediate symbols, $C[0]$, $C[1]$,…, $C[L-1]$.

**D** denote the column vector consisting of $S+H$ zero symbols followed by the $K$ source symbols $C''[0]$, $C''[1]$, …, $C''[K-1]$

Then the above constraints define an $L$x$L$ matrix over $GF(2)$, **A**, such that:

$$\mathbf{A}\cdot\mathbf{C} = \mathbf{D}$$

The matrix **A** can be constructed as follows:

Let:

$\mathbf{G}_{\text{LDPC}}$ be the $S$ x $K$ generator matrix of the LDPC symbols. So,

$$\mathbf{G}_{\text{LDPC}} \cdot (C[0], \dots, C[K-1])^T = (C[K], \dots, C[K+S-1])^T$$

$\mathbf{G}_{\text{Half}}$ be the $H$ x $(K+S)$ generator matrix of the Half symbols, So,

$\mathbf{G}_{\mathrm{Half}} \cdot (C[0], \ldots, C[S+K-1])^{\mathrm{T}} = (\mathrm{C}[K+S], \ldots, C[K+S+H-1])^{\mathrm{T}}$

$\mathbf{I}_S$ be the $S$ x $S$ identity matrix

$\mathbf{I}_H$ be the $H$ x $H$ identity matrix

$\mathbf{0}_{S\mathrm{x}H}$ be the $S$ x $H$ zero matrix

$\mathbf{G}_{\mathrm{LT}}$ be the $K$x$L$ generator matrix of the encoding symbols generated by the LT Encoder.

So,

$\mathbf{G}_{\mathrm{LT}} \cdot (C[0], \ldots, C[L-1])^{\mathrm{T}} = (C''[0], C''[1], \ldots, C''[K-1])^{\mathrm{T}}$

i.e. $\mathbf{G}_{\mathrm{LT}i,j} = 1$ if and only if $C[i]$ is included in the symbols which are XORed to produce LTEnc[$K$, ($C$[0], …, $C$[$L$-1]), ($d$[$i$], $a$[$i$], $b$[$i$])].

Then:

The first $S$ rows of **A** are equal to $\mathbf{G}_{\mathrm{LDPC}} \mid \mathbf{I}_S \mid \mathbf{Z}_{S\mathrm{x}H}$.

The next $H$ rows of **A** are equal to $\mathbf{G}_{\mathrm{Half}} \mid \mathbf{I}_H$.

The remaining $K$ rows of **A** are equal to $\mathbf{G}_{\mathrm{LT}}$.

The matrix **A** is depicted in the figure below:

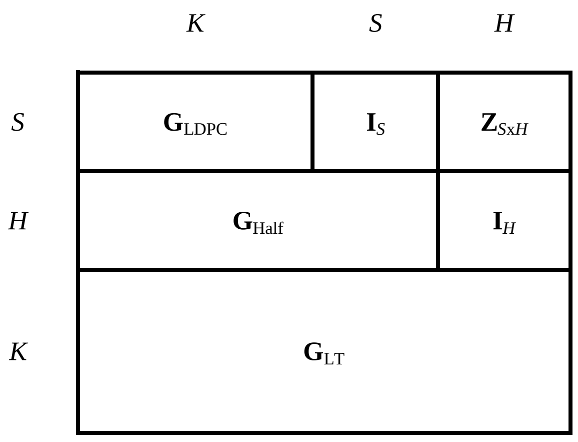


**Figure B.5.2.5.2-1: The matrix A**

The intermediate symbols can then be calculated as:

$\mathbf{C} = \mathbf{A}^{-1} \cdot \mathbf{D}$

The source triples are generated such that for any $K$ matrix **A** has full rank and is therefore invertible. This calculation can be realized by applying a Raptor decoding process to the $K$ source symbols $C''[0], C''[1], \ldots, C''[K-1]$ to produce the $L$ intermediate symbols $C[0], C[1], \ldots, C[L-1]$.

To efficiently generate the intermediate symbols from the source symbols, it is recommended that an efficient decoder implementation such as that described in Section B.8 be used. The source symbol triples are designed to facilitate efficient decoding of the source symbols using that algorithm.

## B.5.3 Second encoding step: LT encoding

In the second encoding step, the repair symbol with ESI $X$ is generated by applying the generator LTEnc[$K$, ($C$[0], $C$[1],…, $C$[$L$-1]), ($d$, $a$, $b$)] defined in Section B.5.4 to the $L$ intermediate symbols $C$[0], $C$[1],…, $C$[$L$-1] using the triple ($d$, $a$, $b$)=Trip[$K$,$X$] generated according to Sections B.3.2.2 and B.4.2.

## B.5.4 Generators

### B.5.4.1 Random Generator

The random number generator Rand[$X$, $i$, $m$] is defined as follows, where $X$ is a non-negative integer, $i$ is a non-negative integer and $m$ is a positive integer and the value produced is an integer between 0 and $m$-1. Let $V_0$ and $V_1$ be arrays of 256 entries each, where each entry is a 4-byte unsigned integer. These arrays are provided in Section B.7.

Then,

Rand[$X$, $i$, $m$] = ($V_0$[($X$ + $i$) % 256] ^ $V_1$[(floor($X$/256)+ $i$) % 256]) % $m$

### B.5.4.2 Degree Generator

The degree generator Deg[$v$] is defined as follows, where $v$ is an integer that is at least 0 and less than $2^{20}$ = 1048576.

In Table 7.3.2-1, find the index $j$ such that $f[j-1] \leq v < f[j]$

Deg[$v$] = $d[j]$

**Table 7.3.2-1 – Defines the degree distribution for encoding symbols**

| Index *j* | *f*[*j*] | *d*[*j*] |
|---|---|---|
| 0 | 0 | -- |
| 1 | 10241 | 1 |
| 2 | 491582 | 2 |
| 3 | 712794 | 3 |
| 4 | 831695 | 4 |
| 5 | 948446 | 10 |
| 6 | 1032189 | 11 |
| 7 | 1048576 | 40 |

### B.5.4.3 LT Encoding Symbol Generator

The encoding symbol generator LTEnc[$K$, ($C$[0], $C$[1],…, $C$[$L$-1]), ($d$, $a$, $b$)] takes the following inputs:

$K$ is the number of source symbols (or sub-symbols) for the source block (sub-block). Let $L$ be derived from $K$ as described in Section B.5.2, and let $L''$ be the smallest prime integer greater than or equal to $L$.

($C$[0], $C$[1],…, $C$[$L$-1]) is the array of $L$ intermediate symbols (sub-symbols) generated as described in Section B.5.2

($d$, $a$, $b$) is a source triple determined using the Triple generator defined in Section B.5.3.4, whereby

$d$ is an integer denoting an encoding symbol degree

$a$ is an integer between 1 and $L''$-1 inclusive

$b$ is an integer between 0 and $L''$-1 inclusive

The encoding symbol generator produces a single encoding symbol as output, according to the following algorithm:

```
While ($b \geq L$) do $b = (b + a) \% L''$
LTEnc[$K$, ($C[0], C[1],\ldots, C[L-1]$), ($d, a, b$)] = $C[b]$.
For $j = 1,\ldots,\min(d-1,L-1)$ do
    $b = (b + a) \% L''$
    While ($b \geq L$) do $b = (b + a) \% L''$
```

LTEnc[$K$, ($C$[0], $C$[1],…, $C$[$L$-1]), ($d$, $a$, $b$)] = *LTEnc*[$K$, ($C$[0], $C$[1],…, $C$[$L$-1]), ($d$, $a$, $b$)] ^ $C$[$b$]

### B.5.4.4 Triple generator

The triple generator Trip[$K$,$X$] takes the following inputs:

$K$ The number of source symbols

$X$ An encoding symbol ID

Let

$L$ be determined from $K$ as described in Section B.5.2

$L''$ be the smallest prime that is greater than or equal to $L$

$Q$ = 65521, the largest prime smaller than $2^{16}$.

$J(K)$ be the systematic index associated with $K$, as defined in Section B.7

The output of the triple generator is a triples, ($d$, $a$, $b$) determined as follows:

1. $A$ = (53591 + $J(K)\cdot 997$) % Q
2. $B$ = $10267\cdot(J(K)+1)$ % $Q$
3. $Y$ = ($B + X\cdot A$) % $Q$
4. $v$ = Rand[$Y$, 0, $2^{20}$]
5. $d$ = Deg[$v$]
6. $a$ = 1 + Rand[$Y$, 1, $L''$-1]
7. $b$ = Rand[$Y$, 2, $L''$]

## B.6 Systematic Indices J(*K*))

For each value of *K* the systematic index *J*(*K*) is designed to have the property that the set of source symbol triples (*d*[0], *a*[0], *b*[0]), …, (*d*[*L*-1], *a*[*L*-1], *b*[*L*-1]) are such that the *L* intermediate symbols are uniquely defined, i.e. the matrix **A** in Section B.5.2.4.2 has full rank and is therefore invertible.

The following is the list of the systematic indices for values of *K* between 4 and 8192 inclusive,

18, 14, 61, 46, 14, 22, 20, 40, 48, 1, 29, 40, 43, 46, 18, 8, 20, 2, 61, 26, 13, 29, 36, 19, 58, 5, 58, 0, 54, 56, 24, 14, 5, 67, 39, 31, 25, 29, 24, 19, 14, 56, 49, 49, 63, 30, 4, 39, 2, 1, 20, 19, 61, 4, 54, 70, 25, 52, 9, 26, 55, 69, 27, 68, 75, 19, 64, 57, 45, 3, 37, 31, 100, 41, 25, 41, 53, 23, 9, 31, 26, 30, 30, 46,
90, 50, 13, 90, 77, 61, 31, 54, 54, 3, 21, 66, 21, 11, 23, 11, 29, 21, 7, 1, 27, 4, 34, 17, 85, 69, 17, 75, 93, 57, 0, 53, 71, 88, 119, 88, 90, 22, 0, 58, 41, 22, 96, 26, 79, 118, 19, 3, 81, 72, 50, 0, 32, 79, 28, 25, 12, 25, 29, 3, 37, 30, 30, 41, 84, 32, 31, 61, 32, 61, 7, 56, 54, 39, 33, 66, 29, 3, 14, 75, 75, 78, 84,
75, 84, 25, 54, 25, 25, 107, 78, 27, 73, 0, 49, 96, 53, 50, 21, 10, 73, 58, 65, 27, 3, 27, 18, 54, 45, 69, 29, 3, 65, 31, 71, 76, 56, 54, 76, 54, 13, 5, 18, 142, 17, 3, 37, 114, 41, 25, 56, 0, 23, 3, 41, 22, 22, 31, 18, 48, 31, 58, 37, 75, 88, 3, 56, 1, 95, 19, 73, 52, 52, 4, 75, 26, 1, 25, 10, 1, 70, 31, 31, 12, 10, 54,
46, 11, 74, 84, 74, 8, 58, 23, 74, 8, 36, 11, 16, 94, 76, 14, 57, 65, 8, 22, 10, 36, 36, 96, 62, 103, 6, 75, 103, 58, 10, 15, 41, 75, 125, 58, 15, 10, 34, 29, 34, 4, 16, 29, 18, 18, 28, 71, 28, 43, 77, 18, 41, 41, 41, 62, 29, 96, 15, 106, 43, 15, 3, 43, 61, 3, 18, 103, 77, 29, 103, 19, 58, 84, 58, 1, 146, 32, 3, 70, 52,
54, 29, 70, 69, 124, 62, 1, 26, 38, 26, 3, 16, 26, 5, 51, 120, 41, 16, 1, 43, 34, 34, 29, 37, 56, 29, 96, 86, 54, 25, 84, 50, 34, 34, 93, 84, 96, 29, 29, 50, 50, 6, 1, 105, 78, 15, 37, 19, 50, 71, 36, 6, 54, 8, 28, 54, 75, 75, 16, 75, 131, 5, 25, 16, 69, 17, 69, 6, 96, 53, 96, 41, 119, 6, 6, 88, 50, 88, 52, 37, 0, 124,
73, 73, 7, 14, 36, 69, 79, 6, 114, 40, 79, 17, 77, 24, 44, 37, 69, 27, 37, 29, 33, 37, 50, 31, 69, 29, 101, 7, 61, 45, 17, 73, 37, 34, 18, 94, 22, 22, 63, 3, 25, 25, 17, 3, 90, 34, 34, 41, 34, 41, 54, 41, 54, 41, 41, 41, 163, 143, 96, 18, 32, 39, 86, 104, 11, 17, 17, 11, 86, 104, 78, 70, 52, 78, 17, 73, 91, 62, 7, 128,
50, 124, 18, 101, 46, 10, 75, 104, 73, 58, 132, 34, 13, 4, 95, 88, 33, 76, 74, 54, 62, 113, 114, 103, 32, 103, 69, 54, 53, 3, 11, 72, 31, 53, 102, 37, 53, 11, 81, 41, 10, 164, 10, 41, 31, 36, 113, 82, 3, 125, 62, 16, 4, 41, 41, 4, 128, 49, 138, 128, 74, 103, 0, 6, 101, 41, 142, 171, 39, 105, 121, 81, 62, 41, 81,
37, 3, 81, 69, 62, 3, 69, 70, 21, 29, 4, 91, 87, 37, 79, 36, 21, 71, 37, 41, 75, 128, 128, 15, 25, 3, 108, 73, 91, 62, 114, 62, 62, 36, 36, 15, 58, 114, 61, 114, 58, 105, 114, 41, 61, 176, 145, 46, 37, 30, 220, 77, 138, 15, 1, 128, 53, 50, 50, 58, 8, 91, 114, 105, 63, 91, 37, 37, 13, 169, 51, 102, 6, 102, 23, 105,
23, 58, 6, 29, 29, 19, 82, 29, 13, 36, 27, 29, 61, 12, 18, 127, 127, 12, 44, 102, 18, 4, 15, 206, 53, 127, 53, 17, 69, 69, 69, 29, 29, 109, 25, 102, 25, 53, 62, 99, 62, 62, 29, 62, 62, 45, 91, 125, 29, 29, 29, 4, 117, 72, 4, 30, 71, 71, 95, 79, 179, 71, 30, 53, 32, 32, 49, 25, 91, 25, 26, 26, 103, 123, 26, 41, 162,
78, 52, 103, 25, 6, 142, 94, 45, 45, 94, 127, 94, 94, 94, 47, 209, 138, 39, 39, 19, 154, 73, 67, 91, 27, 91, 84, 4, 84, 91, 12, 14, 165, 142, 54, 69, 192, 157, 185, 8, 95, 25, 62, 103, 103, 95, 71, 97, 62, 128, 0, 29, 51, 16, 94, 16, 16, 51, 0, 29, 85, 10, 105, 16, 29, 29, 13, 29, 4, 4, 132, 23, 95, 25, 54, 41, 29,
50, 70, 58, 142, 72, 70, 15, 72, 54, 29, 22, 145, 29, 127, 29, 85, 58, 101, 34, 165, 91, 46, 46, 25, 185, 25, 77, 128, 46, 128, 46, 188, 114, 46, 25, 45, 45, 114, 145, 114, 15, 102, 142, 8, 73, 31, 139, 157, 13, 79, 13, 114, 150, 8, 90, 91, 123, 69, 82, 132, 8, 18, 10, 102, 103, 114, 103, 8, 103, 13, 115, 55,
62, 3, 8, 154, 114, 99, 19, 8, 31, 73, 19, 99, 10, 6, 121, 32, 13, 32, 119, 32, 29, 145, 30, 13, 13, 114, 145, 32, 1, 123, 39, 29, 31, 69, 31, 140, 72, 72, 25, 25, 123, 25, 123, 8, 4, 85, 8, 25, 39, 25, 39, 85, 138, 25, 138, 25, 33, 102, 70, 25, 25, 31, 25, 25, 192, 69, 69, 114, 145, 120, 120, 8, 33, 98, 15, 212,
155, 8, 101, 8, 8, 98, 68, 155, 102, 132, 120, 30, 25, 123, 123, 101, 25, 123, 32, 24, 94, 145, 32, 24, 94, 118, 145, 101, 53, 53, 25, 128, 173, 142, 81, 81, 69, 33, 33, 125, 4, 1, 17, 27, 4, 17, 102, 27, 13, 25, 128, 71, 13, 39, 53, 13, 53, 47, 39, 23, 128, 53, 39, 47, 39, 135, 158, 136, 36, 36, 27, 157, 47, 76,
213, 47, 156, 25, 25, 53, 25, 53, 25, 86, 27, 159, 25, 62, 79, 39, 79, 25, 145, 49, 25, 143, 13, 114, 150, 130, 94, 102, 39, 4, 39, 61, 77, 228, 22, 25, 47, 119, 205, 122, 119, 205, 119, 22, 119, 258, 143, 22, 81, 179, 22, 22, 143, 25, 65, 53, 168, 36, 79, 175, 37, 79, 70, 79, 103, 70, 25, 175, 4, 96, 96, 49,
128, 138, 96, 22, 62, 47, 95, 105, 95, 62, 95, 62, 142, 103, 69, 103, 30, 103, 34, 173, 127, 70, 127, 132, 18, 85, 22, 71, 18, 206, 206, 18, 128, 145, 70, 193, 188, 8, 125, 114, 70, 128, 114, 145, 102, 25, 12, 108, 102, 94, 10, 102, 1, 102, 124, 22, 22, 118, 132, 22, 116, 75, 41, 63, 41, 189, 208, 55, 85, 69,
8, 71, 53, 71, 69, 102, 165, 41, 99, 69, 33, 33, 29, 156, 102, 13, 251, 102, 25, 13, 109, 102, 164, 102, 164, 102, 25, 29, 228, 29, 259, 179, 222, 95, 94, 30, 30, 30, 142, 55, 142, 72, 55, 102, 128, 17, 69, 164, 165, 3, 164, 36, 165, 27, 27, 45, 21, 21, 237, 113, 83, 231, 106, 13, 154, 13, 154, 128, 154, 148,
258, 25, 154, 128, 3, 27, 10, 145, 145, 21, 146, 25, 1, 185, 121, 0, 1, 95, 55, 95, 95, 30, 0, 27, 95, 0, 95, 8, 222, 27, 121, 30, 95, 121, 0, 98, 94, 131, 55, 95, 95, 30, 98, 30, 0, 91, 145, 66, 179, 66, 58, 175, 29, 0, 31, 173, 146, 160, 39, 53, 28, 123, 199, 123, 175, 146, 156, 54, 54, 149, 25, 70, 178, 128,
25, 70, 70, 94, 224, 54, 4, 54, 54, 25, 228, 160, 206, 165, 143, 206, 108, 220, 234, 160, 13, 169, 103, 103, 103, 91, 213, 222, 91, 103, 91, 103, 31, 30, 123, 13, 62, 103, 50, 106, 42, 13, 145, 114, 220, 65, 8, 8, 175, 11, 104, 94, 118, 132, 27, 118, 193, 27, 128, 127, 127, 183, 33, 30, 29, 103, 128, 61,
234, 165, 41, 29, 193, 33, 207, 41, 165, 165, 55, 81, 157, 157, 8, 81, 11, 27, 8, 8, 98, 96, 142, 145, 41, 179, 112, 62, 180, 206, 206, 165, 39, 241, 45, 151, 26, 197, 102, 192, 125, 128, 67, 128, 69, 128, 197, 33, 125, 102, 13, 103, 25, 30, 12, 30, 12, 30, 25, 77, 12, 25, 180, 27, 10, 69, 235, 228, 343, 118,
69, 41, 8, 69, 175, 25, 69, 25, 125, 41, 25, 41, 8, 155, 146, 155, 146, 155, 206, 168, 128, 157, 27, 273, 211, 211, 168, 11, 173, 154, 77, 173, 77, 102, 102, 102, 8, 85, 95, 102, 157, 28, 122, 234, 122, 157, 235, 222, 241, 10, 91, 179, 25, 13, 25, 41, 25, 206, 41, 6, 41, 158, 206, 206, 33, 296, 296, 33, 228,
69, 8, 114, 148, 33, 29, 66, 27, 27, 30, 233, 54, 173, 108, 106, 108, 108, 53, 103, 33, 33, 33, 176, 27, 27, 205, 164, 105, 237, 41, 27, 72, 165, 29, 29, 259, 132, 132, 132, 364, 71, 71, 27, 94, 160, 127, 51, 234, 55, 27, 95, 94, 165, 55, 55, 41, 0, 41, 128, 4, 123, 173, 6, 164, 157, 121, 121, 154, 86, 164,
164, 25, 93, 164, 25, 164, 210, 284, 62, 93, 30, 25, 25, 30, 30, 260, 130, 25, 125, 57, 53, 166, 166, 166, 185, 166, 158, 94, 113, 215, 159, 62, 99, 21, 172, 99, 184, 62, 259, 4, 21, 21, 77, 62, 173, 41, 146, 6, 41, 128, 121, 41, 11, 121, 103, 159, 164, 175, 206, 91, 103, 164, 72, 25, 129, 72, 206, 129, 33,
103, 102, 102, 29, 13, 11, 251, 234, 135, 31, 8, 123, 65, 91, 121, 129, 65, 243, 10, 91, 8, 65, 70, 228, 220, 243, 91, 10, 10, 30, 178, 91, 178, 33, 21, 25, 235, 165, 11, 161, 158, 27, 27, 30, 128, 75, 36, 30, 36, 36, 173, 25, 33, 178, 112, 162, 112, 112, 112, 162, 33, 33, 178, 123, 123, 39, 106, 91, 106,
106, 158, 106, 106, 284, 39, 230, 21, 228, 11, 21, 228, 159, 241, 62, 10, 62, 10, 68, 234, 39, 39, 138, 62, 22, 27, 183, 22, 215, 10, 175, 175, 353, 228, 42, 193, 175, 175, 27, 98, 27, 193, 150, 27, 173, 17, 233, 233, 25, 102, 123, 152, 242, 108, 4, 94, 176, 13, 41, 219, 17, 151, 22, 103, 103, 53, 128, 233,
284, 25, 265, 128, 39, 39, 138, 42, 39, 21, 86, 95, 127, 29, 91, 46, 103, 103, 215, 25, 123, 123, 230, 25, 193, 180, 30, 60, 30, 242, 136, 180, 193, 30, 206, 180, 60, 165, 206, 193, 165, 123, 164, 103, 68, 25, 70, 91, 25, 82, 53, 82, 186, 53, 82, 53, 25, 30, 282, 91, 13, 234, 160, 160, 126, 149, 36, 36, 160,
149, 178, 160, 39, 294, 149, 149, 160, 39, 95, 221, 186, 106, 178, 316, 267, 53, 53, 164, 159, 164, 165, 94, 228, 53, 52, 178, 183, 53, 294, 128, 55, 140, 294, 25, 95, 366, 15, 304, 13, 183, 77, 230, 6, 136, 235, 121, 311, 273, 36, 158, 235, 230, 98, 201, 165, 165, 165, 91, 175, 248, 39, 185, 128, 39, 39,
128, 313, 91, 36, 219, 130, 25, 130, 234, 234, 130, 234, 121, 205, 304, 94, 77, 64, 259, 60, 60, 60, 77, 242, 60, 145, 95, 270, 18, 91, 199, 159, 91, 235, 58, 249, 26, 123, 114, 29, 15, 191, 15, 30, 55, 55, 347, 4, 29, 15, 4, 341, 93, 7, 30, 23, 7, 121, 266, 178, 261, 70, 169, 25, 25, 158, 169, 25, 169, 270,
270, 13, 128, 327, 103, 55, 128, 103, 136, 159, 103, 327, 41, 32, 111, 111, 114, 173, 215, 173, 25, 173, 180, 114, 173, 173, 98, 93, 25, 160, 157, 159, 160, 159, 159, 160, 320, 35, 193, 221, 33, 36, 136, 248, 91, 215, 125, 215, 156, 68, 125, 125, 1, 287, 123, 94, 30, 184, 13, 30, 94, 123, 206, 12, 206,
289, 128, 122, 184, 128, 289, 178, 29, 26, 206, 178, 65, 206, 128, 192, 102, 197, 36, 94, 94, 155, 10, 36, 121, 280, 121, 368, 192, 121, 121, 179, 121, 36, 54, 192, 121, 192, 197, 118, 123, 224, 118, 10, 192, 10, 91, 269, 91, 49, 206, 184, 185, 62, 8, 49, 289, 30, 5, 55, 30, 42, 39, 220, 298, 42, 347, 42,
234, 42, 70, 42, 55, 321, 129, 172, 173, 172, 13, 98, 129, 325, 235, 284, 362, 129, 233, 345, 175, 261, 175, 60, 261, 58, 289, 99, 99, 99, 206, 99, 36, 175, 29, 25, 432, 125, 264, 168, 173, 69, 158, 273, 179, 164, 69, 158, 69, 8, 95, 192, 30, 164, 101, 44, 53, 273, 335, 273, 53, 45, 128, 45, 234, 123, 105,
103, 103, 224, 36, 90, 211, 282, 264, 91, 228, 91, 166, 264, 228, 398, 50, 101, 91, 264, 73, 36, 25, 73, 50, 50, 242, 36, 36, 58, 165, 204, 353, 165, 125, 320, 128, 298, 298, 180, 128, 60, 102, 30, 30, 53, 179, 234, 325, 234, 175, 21, 250, 215, 103, 21, 21, 250, 91, 211, 91, 313, 301, 323, 215, 228, 160,
29, 29, 81, 53, 180, 146, 248, 66, 159, 39, 98, 323, 98, 36, 95, 218, 234, 39, 82, 82, 230, 62, 13, 62, 230, 13, 30, 98, 0, 8, 98, 8, 98, 91, 267, 121, 197, 30, 78, 27, 78, 102, 27, 298, 160, 103, 264, 264, 264, 175, 17, 273, 273, 165, 31, 160, 17, 99, 17, 99, 234, 31, 17, 99, 36, 26, 128, 29, 214, 353, 264,
102, 36, 102, 264, 264, 273, 273, 4, 16, 138, 138, 264, 128, 313, 25, 420, 60, 10, 280, 264, 60, 60, 103, 178, 125, 178, 29, 327, 29, 36, 30, 36, 4, 52, 183, 183, 173, 52, 31, 173, 31, 158, 31, 158, 31, 9, 31, 31, 353, 31, 353, 173, 415, 9, 17, 222, 31, 103, 31, 165, 27, 31, 31, 165, 27, 27, 206, 31, 31, 4, 4,
30, 4, 4, 264, 185, 159, 310, 273, 310, 173, 40, 4, 173, 4, 173, 4, 250, 250, 62, 188, 119, 250, 233, 62, 121, 105, 105, 54, 103, 111, 291, 236, 236, 103, 297, 36, 26, 316, 69, 183, 158, 206, 129, 160, 129, 184, 55, 179, 279, 11, 179, 347, 160, 184, 129, 179, 351, 179, 353, 179, 129, 129, 351, 11, 111,
93, 93, 235, 103, 173, 53, 93, 50, 111, 86, 123, 94, 36, 183, 60, 55, 55, 178, 219, 253, 321, 178, 235, 235, 183, 183, 204, 321, 219, 160, 193, 335, 121, 70, 69, 295, 159, 297, 231, 121, 231, 136, 353, 136, 121, 279, 215, 366, 215, 353, 159, 353, 353, 103, 31, 31, 298, 298, 30, 30, 165, 273, 25, 219, 35,
165, 259, 54, 36, 54, 54, 165, 71, 250, 327, 13, 289, 165, 196, 165, 165, 94, 233, 165, 94, 60, 165, 96, 220, 166, 271, 158, 397, 122, 53, 53, 137, 280, 272, 62, 30, 30, 30, 105, 102, 67, 140, 8, 67, 21, 270, 298, 69, 173, 298, 91, 179, 327, 86, 179, 88, 179, 179, 55, 123, 220, 233, 94, 94, 175, 13, 53, 13,
154, 191, 74, 83, 83, 325, 207, 83, 74, 83, 325, 74, 316, 388, 55, 55, 364, 55, 183, 434, 273, 273, 273, 164, 213, 11, 213, 327, 321, 21, 352, 185, 103, 13, 13, 55, 30, 323, 123, 178, 435, 178, 30, 175, 175, 30, 481, 527, 175, 125, 232, 306, 232, 206, 306, 364, 206, 270, 206, 232, 10, 30, 130, 160, 130,
347, 240, 30, 136, 130, 347, 136, 279, 298, 206, 30, 103, 273, 241, 70, 206, 306, 434, 206, 94, 94, 156, 161, 321, 321, 64, 161, 13, 183, 183, 83, 161, 13, 169, 13, 159, 36, 173, 159, 36, 36, 230, 235, 235, 159, 159, 335, 312, 42, 342, 264, 39, 39, 39, 34, 298, 36, 36, 252, 164, 29, 493, 29, 387, 387,
435, 493, 132, 273, 105, 132, 74, 73, 206, 234, 273, 206, 95, 15, 280, 280, 280, 280, 397, 273, 273, 242, 397, 280, 397, 397, 397, 273, 397, 280, 230, 137, 353, 67, 81, 137, 137, 353, 259, 312, 114, 164, 164, 25, 77, 21, 77, 165, 30, 30, 231, 234, 121, 234, 312, 121, 364, 136, 123, 123, 136, 123, 136,
150, 264, 285, 30, 166, 93, 30, 39, 224, 136, 39, 355, 355, 397, 67, 67, 25, 67, 25, 298, 11, 67, 264, 374, 99, 150, 321, 67, 70, 67, 295, 150, 29, 321, 150, 70, 29, 142, 355, 311, 173, 13, 253, 103, 114, 114, 70, 192, 22, 128, 128, 183, 184, 70, 77, 215, 102, 292, 30, 123, 279, 292, 142, 33, 215, 102,
468, 123, 468, 473, 30, 292, 215, 30, 213, 443, 473, 215, 234, 279, 279, 279, 279, 265, 443, 206, 66, 313, 34, 30, 206, 30, 51, 15, 206, 41, 434, 41, 398, 67, 30, 301, 67, 36, 3, 285, 437, 136, 136, 22, 136, 145, 365, 323, 323, 145, 136, 22, 453, 99, 323, 353, 9, 258, 323, 231, 128, 231, 382, 150, 420,
39, 94, 29, 29, 353, 22, 22, 347, 353, 39, 29, 22, 183, 8, 284, 355, 388, 284, 60, 64, 99, 60, 64, 150, 95, 150, 364, 150, 95, 150, 6, 236, 383, 544, 81, 206, 388, 206, 58, 159, 99, 231, 228, 363, 363, 121, 99, 121, 121, 99, 422, 544, 273, 173, 121, 427, 102, 121, 235, 284, 179, 25, 197, 25, 179, 511, 70,
368, 70, 25, 388, 123, 368, 159, 213, 410, 159, 236, 127, 159, 21, 373, 184, 424, 327, 250, 176, 176, 175, 284, 316, 176, 284, 327, 111, 250, 284, 175, 175, 264, 111, 176, 219, 111, 427, 427, 176, 284, 427, 353, 428, 55, 184, 493, 158, 136, 99, 287, 264, 334, 264, 213, 213, 292, 481, 93, 264, 292,
295, 295, 6, 367, 279, 173, 308, 285, 158, 308, 335, 299, 137, 137, 572, 41, 137, 137, 41, 94, 335, 220, 36, 224, 420, 36, 265, 265, 91, 91, 71, 123, 264, 91, 91, 123, 107, 30, 22, 292, 35, 241, 356, 298, 14, 298, 441, 35, 121, 71, 63, 130, 63, 488, 363, 71, 63, 307, 194, 71, 71, 220, 121, 125, 71, 220,
71, 71, 71, 71, 235, 265, 353, 128, 155, 128, 420, 400, 130, 173, 183, 183, 184, 130, 173, 183, 13, 183, 130, 130, 183, 183, 353, 353, 183, 242, 183, 183, 306, 324, 324, 321, 306, 321, 6, 6, 128, 306, 242, 242, 306, 183, 183, 6, 183, 321, 486, 183, 164, 30, 78, 138, 158, 138, 34, 206, 362, 55, 70, 67,
21, 375, 136, 298, 81, 298, 298, 298, 230, 121, 30, 230, 311, 240, 311, 311, 158, 204, 136, 136, 184, 136, 264, 311, 311, 312, 312, 72, 311, 175, 264, 91, 175, 264, 121, 461, 312, 312, 238, 475, 350, 512, 350, 312, 313, 350, 312, 366, 294, 30, 253, 253, 253, 388, 158, 388, 22, 388, 22, 388, 103, 321,
321, 253, 7, 437, 103, 114, 242, 114, 114, 242, 114, 114, 242, 242, 242, 306, 242, 114, 7, 353, 335, 27, 241, 299, 312, 364, 506, 409, 94, 462, 230, 462, 243, 230, 175, 175, 462, 461, 230, 428, 426, 175, 175, 165, 175, 175, 372, 183, 572, 102, 85, 102, 538, 206, 376, 85, 85, 284, 85, 85, 284, 398, 83,
160, 265, 308, 398, 310, 583, 289, 279, 273, 285, 490, 490, 211, 292, 292, 158, 398, 30, 220, 169, 368, 368, 368, 169, 159, 368, 93, 368, 368, 93, 169, 368, 368, 443, 368, 298, 443, 368, 298, 538, 345, 345, 311, 178, 54, 311, 215, 178, 175, 222, 264, 475, 264, 264, 475, 478, 289, 63, 236, 63, 299,
231, 296, 397, 299, 158, 36, 164, 164, 21, 492, 21, 164, 21, 164, 403, 26, 26, 588, 179, 234, 169, 465, 295, 67, 41, 353, 295, 538, 161, 185, 306, 323, 68, 420, 323, 82, 241, 241, 36, 53, 493, 301, 292, 241, 250, 63, 63, 103, 442, 353, 185, 353, 321, 353, 185, 353, 353, 185, 409, 353, 589, 34, 271, 271,
34, 86, 34, 34, 353, 353, 39, 414, 4, 95, 95, 4, 225, 95, 4, 121, 30, 552, 136, 159, 159, 514, 159, 159, 54, 514, 206, 136, 206, 159, 74, 235, 235, 312, 54, 312, 42, 156, 422, 629, 54, 465, 265, 165, 250, 35, 165, 175, 659, 175, 175, 8, 8, 8, 8, 206, 206, 206, 50, 435, 206, 432, 230, 230, 234, 230, 94, 299,
299, 285, 184, 41, 93, 299, 299, 285, 41, 285, 158, 285, 206, 299, 41, 36, 396, 364, 364, 120, 396, 514, 91, 382, 538, 807, 717, 22, 93, 412, 54, 215, 54, 298, 308, 148, 298, 148, 298, 308, 102, 656, 6, 148, 745, 128, 298, 64, 407, 273, 41, 172, 64, 234, 250, 398, 181, 445, 95, 236, 441, 477, 504, 102,
196, 137, 364, 60, 453, 137, 364, 367, 334, 364, 299, 196, 397, 630, 589, 589, 196, 646, 337, 235, 128, 128, 343, 289, 235, 324, 427, 324, 58, 215, 215, 461, 425, 461, 387, 440, 285, 440, 440, 285, 387, 632, 325, 325, 440, 461, 425, 425, 387, 627, 191, 285, 440, 308, 55, 219, 280, 308, 265, 538, 183,
121, 30, 236, 206, 30, 455, 236, 30, 30, 705, 83, 228, 280, 468, 132, 8, 132, 132, 128, 409, 173, 353, 132, 409, 35, 128, 450, 137, 398, 67, 432, 423, 235, 235, 388, 306, 93, 93, 452, 300, 190, 13, 452, 388, 30, 452, 13, 30, 13, 30, 306, 362, 234, 721, 635, 809, 784, 67, 498, 498, 67, 353, 635, 67, 183,
159, 445, 285, 183, 53, 183, 445, 265, 432, 57, 420, 432, 420, 477, 327, 55, 60, 105, 183, 218, 104, 104, 475, 239, 582, 151, 239, 104, 732, 41, 26, 784, 86, 300, 215, 36, 64, 86, 86, 675, 294, 64, 86, 528, 550, 493, 565, 298, 230, 312, 295, 538, 298, 295, 230, 54, 374, 516, 441, 54, 54, 323, 401, 401,
382, 159, 837, 159, 54, 401, 592, 159, 401, 417, 610, 264, 150, 323, 452, 185, 323, 323, 185, 403, 185, 423, 165, 425, 219, 407, 270, 231, 99, 93, 231, 631, 756, 71, 364, 434, 213, 86, 102, 434, 102, 86, 23, 71, 335, 164, 323, 409, 381, 4, 124, 41, 424, 206, 41, 124, 41, 41, 703, 635, 124, 493, 41, 41,
487, 492, 124, 175, 124, 261, 600, 488, 261, 488, 261, 206, 677, 261, 308, 723, 908, 704, 691, 723, 488, 488, 441, 136, 476, 312, 136, 550, 572, 728, 550, 22, 312, 312, 22, 55, 413, 183, 280, 593, 191, 36, 36, 427, 36, 695, 592, 19, 544, 13, 468, 13, 544, 72, 437, 321, 266, 461, 266, 441, 230, 409, 93,
521, 521, 345, 235, 22, 142, 150, 102, 569, 235, 264, 91, 521, 264, 7, 102, 7, 498, 521, 235, 537, 235, 6, 241, 420, 420, 631, 41, 527, 103, 67, 337, 62, 264, 527, 131, 67, 174, 263, 264, 36, 36, 263, 581, 253, 465, 160, 286, 91, 160, 55, 4, 4, 631, 631, 608, 365, 465, 294, 427, 427, 335, 669, 669, 129,
93, 93, 93, 93, 74, 66, 758, 504, 347, 130, 505, 504, 143, 505, 550, 222, 13, 352, 529, 291, 538, 50, 68, 269, 130, 295, 130, 511, 295, 295, 130, 486, 132, 61, 206, 185, 368, 669, 22, 175, 492, 207, 373, 452, 432, 327, 89, 550, 496, 611, 527, 89, 527, 496, 550, 516, 516, 91, 136, 538, 264, 264, 124,
264, 264, 264, 264, 264, 535, 264, 150, 285, 398, 285, 582, 398, 475, 81, 694, 694, 64, 81, 694, 234, 607, 723, 513, 234, 64, 581, 64, 124, 64, 607, 234, 723, 717, 367, 64, 513, 607, 488, 183, 488, 450, 183, 550, 286, 183, 363, 286, 414, 67, 449, 449, 366, 215, 235, 95, 295, 295, 41, 335, 21, 445, 225,
21, 295, 372, 749, 461, 53, 481, 397, 427, 427, 427, 714, 481, 714, 427, 717, 165, 245, 486, 415, 245, 415, 486, 274, 415, 441, 456, 300, 548, 300, 422, 422, 757, 11, 74, 430, 430, 136, 409, 430, 749, 191, 819, 592, 136, 364, 465, 231, 231, 918, 160, 589, 160, 160, 465, 465, 231, 157, 538, 538, 259,
538, 326, 22, 22, 22, 179, 22, 22, 550, 179, 287, 287, 417, 327, 498, 498, 287, 488, 327, 538, 488, 583, 488, 287, 335, 287, 335, 287, 41, 287, 335, 287, 327, 441, 335, 287, 488, 538, 327, 498, 8, 8, 374, 8, 64, 427, 8, 374, 417, 760, 409, 373, 160, 423, 206, 160, 106, 499, 160, 271, 235, 160, 590, 353,
695, 478, 619, 590, 353, 13, 63, 189, 420, 605, 427, 643, 121, 280, 415, 121, 415, 595, 417, 121, 398, 55, 330, 463, 463, 123, 353, 330, 582, 309, 582, 582, 405, 330, 550, 405, 582, 353, 309, 308, 60, 353, 7, 60, 71, 353, 189, 183, 183, 183, 582, 755, 189, 437, 287, 189, 183, 668, 481, 384, 384, 481,
481, 481, 477, 582, 582, 499, 650, 481, 121, 461, 231, 36, 235, 36, 413, 235, 209, 36, 689, 114, 353, 353, 235, 592, 36, 353, 413, 209, 70, 308, 70, 699, 308, 70, 213, 292, 86, 689, 465, 55, 508, 128, 452, 29, 41, 681, 573, 352, 21, 21, 648, 648, 69, 509, 409, 21, 264, 21, 509, 514, 514, 409, 21, 264,
443, 443, 427, 160, 433, 663, 433, 231, 646, 185, 482, 646, 433, 13, 398, 172, 234, 42, 491, 172, 234, 234, 832, 775, 172, 196, 335, 822, 461, 298, 461, 364, 1120, 537, 169, 169, 364, 694, 219, 612, 231, 740, 42, 235, 321, 279, 960, 279, 353, 492, 159, 572, 321, 159, 287, 353, 287, 287, 206, 206,
321, 287, 159, 321, 492, 159, 55, 572, 600, 270, 492, 784, 173, 91, 91, 443, 443, 582, 261, 497, 572, 91, 555, 352, 206, 261, 555, 285, 91, 555, 497, 83, 91, 619, 353, 488, 112, 4, 592, 295, 295, 488, 235, 231, 769, 568, 581, 671, 451, 451, 483, 299, 1011, 432, 422, 207, 106, 701, 508, 555, 508, 555,
125, 870, 555, 589, 508, 125, 749, 482, 125, 125, 130, 544, 643, 643, 544, 488, 22, 643, 130, 335, 544, 22, 130, 544, 544, 488, 426, 426, 4, 180, 4, 695, 35, 54, 433, 500, 592, 433, 262, 94, 401, 401, 106, 216, 216, 106, 521, 102, 462, 518, 271, 475, 365, 193, 648, 206, 424, 206, 193, 206, 206, 424,
299, 590, 590, 364, 621, 67, 538, 488, 567, 51, 51, 513, 194, 81, 488, 486, 289, 567, 563, 749, 563, 338, 338, 502, 563, 822, 338, 563, 338, 502, 201, 230, 201, 533, 445, 175, 201, 175, 13, 85, 960, 103, 85, 175, 30, 445, 445, 175, 573, 196, 877, 287, 356, 678, 235, 489, 312, 572, 264, 717, 138, 295,
6, 295, 523, 55, 165, 165, 295, 138, 663, 6, 295, 6, 353, 138, 6, 138, 169, 129, 784, 12, 129, 194, 605, 784, 445, 234, 627, 563, 689, 627, 647, 570, 627, 570, 647, 206, 234, 215, 234, 816, 627, 816, 234, 627, 215, 234, 627, 264, 427, 427, 30, 424, 161, 161, 916, 740, 180, 616, 481, 514, 383, 265, 481,
164, 650, 121, 582, 689, 420, 669, 589, 420, 788, 549, 165, 734, 280, 224, 146, 681, 788, 184, 398, 784, 4, 398, 417, 417, 398, 636, 784, 417, 81, 398, 417, 81, 185, 827, 420, 241, 420, 41, 185, 185, 718, 241, 101, 185, 185, 241, 241, 241, 241, 241, 185, 324, 420, 420, 1011, 420, 827, 241, 184, 563,
241, 183, 285, 529, 285, 808, 822, 891, 822, 488, 285, 486, 619, 55, 869, 39, 567, 39, 289, 203, 158, 289, 710, 818, 158, 818, 355, 29, 409, 203, 308, 648, 792, 308, 308, 91, 308, 6, 592, 792, 106, 106, 308, 41, 178, 91, 751, 91, 259, 734, 166, 36, 327, 166, 230, 205, 205, 172, 128, 230, 432, 623, 838,
623, 432, 278, 432, 42, 916, 432, 694, 623, 352, 452, 93, 314, 93, 93, 641, 88, 970, 914, 230, 61, 159, 270, 159, 493, 159, 755, 159, 409, 30, 30, 836, 128, 241, 99, 102, 984, 538, 102, 102, 273, 639, 838, 102, 102, 136, 637, 508, 627, 285, 465, 327, 327, 21, 749, 327, 749, 21, 845, 21, 21, 409, 749,
1367, 806, 616, 714, 253, 616, 714, 714, 112, 375, 21, 112, 375, 375, 51, 51, 51, 51, 393, 206, 870, 713, 193, 802, 21, 1061, 42, 382, 42, 543, 876, 42, 876, 382, 696, 543, 635, 490, 353, 353, 417, 64, 1257, 271, 64, 377, 127, 127, 537, 417, 905, 353, 538, 465, 605, 876, 427, 324, 514, 852, 427, 53,
427, 557, 173, 173, 7, 1274, 563, 31, 31, 31, 745, 392, 289, 230, 230, 230, 91, 218, 327, 420, 420, 128, 901, 552, 420, 230, 608, 552, 476, 347, 476, 231, 159, 137, 716, 648, 716, 627, 740, 718, 679, 679, 6, 718, 740, 6, 189, 679, 125, 159, 757, 1191, 409, 175, 250, 409, 67, 324, 681, 605, 550, 398,
550, 931, 478, 174, 21, 316, 91, 316, 654, 409, 425, 425, 699, 61, 699, 321, 698, 321, 698, 61, 425, 699, 321, 409, 699, 299, 335, 321, 335, 61, 698, 699, 654, 698, 299, 425, 231, 14, 121, 515, 121, 14, 165, 81, 409, 189, 81, 373, 465, 463, 1055, 507, 81, 81, 189, 1246, 321, 409, 886, 104, 842, 689,
300, 740, 380, 656, 656, 832, 656, 380, 300, 300, 206, 187, 175, 142, 465, 206, 271, 468, 215, 560, 83, 215, 83, 215, 215, 83, 175, 215, 83, 83, 111, 206, 756, 559, 756, 1367, 206, 559, 1015, 559, 559, 946, 1015, 548, 559, 756, 1043, 756, 698, 159, 414, 308, 458, 997, 663, 663, 347, 39, 755, 838,
323, 755, 323, 159, 159, 717, 159, 21, 41, 128, 516, 159, 717, 71, 870, 755, 159, 740, 717, 374, 516, 740, 51, 148, 335, 148, 335, 791, 120, 364, 335, 335, 51, 120, 251, 538, 251, 971, 1395, 538, 78, 178, 538, 538, 918, 129, 918, 129, 538, 538, 656, 129, 538, 538, 129, 538, 1051, 538, 128, 838, 931,
998, 823, 1095, 334, 870, 334, 367, 550, 1061, 498, 745, 832, 498, 745, 716, 498, 498, 128, 997, 832, 716, 832, 130, 642, 616, 497, 432, 432, 432, 432, 642, 159, 432, 46, 230, 788, 160, 230, 478, 46, 693, 103, 920, 230, 589, 643, 160, 616, 432, 165, 165, 583, 592, 838, 784, 583, 710, 6, 583, 583, 6,
35, 230, 838, 592, 710, 6, 589, 230, 838, 30, 592, 583, 6, 583, 6, 6, 583, 30, 30, 6, 375, 375, 99, 36, 1158, 425, 662, 417, 681, 364, 375, 1025, 538, 822, 669, 893, 538, 538, 450, 409, 632, 527, 632, 563, 632, 527, 550, 71, 698, 550, 39, 550, 514, 537, 514, 537, 111, 41, 173, 592, 173, 648, 173, 173,
173, 1011, 514, 173, 173, 514, 166, 648, 355, 161, 166, 648, 497, 327, 327, 550, 650, 21, 425, 605, 555, 103, 425, 605, 842, 836, 1011, 636, 138, 756, 836, 756, 756, 353, 1011, 636, 636, 1158, 741, 741, 842, 756, 741, 1011, 677, 1011, 770, 366, 306, 488, 920, 920, 665, 775, 502, 500, 775, 775, 648,
364, 833, 207, 13, 93, 500, 364, 500, 665, 500, 93, 295, 183, 1293, 313, 272, 313, 279, 303, 93, 516, 93, 1013, 381, 6, 93, 93, 303, 259, 643, 168, 673, 230, 1261, 230, 230, 673, 1060, 1079, 1079, 550, 741, 741, 590, 527, 741, 741, 442, 741, 442, 848, 741, 590, 925, 219, 527, 925, 335, 442, 590, 239,
590, 590, 590, 239, 527, 239, 1033, 230, 734, 241, 741, 230, 549, 548, 1015, 1015, 32, 36, 433, 465, 724, 465, 73, 73, 73, 465, 808, 73, 592, 1430, 250, 154, 154, 250, 538, 353, 353, 353, 353, 353, 175, 194, 206, 538, 632, 1163, 960, 175, 175, 538, 452, 632, 1163, 175, 538, 960, 194, 175, 194, 632,
960, 632, 94, 632, 461, 960, 1163, 1163, 461, 632, 960, 755, 707, 105, 382, 625, 382, 382, 784, 707, 871, 559, 387, 387, 871, 784, 559, 784, 88, 36, 570, 314, 1028, 975, 335, 335, 398, 573, 573, 573, 21, 215, 562, 738, 612, 424, 21, 103, 788, 870, 912, 23, 186, 757, 73, 818, 23, 73, 563, 952, 262,
563, 137, 262, 1022, 952, 137, 1273, 442, 952, 604, 137, 308, 384, 913, 235, 325, 695, 398, 95, 668, 776, 713, 309, 691, 22, 10, 364, 682, 682, 578, 481, 1252, 1072, 1252, 825, 578, 825, 1072, 1149, 592, 273, 387, 273, 427, 155, 1204, 50, 452, 50, 1142, 50, 367, 452, 1142, 611, 367, 50, 50, 367, 50,
1675, 99, 367, 50, 1501, 1099, 830, 681, 689, 917, 1089, 453, 425, 235, 918, 538, 550, 335, 161, 387, 859, 324, 21, 838, 859, 1123, 21, 723, 21, 335, 335, 206, 21, 364, 1426, 21, 838, 838, 335, 364, 21, 21, 859, 920, 838, 838, 397, 81, 639, 397, 397, 588, 933, 933, 784, 222, 830, 36, 36, 222, 1251,
266, 36, 146, 266, 366, 581, 605, 366, 22, 966, 681, 681, 433, 730, 1013, 550, 21, 21, 938, 488, 516, 21, 21, 656, 420, 323, 323, 323, 327, 323, 918, 581, 581, 830, 361, 830, 364, 259, 364, 496, 496, 364, 691, 705, 691, 475, 427, 1145, 600, 179, 427, 527, 749, 869, 689, 335, 347, 220, 298, 689, 1426,
183, 554, 55, 832, 550, 550, 165, 770, 957, 67, 1386, 219, 683, 683, 355, 683, 355, 355, 738, 355, 842, 931, 266, 325, 349, 256, 1113, 256, 423, 960, 554, 554, 325, 554, 508, 22, 142, 22, 508, 916, 767, 55, 1529, 767, 55, 1286, 93, 972, 550, 931, 1286, 1286, 972, 93, 1286, 1392, 890, 93, 1286, 93,
1286, 972, 374, 931, 890, 808, 779, 975, 975, 175, 173, 4, 681, 383, 1367, 173, 383, 1367, 383, 173, 175, 69, 238, 146, 238, 36, 148, 888, 238, 173, 238, 148, 238, 888, 185, 925, 925, 797, 925, 815, 925, 469, 784, 289, 784, 925, 797, 925, 925, 1093, 925, 925, 925, 1163, 797, 797, 815, 925, 1093,
784, 636, 663, 925, 187, 922, 316, 1380, 709, 916, 916, 187, 355, 948, 916, 187, 916, 916, 948, 948, 916, 355, 316, 316, 334, 300, 1461, 36, 583, 1179, 699, 235, 858, 583, 699, 858, 699, 1189, 1256, 1189, 699, 797, 699, 699, 699, 699, 427, 488, 427, 488, 175, 815, 656, 656, 150, 322, 465, 322, 870,
465, 1099, 582, 665, 767, 749, 635, 749, 600, 1448, 36, 502, 235, 502, 355, 502, 355, 355, 355, 172, 355, 355, 95, 866, 425, 393, 1165, 42, 42, 42, 393, 939, 909, 909, 836, 552, 424, 1333, 852, 897, 1426, 1333, 1446, 1426, 997, 1011, 852, 1198, 55, 32, 239, 588, 681, 681, 239, 1401, 32, 588, 239,
462, 286, 1260, 984, 1160, 960, 960, 486, 828, 462, 960, 1199, 581, 850, 663, 581, 751, 581, 581, 1571, 252, 252, 1283, 264, 430, 264, 430, 430, 842, 252, 745, 21, 307, 681, 1592, 488, 857, 857, 1161, 857, 857, 857, 138, 374, 374, 1196, 374, 1903, 1782, 1626, 414, 112, 1477, 1040, 356, 775, 414,
414, 112, 356, 775, 435, 338, 1066, 689, 689, 1501, 689, 1249, 205, 689, 765, 220, 308, 917, 308, 308, 220, 327, 387, 838, 917, 917, 917, 220, 662, 308, 220, 387, 387, 220, 220, 308, 308, 308, 387, 1009, 1745, 822, 279, 554, 1129, 543, 383, 870, 1425, 241, 870, 241, 383, 716, 592, 21, 21, 592, 425,
550, 550, 550, 427, 230, 57, 483, 784, 860, 57, 308, 57, 486, 870, 447, 486, 433, 433, 870, 433, 997, 486, 443, 433, 433, 997, 486, 1292, 47, 708, 81, 895, 394, 81, 935, 81, 81, 81, 374, 986, 916, 1103, 1095, 465, 495, 916, 667, 1745, 518, 220, 1338, 220, 734, 1294, 741, 166, 828, 741, 741, 1165,
1371, 1371, 471, 1371, 647, 1142, 1878, 1878, 1371, 1371, 822, 66, 327, 158, 427, 427, 465, 465, 676, 676, 30, 30, 676, 676, 893, 1592, 93, 455, 308, 582, 695, 582, 629, 582, 85, 1179, 85, 85, 1592, 1179, 280, 1027, 681, 398, 1027, 398, 295, 784, 740, 509, 425, 968, 509, 46, 833, 842, 401, 184,
401, 464, 6, 1501, 1501, 550, 538, 883, 538, 883, 883, 1129, 550, 550, 333, 689, 948, 21, 21, 241, 2557, 2094, 273, 308, 58, 863, 893, 1086, 409, 136, 1086, 592, 592, 830, 830, 883, 830, 277, 68, 689, 902, 277, 453, 507, 129, 689, 630, 664, 550, 128, 1626, 1626, 128, 902, 312, 589, 755, 755,
589, 755, 407, 1782, 589, 784, 1516, 1118, 407, 407, 1447, 589, 235, 755, 1191, 235, 235, 407, 128, 589, 1118, 21, 383, 1331, 691, 481, 383, 1129, 1129, 1261, 1104, 1378, 1129, 784, 1129, 1261, 1129, 947, 1129, 784, 784, 1129, 1129, 35, 1104, 35, 866, 1129, 1129, 64, 481, 730, 1260, 481, 970,
481, 481, 481, 481, 863, 481, 681, 699, 863, 486, 681, 481, 481, 55, 55, 235, 1364, 944, 632, 822, 401, 822, 952, 822, 822, 99, 550, 2240, 550, 70, 891, 860, 860, 550, 550, 916, 1176, 1530, 425, 1530, 916, 628, 1583, 916, 628, 916, 916, 628, 628, 425, 916, 1062, 1265, 916, 916, 916, 280, 461, 916,
916, 1583, 628, 1062, 916, 916, 677, 1297, 924, 1260, 83, 1260, 482, 433, 234, 462, 323, 1656, 997, 323, 323, 931, 838, 931, 1933, 1391, 367, 323, 931, 1391, 1391, 103, 1116, 1116, 1116, 769, 1195, 1218, 312, 791, 312, 741, 791, 997, 312, 334, 334, 312, 287, 287, 633, 1397, 1426, 605, 1431, 327,
592, 705, 1194, 592, 1097, 1118, 1503, 1267, 1267, 1267, 618, 1229, 734, 1089, 785, 1089, 1129, 1148, 1148, 1089, 915, 1148, 1129, 1148, 1011, 1011, 1229, 871, 1560, 1560, 1560, 563, 1537, 1009, 1560, 632, 985, 592, 1308, 592, 882, 145, 145, 397, 837, 383, 592, 592, 832, 36, 2714, 2107, 1588,
1347, 36, 36, 1443, 1453, 334, 2230, 1588, 1169, 650, 1169, 2107, 425, 425, 891, 891, 425, 2532, 679, 274, 274, 274, 325, 274, 1297, 194, 1297, 627, 314, 917, 314, 314, 1501, 414, 1490, 1036, 592, 1036, 1025, 901, 1218, 1025, 901, 280, 592, 592, 901, 1461, 159, 159, 159, 2076, 1066, 1176, 1176,
516, 327, 516, 1179, 1176, 899, 1176, 1176, 323, 1187, 1229, 663, 1229, 504, 1229, 916, 1229, 916, 1661, 41, 36, 278, 1027, 648, 648, 648, 1626, 648, 646, 1179, 1580, 1061, 1514, 1008, 1741, 2076, 1514, 1008, 952, 1089, 427, 952, 427, 1083, 425, 427, 1089, 1083, 425, 427, 425, 230, 920, 1678,
920, 1678, 189, 189, 953, 189, 133, 189, 1075, 189, 189, 133, 1264, 725, 189, 1629, 189, 808, 230, 230, 2179, 770, 230, 770, 230, 21, 21, 784, 1118, 230, 230, 230, 770, 1118, 986, 808, 916, 30, 327, 918, 679, 414, 916, 1165, 1355, 916, 755, 733, 433, 1490, 433, 433, 433, 605, 433, 433, 433, 1446,
679, 206, 433, 21, 2452, 206, 206, 433, 1894, 206, 822, 206, 2073, 206, 206, 21, 822, 21, 206, 206, 21, 383, 1513, 375, 1347, 432, 1589, 172, 954, 242, 1256, 1256, 1248, 1256, 1256, 1248, 1248, 1256, 842, 13, 592, 13, 842, 1291, 592, 21, 175, 13, 592, 13, 13, 1426, 13, 1541, 445, 808, 808, 863,
647, 219, 1592, 1029, 1225, 917, 1963, 1129, 555, 1313, 550, 660, 550, 220, 660, 552, 663, 220, 533, 220, 383, 550, 1278, 1495, 636, 842, 1036, 425, 842, 425, 1537, 1278, 842, 554, 1508, 636, 554, 301, 842, 792, 1392, 1021, 284, 1172, 997, 1021, 103, 1316, 308, 1210, 848, 848, 1089, 1089, 848,
848, 67, 1029, 827, 1029, 2078, 827, 1312, 1029, 827, 590, 872, 1312, 427, 67, 67, 67, 67, 872, 827, 872, 2126, 1436, 26, 2126, 67, 1072, 2126, 1610, 872, 1620, 883, 883, 1397, 1189, 555, 555, 563, 1189, 555, 640, 555, 640, 1089, 1089, 610, 610, 1585, 610, 1355, 610, 1015, 616, 925, 1015, 482,
230, 707, 231, 888, 1355, 589, 1379, 151, 931, 1486, 1486, 393, 235, 960, 590, 235, 960, 422, 142, 285, 285, 327, 327, 442, 2009, 822, 445, 822, 567, 888, 2611, 1537, 323, 55, 1537, 323, 888, 2611, 323, 1537, 323, 58, 445, 593, 2045, 593, 58, 47, 770, 842, 47, 47, 842, 842, 648, 2557, 173, 689,
2291, 1446, 2085, 2557, 2557, 2291, 1780, 1535, 2291, 2391, 808, 691, 1295, 1165, 983, 948, 2000, 948, 983, 983, 2225, 2000, 983, 983, 705, 948, 2000, 1795, 1592, 478, 592, 1795, 1795, 663, 478, 1790, 478, 592, 1592, 173, 901, 312, 4, 1606, 173, 838, 754, 754, 128, 550, 1166, 551, 1480, 550,
550, 1875, 1957, 1166, 902, 1875, 550, 550, 551, 2632, 551, 1875, 1875, 551, 2891, 2159, 2632, 3231, 551, 815, 150, 1654, 1059, 1059, 734, 770, 555, 1592, 555, 2059, 770, 770, 1803, 627, 627, 627, 2059, 931, 1272, 427, 1606, 1272, 1606, 1187, 1204, 397, 822, 21, 1645, 263, 263, 822, 263, 1645,
280, 263, 605, 1645, 2014, 21, 21, 1029, 263, 1916, 2291, 397, 397, 496, 270, 270, 1319, 264, 1638, 264, 986, 1278, 1397, 1278, 1191, 409, 1191, 740, 1191, 754, 754, 387, 63, 948, 666, 666, 1198, 548, 63, 1248, 285, 1248, 169, 1248, 1248, 285, 918, 224, 285, 1426, 1671, 514, 514, 717, 514, 51,
1521, 1745, 51, 605, 1191, 51, 128, 1191, 51, 51, 1521, 267, 513, 952, 966, 1671, 897, 51, 71, 592, 986, 986, 1121, 592, 280, 2000, 2000, 1165, 1165, 1165, 1818, 222, 1818, 1165, 1252, 506, 327, 443, 432, 1291, 1291, 2755, 1413, 520, 1318, 227, 1047, 828, 520, 347, 1364, 136, 136, 452, 457, 457,
132, 457, 488, 1087, 1013, 2225, 32, 1571, 2009, 483, 67, 483, 740, 740, 1013, 2854, 866, 32, 2861, 866, 887, 32, 2444, 740, 32, 32, 866, 2225, 866, 32, 1571, 2627, 32, 850, 1675, 569, 1158, 32, 1158, 1797, 2641, 1565, 1158, 569, 1797, 1158, 1797, 55, 1703, 42, 55, 2562, 675, 1703, 42, 55, 749,
488, 488, 347, 1206, 1286, 1286, 488, 488, 1206, 1286, 1206, 1286, 550, 550, 1790, 860, 550, 2452, 550, 550, 2765, 1089, 1633, 797, 2244, 1313, 194, 2129, 194, 194, 194, 818, 32, 194, 450, 1313, 2387, 194, 1227, 2387, 308, 2232, 526, 476, 278, 830, 830, 194, 830, 194, 278, 194, 714, 476, 830,
714, 830, 278, 830, 2532, 1218, 1759, 1446, 960, 1747, 187, 1446, 1759, 960, 105, 1446, 1446, 1271, 1446, 960, 960, 1218, 1446, 1446, 105, 1446, 960, 488, 1446, 427, 534, 842, 1969, 2460, 1969, 842, 842, 1969, 427, 941, 2160, 427, 230, 938, 2075, 1675, 1675, 895, 1675, 34, 129, 1811, 239, 749,
1957, 2271, 749, 1908, 129, 239, 239, 129, 129, 2271, 2426, 1355, 1756, 194, 1583, 194, 194, 1583, 194, 1355, 194, 1628, 2221, 1269, 2425, 1756, 1355, 1355, 1583, 1033, 427, 582, 30, 582, 582, 935, 1444, 1962, 915, 733, 915, 938, 1962, 767, 353, 1630, 1962, 1962, 563, 733, 563, 733, 353, 822,
1630, 740, 2076, 2076, 2076, 589, 589, 2636, 866, 589, 947, 1528, 125, 273, 1058, 1058, 1161, 1635, 1355, 1161, 1161, 1355, 1355, 650, 1206, 1206, 784, 784, 784, 784, 784, 412, 461, 412, 2240, 412, 679, 891, 461, 679, 679, 189, 189, 1933, 1651, 2515, 189, 1386, 538, 1386, 1386, 1187, 1386,
2423, 2601, 2285, 175, 175, 2331, 194, 3079, 384, 538, 2365, 2294, 538, 2166, 1841, 3326, 1256, 3923, 976, 85, 550, 550, 1295, 863, 863, 550, 1249, 550, 1759, 146, 1069, 920, 2633, 885, 885, 1514, 1489, 166, 1514, 2041, 885, 2456, 885, 2041, 1081, 1948, 362, 550, 94, 324, 2308, 94, 2386, 94,
550, 874, 1329, 1759, 2280, 1487, 493, 493, 2099, 2599, 1431, 1086, 1514, 1086, 2099, 1858, 368, 1330, 2599, 1858, 2846, 2846, 2907, 2846, 713, 713, 1854, 1123, 713, 713, 3010, 1123, 3010, 538, 713, 1123, 447, 822, 555, 2011, 493, 508, 2292, 555, 1736, 2135, 2704, 555, 2814, 555, 2000, 555,
555, 822, 914, 327, 679, 327, 648, 537, 2263, 931, 1496, 537, 1296, 1745, 1592, 1658, 1795, 650, 1592, 1745, 1745, 1658, 1592, 1745, 1592, 1745, 1658, 1338, 2124, 1592, 1745, 1745, 1745, 837, 1726, 2897, 1118, 1118, 230, 1118, 1118, 1118, 1388, 1748, 514, 128, 1165, 931, 514, 2974, 2041,
2387, 2041, 979, 185, 36, 1269, 550, 173, 812, 36, 1165, 2676, 2562, 1473, 2885, 1982, 1578, 1578, 383, 383, 2360, 383, 1578, 2360, 1584, 1982, 1578, 1578, 1578, 2019, 1036, 355, 724, 2023, 205, 303, 355, 1036, 1966, 355, 1036, 401, 401, 401, 830, 401, 849, 578, 401, 849, 849, 578, 1776, 1123,
552, 2632, 808, 1446, 1120, 373, 1529, 1483, 1057, 893, 1284, 1430, 1529, 1529, 2632, 1352, 2063, 1606, 1352, 1606, 2291, 3079, 2291, 1529, 506, 838, 1606, 1606, 1352, 1529, 1529, 1483, 1529, 1606, 1529, 259, 902, 259, 902, 612, 612, 284, 398, 2991, 1534, 1118, 1118, 1118, 1118, 1118, 734,
284, 2224, 398, 734, 284, 734, 398, 3031, 398, 734, 1707, 2643, 1344, 1477, 475, 1818, 194, 1894, 691, 1528, 1184, 1207, 1501, 6, 2069, 871, 2069, 3548, 1443, 2069, 2685, 3265, 1350, 3265, 2069, 2069, 128, 1313, 128, 663, 414, 1313, 414, 2000, 128, 2000, 663, 1313, 699, 1797, 550, 327, 550,
1526, 699, 327, 1797, 1526, 550, 550, 327, 550, 1426, 1426, 1426, 2285, 1123, 890, 728, 1707, 728, 728, 327, 253, 1187, 1281, 1364, 1571, 2170, 755, 3232, 925, 1496, 2170, 2170, 1125, 443, 902, 902, 925, 755, 2078, 2457, 902, 2059, 2170, 1643, 1129, 902, 902, 1643, 1129, 606, 36, 103, 338,
338, 1089, 338, 338, 338, 1089, 338, 36, 340, 1206, 1176, 2041, 833, 1854, 1916, 1916, 1501, 2132, 1736, 3065, 367, 1934, 833, 833, 833, 2041, 3017, 2147, 818, 1397, 828, 2147, 398, 828, 818, 1158, 818, 689, 327, 36, 1745, 2132, 582, 1475, 189, 582, 2132, 1191, 582, 2132, 1176, 1176, 516,
2610, 2230, 2230, 64, 1501, 537, 1501, 173, 2230, 2988, 1501, 2694, 2694, 537, 537, 173, 173, 1501, 537, 64, 173, 173, 64, 2230, 537, 2230, 537, 2230, 2069, 3142, 1645, 689, 1165, 1165, 1963, 514, 488, 1963, 1145, 235, 1145, 1078, 1145, 231, 2405, 552, 21, 57, 57, 57, 1297, 1455, 1988,
2310, 1885, 2854, 2014, 734, 1705, 734, 2854, 734, 677, 1988, 1660, 734, 677, 734, 677, 677, 734, 2854, 1355, 677, 1397, 2947, 2386, 1698, 128, 1698, 3028, 2386, 2437, 2947, 2386, 2643, 2386, 2804, 1188, 335, 746, 1187, 1187, 861, 2519, 1917, 2842, 1917, 675, 1308, 234, 1917, 314, 314, 2339,
2339, 2592, 2576, 902, 916, 2339, 916, 2339, 916, 2339, 916, 1089, 1089, 2644, 1221, 1221, 2446, 308, 308, 2225, 2225, 3192, 2225, 555, 1592, 1592, 555, 893, 555, 550, 770, 3622, 2291, 2291, 3419, 465, 250, 2842, 2291, 2291, 2291, 935, 160, 1271, 308, 325, 935, 1799, 1799, 1891, 2227, 1799,
1598, 112, 1415, 1840, 2014, 1822, 2014, 677, 1822, 1415, 1415, 1822, 2014, 2386, 2159, 1822, 1415, 1822, 179, 1976, 1033, 179, 1840, 2014, 1415, 1970, 1970, 1501, 563, 563, 563, 462, 563, 1970, 1158, 563, 563, 1541, 1238, 383, 235, 1158, 383, 1278, 383, 1898, 2938, 21, 2938, 1313, 2201,
2059, 423, 2059, 1313, 872, 1313, 2044, 89, 173, 3327, 1660, 2044, 1623, 173, 1114, 1114, 1592, 1868, 1651, 1811, 383, 3469, 1811, 1651, 869, 383, 383, 1651, 1651, 3223, 2166, 3469, 767, 383, 1811, 767, 2323, 3355, 1457, 3341, 2640, 2976, 2323, 3341, 2323, 2640, 103, 103, 1161, 1080, 2429,
370, 2018, 2854, 2429, 2166, 2429, 2094, 2207, 871, 1963, 1963, 2023, 2023, 2336, 663, 2893, 1580, 691, 663, 705, 2046, 2599, 409, 2295, 1118, 2494, 1118, 1950, 549, 2494, 2453, 2046, 2494, 2453, 2046, 2453, 2046, 409, 1118, 4952, 2291, 2225, 1894, 1423, 2498, 567, 4129, 1475, 1501, 795,
463, 2084, 828, 828, 232, 828, 232, 232, 1818, 1818, 666, 463, 232, 220, 220, 2162, 2162, 833, 4336, 913, 35, 913, 21, 2927, 886, 3037, 383, 886, 876, 1747, 383, 916, 916, 916, 2927, 916, 1747, 837, 1894, 717, 423, 481, 1894, 1059, 2262, 3206, 4700, 1059, 3304, 2262, 871, 1831, 871, 3304, 1059,
1158, 1934, 1158, 756, 1511, 41, 978, 1934, 2603, 720, 41, 756, 41, 325, 2611, 1158, 173, 1123, 1934, 1934, 1511, 2045, 2045, 2045, 1423, 3206, 3691, 2512, 3206, 2512, 2000, 1811, 2504, 2504, 2611, 2437, 2437, 2437, 1455, 893, 150, 2665, 1966, 605, 398, 2331, 1177, 516, 1962, 4241, 94, 1252,
760, 1292, 1962, 1373, 2000, 1990, 3684, 42, 1868, 3779, 1811, 1811, 2041, 3010, 5436, 1780, 2041, 1868, 1811, 1780, 1811, 1868, 1811, 2041, 1868, 1811, 5627, 4274, 1811, 1868, 4602, 1811, 1811, 1474, 2665, 235, 1474, 2665

# B.7 Random Numbers

The two tables $V_0$ and $V_1$ described in Section B.5.4.1 are given below. Each entry is a 32-bit integer in decimal representation.

## B.7.1 The table $V_0$

251291136, 3952231631, 3370958628, 4070167936, 123631495, 3351110283, 3218676425, 2011642291, 774603218, 2402805061, 1004366930, 1843948209, 428891132, 3746331984, 1591258008, 3067016507, 1433388735, 504005498, 2032657933, 3419319784,
2805686246, 3102436986, 3808671154, 2501582075, 3978944421, 246043949, 4016898363, 649743608, 1974987508, 2651273766, 2357956801, 689605112, 715807172, 2722736134, 191939188, 3535520147, 3277019569, 1470435941, 3763101702, 3232409631,
122701163, 3920852693, 782246947, 372121310, 2995604341, 2045698575, 2332962102, 4005368743, 218596347, 3415381967, 4207612806, 861117671, 3676575285, 2581671944, 3312220480, 681232419, 307306866, 4112503940, 1158111502, 709227802,
2724140433, 4201101115, 4215970289, 4048876515, 3031661061, 1909085522, 510985033, 1361682810, 129243379, 3142379587, 2569842483, 3033268270, 1658118006, 932109358, 1982290045, 2983082771, 3007670818, 3448104768, 683749698, 778296777,
1399125101, 1939403708, 1692176003, 3868299200, 1422476658, 593093658, 1878973865, 2526292949, 1591602827, 3986158854, 3964389521, 2695031039, 1942050155, 424618399, 1347204291, 2669179716, 2434425874, 2540801947, 1384069776, 4123580443,
1523670218, 2708475297, 1046771089, 2229796016, 1255426612, 4213663089, 1521339547, 3041843489, 420130494, 10677091, 515623176, 3457502702, 2115821274, 2720124766, 3242576090, 854310108, 425973987, 325832382, 1796851292, 2462744411,
1976681690, 1408671665, 1228817808, 3917210003, 263976645, 2593736473, 2471651269, 4291353919, 650792940, 1191583883, 3046561335, 2466530435, 2545983082, 969168436, 2019348792, 2268075521, 1169345068, 3250240009, 3963499681, 2560755113,
911182396, 760842409, 3569308693, 2687243553, 381854665, 2613828404, 2761078866, 1456668111, 883760091, 3294951678, 1604598575, 1985308198, 1014570543, 2724959607, 3062518035, 3115293053, 138853680, 4160398285, 3322241130, 2068983570,
2247491078, 3669524410, 1575146607, 828029864, 3732001371, 3422026452, 3370954177, 4006626915, 543812220, 1243116171, 3928372514, 2791443445, 4081325272, 2280435605, 885616073, 616452097, 3188863436, 2780382310, 2340014831, 1208439576,
258356309, 3837963200, 2075009450, 3214181212, 3303882142, 880813252, 1355575717, 207231484, 2420803184, 358923368, 1617557768, 3272161958, 1771154147, 2842106362, 1751209208, 1421030790, 658316681, 194065839, 3241510581, 38625260,
301875395, 4176141739, 297312930, 2137802113, 1502984205, 3669376622, 3728477036, 234652930, 2213589897, 2734638932, 1129721478, 3187422815, 2859178611, 3284308411, 3819792700, 3557526733, 451874476, 1740576081, 3592838701, 1709429513,
3702918379, 3533351328, 1641660745, 179350258, 2380520112, 3936163904, 3685256204, 3156252216, 1854258901, 2861641019,\ 3176611298, 834787554, 331353807, 517858103, 3010168884, 4012642001, 2217188075, 3756943137, 3077882590, 2054995199,
3081443129, 3895398812, 1141097543, 2376261053, 2626898255, 2554703076, 401233789, 1460049922, 678083952, 1064990737, 940909784, 1673396780, 528881783, 1712547446, 3629685652, 1358307511

## B.7.2 The table $V_1$

807385413, 2043073223, 3336749796, 1302105833, 2278607931, 541015020, 1684564270, 372709334, 3508252125, 1768346005, 1270451292, 2603029534, 2049387273, 3891424859, 2152948345, 4114760273, 915180310, 3754787998, 700503826, 2131559305,
1308908630, 224437350, 4065424007, 3638665944, 1679385496, 3431345226, 1779595665, 3068494238, 1424062773, 1033448464, 4050396853, 3302235057, 420600373, 2868446243, 311689386, 259047959, 4057180909, 1575367248, 4151214153, 110249784,
3006865921, 4293710613, 3501256572, 998007483, 499288295, 1205710710, 2997199489, 640417429, 3044194711, 486690751, 2686640734, 2394526209, 2521660077, 49993987, 3843885867, 4201106668, 415906198, 19296841, 2402488407, 2137119134,
1744097284, 579965637, 2037662632, 852173610, 2681403713, 1047144830, 2982173936, 910285038, 4187576520, 2589870048, 989448887, 3292758024, 506322719, 176010738, 1865471968, 2619324712, 564829442, 1996870325, 339697593, 4071072948,
3618966336, 2111320126, 1093955153, 957978696, 892010560, 1854601078, 1873407527, 2498544695, 2694156259, 1927339682, 1650555729, 183933047, 3061444337, 2067387204, 228962564, 3904109414, 1595995433, 1780701372, 2463145963, 307281463,
3237929991, 3852995239, 2398693510, 3754138664, 522074127, 146352474, 4104915256, 3029415884, 3545667983, 332038910, 976628269, 3123492423, 3041418372, 2258059298, 2139377204, 3243642973, 3226247917, 3674004636, 2698992189, 3453843574,
1963216666, 3509855005, 2358481858, 747331248, 1957348676, 1097574450, 2435697214, 3870972145, 1888833893, 2914085525, 4161315584, 1273113343, 3269644828, 3681293816, 412536684, 1156034077, 3823026442, 1066971017, 3598330293, 1979273937,
2079029895, 1195045909, 1071986421, 2712821515, 3377754595, 2184151095, 750918864, 2585729879, 4249895712, 1832579367, 1192240192, 946734366, 31230688, 3174399083, 3549375728, 1642430184, 1904857554, 861877404, 3277825584, 4267074718,
3122860549, 666423581, 644189126, 226475395, 307789415, 1196105631, 3191691839, 782852669, 1608507813, 1847685900, 4069766876, 3931548641, 2526471011, 766865139, 2115084288, 4259411376, 3323683436, 568512177, 3736601419, 1800276898,
4012458395, 1823982, 27980198, 2023839966, 869505096, 431161506, 1024804023, 1853869307, 3393537983, 1500703614, 3019471560, 1351086955, 3096933631, 3034634988, 2544598006, 1230942551, 3362230798, 159984793, 491590373, 3993872886,
3681855622, 903593547, 3535062472, 1799803217, 772984149, 895863112, 1899036275, 4187322100, 101856048, 234650315, 3183125617, 3190039692, 525584357, 1286834489, 455810374, 1869181575, 922673938, 3877430102, 3422391938, 1414347295,
1971054608, 3061798054, 830555096, 2822905141, 167033190, 1079139428, 4210126723, 3593797804, 429192890, 372093950, 1779187770, 3312189287, 204349348, 452421568, 2800540462, 3733109044, 1235082423, 1765319556, 3174729780, 3762994475,
3171962488, 442160826, 198349622, 45942637, 1324086311, 2901868599, 678860040, 3812229107, 19936821, 1119590141, 3640121682, 3545931032, 2102949142, 2828208598, 3603378023, 4135048896

# B.8 Example FEC decoder

## B.8.1 General

This section describes an efficient decoding algorithm for the Raptor codes described in this specification. Note that each received encoding symbol can be considered as the value of an equation amongst the intermediate symbols. From these simultaneous equations, and the known pre-coding relationships amongst the intermediate symbols, any algorithm for solving simultaneous equations can successfully decode the intermediate symbols and hence the source symbols. However, the algorithm chosen has a major effect on the computational efficiency of the decoding.

## B.8.2 Decoding a source block

### B.8.2.1 General

It is assumed that the decoder knows the structure of the source block it is to decode, including the symbol size, $T$, and the number $K$ of symbols in the source block.

From the algorithms described in Sections B.5, the Raptor decoder can calculate the total number $L = K+S+H$ of pre-coding symbols and determine how they were generated from the source block to be decoded. In this description it is assumed that the received encoding symbols for the source block to be decoded are passed to the decoder. Furthermore, for each such encoding symbol it is assumed that the number and set of intermediate symbols whose exclusive-or is equal to the encoding symbol is passed to the decoder. In the case of source symbols, the source symbol triples described in Section B.5.2.2 indicate the number and set of intermediate symbols which sum to give each source symbol.

Let $N \geq K$ be the number of received encoding symbols for a source block and let $M = S+H+N$. The following $M$ by $L$ bit matrix **A** can be derived from the information passed to the decoder for the source block to be decoded. Let **C** be the column vector of the $L$ intermediate symbols, and let **D** be the column vector of $M$ symbols with values known to the receiver, where the first $S+H$ of the $M$ symbols are zero-valued symbols that correspond to LDPC and Half symbols (these are check symbols for the LDPC and Half symbols, and not the LDPC and Half symbols themselves), and the remaining $N$ of the $M$ symbols are the received encoding symbols for the source block. Then, **A** is the bit matrix that satisfies **A**·**C** = **D**, where here · denotes matrix multiplication over GF[2]. In particular, *A*[*i*,*j*] = 1 if the intermediate symbol corresponding to index *j* is exclusive-ORed into the LDPC, Half or encoding symbol corresponding to index *i* in the encoding, or if index *i* corresponds to a LDPC or Half symbol and index *j* corresponds to the same LDPC or Half symbol. For all other *i* and *j*, *A*[*i*,*j*] = 0.

Decoding a source block is equivalent to decoding **C** from known **A** and **D**. It is clear that **C** can be decoded if and only if the rank of **A** over GF[2] is *L*. Once **C** has been decoded, missing source symbols can be obtained by using the source symbol triples to determine the number and set of intermediate symbols which must be exclusive-ORed to obtain each missing source symbol.

The first step in decoding **C** is to form a decoding schedule. In this step **A** is converted, using Gaussian elimination (using row operations and row and column reorderings) and after discarding *M* – *L* rows, into the *L* by *L* identity matrix. The decoding schedule consists of the sequence of row operations and row and column re-orderings during the Gaussian elimination process, and only depends on **A** and not on **D**. The decoding of **C** from **D** can take place concurrently with the forming of the decoding schedule, or the decoding can take place afterwards based on the decoding schedule.

The correspondence between the decoding schedule and the decoding of **C** is as follows. Let *c*[0] = 0, *c*[1] = 1…,*c*[*L*-1] = *L*-1 and *d*[0] = 0, *d*[1] = 1…,*d*[*M*-1] = *M*-1 initially.

- Each time row *i* of **A** is exclusive-ORed into row *i*" in the decoding schedule then in the decoding process symbol *D*[*d*[*i*]] is exclusive-ORed into symbol *D*[*d*[*i*"]] .
- Each time row *i* is exchanged with row *i*" in the decoding schedule then in the decoding process the value of *d*[*i*] is exchanged with the value of *d*[*i*"].
- Each time column *j* is exchanged with column *j*" in the decoding schedule then in the decoding process the value of *c*[*j*] is exchanged with the value of *c*[*j*"].

From this correspondence it is clear that the total number of exclusive-ORs of symbols in the decoding of the source block is the number of row operations (not exchanges) in the Gaussian elimination. Since **A** is the *L* by *L* identity matrix after the Gaussian elimination and after discarding the last *M* – *L* rows, it is clear at the end of successful decoding that the *L* symbols *D*[*d*[0]], *D*[*d*[1]],…, *D*[*d*[*L*-1]] are the values of the *L* symbols *C*[*c*[0]], *C*[*c*[1]],…, *C*[*c*[*L*-1]].

The order in which Gaussian elimination is performed to form the decoding schedule has no bearing on whether or not the decoding is successful. However, the speed of the decoding depends heavily on the order in which Gaussian elimination is performed. (Furthermore, maintaining a sparse representation of **A** is crucial, although this is not described here). The remainder of this section describes an order in which Gaussian elimination could be performed that is relatively efficient.

### B.8.2.2 First Phase

The first phase of the Gaussian elimination the matrix **A** is conceptually partitioned into submatrices. The submatrix sizes are parameterized by non-negative integers *i* and *u* which are initialized to 0. The submatrices of **A** are:

(1) The submatrix **I** defined by the intersection of the first *i* rows and first *i* columns. This is the identity matrix at the end of each step in the phase.

(2) The submatrix defined by the intersection of the first *i* rows and all but the first *i* columns and last *u* columns. All entries of this submatrix are zero.

(3) The submatrix defined by the intersection of the first *i* columns and all but the first *i* rows. All entries of this submatrix are zero.

(4) The submatrix **U** defined by the intersection of all the rows and the last *u* columns.

(5) The submatrix **V** formed by the intersection of all but the first *i* columns and the last *u* columns and all but the first *i* rows.

Figure B.2.2-1 illustrates the submatrices of **A**. At the beginning of the first phase **V** = **A**. In each step, a row of **A** is chosen.

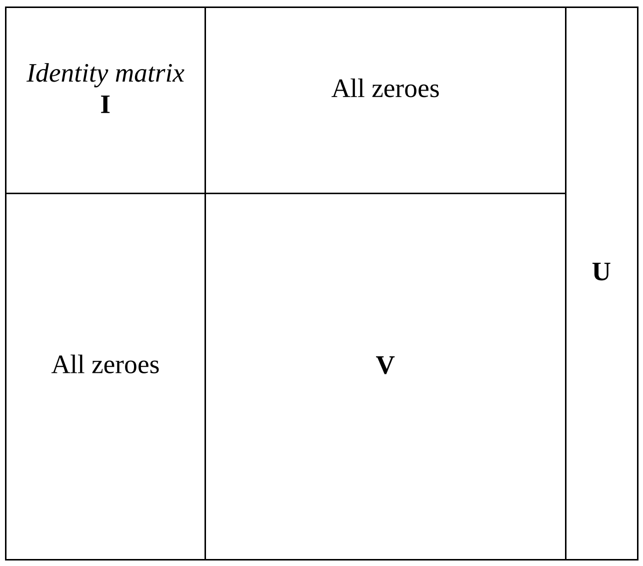


**Figure B.2.2-1 – Submatrices of A in the first phase**

The following graph defined by the structure of **V** is used in determining which row of **A** is chosen. The columns that intersect **V** are the nodes in the graph, and the rows that have exactly 2 ones in **V** are the edges of the graph that connect the two columns (nodes) in the positions of the two ones. A component in this graph is a maximal set of nodes (columns) and edges (rows) such that there is a path between each pair of nodes/edges in the graph. The size of a component is the number of nodes (columns) in the component.

There are at most $L$ steps in the first phase. The phase ends successfully when $i + u = L$, i.e., when **V** and the all zeroes submatrix above **V** have disappeared and $A$ consists of **I**, the all zeroes submatrix below **I**, and **U**. The phase ends unsuccessfully in decoding failure if at some step before **V** disappears there is no non-zero row in **V** to choose in that step. In each step, a row of **A** is chosen as follows:

- If all entries of **V** are zero then no row is chosen and decoding fails.
- Let $r$ be the minimum integer such that at least one row of **A** has exactly $r$ ones in **V**.
- If $r \neq 2$ then choose a row with exactly $r$ ones in **V** with minimum original degree among all such rows.
- If $r = 2$ then choose any row with exactly 2 ones in **V** that is part of a maximum size component in the graph defined by $X$.

After the row is chosen in this step the first row of **A** that intersects **V** is exchanged with the chosen row so that the chosen row is the first row that intersects **V**. The columns of **A** among those that intersect **V** are reordered so that one of the $r$ ones in the chosen row appears in the first column of **V** and so that the remaining $r$-1 ones appear in the last columns of **V**. Then, the chosen row is exclusive-ORed into all the other rows of **A** below the chosen row that have a one in the first column of **V**. Finally, $i$ is incremented by 1 and $u$ is incremented by $r$-1, which completes the step.

## B.8.2.3 Second Phase

The submatrix **U** is further partitioned into the first $i$ rows, $\mathbf{U}_{\text{upper}}$, and the remaining $M – i$ rows, $\mathbf{U}_{\text{lower}}$. Gaussian elimination is performed in the second phase on $\mathbf{U}_{\text{lower}}$ to either determine that its rank is less than $u$ (decoding failure) or to convert it into a matrix where the first $u$ rows is the identity matrix (success of the second phase). Call this $u$ by $u$ identity matrix $\mathbf{I}_u$. The $M – L$ rows of $A$ that intersect $\mathbf{U}_{\text{lower}} – \mathbf{I}_u$ are discarded. After this phase **A** has $L$ rows and $L$ columns.

## B.8.2.4 Third Phase

After the second phase the only portion of **A** which needs to be zeroed out to finish converting **A** into the $L$ by $L$ identity matrix is $\mathbf{U}_{\text{upper}}$. The number of rows $i$ of the submatrix $\mathbf{U}_{\text{upper}}$ is generally much larger than the number of columns $u$ of $\mathbf{U}_{\text{upper}}$. To zero out $\mathbf{U}_{\text{upper}}$ efficiently, the following precomputation matrix $\mathbf{U}''$ is computed based on $\mathbf{I}_u$ in the third phase and then **U″** is used in the fourth phase to zero out $\mathbf{U}_{\text{upper}}$. The *u* rows of $\mathbf{I}_u$ are partitioned into ceil(*u*/8) groups of 8 rows each. Then, for each group of 8 rows all non-zero combinations of the 8 rows are computed, resulting in $2^8 - 1 = 255$ rows (this can be done with $2^8$-8-1 = 247 exclusive-ors of rows per group, since the combinations of Hamming weight one that appear in $\mathbf{I}_u$ do not need to be recomputed). Thus, the resulting precomputation matrix **U″** has ceil(*u*/8) ·255 rows and *u* columns. Note that **U″** is not formally a part of matrix **A**, but will be used in the fourth phase to zero out $\mathbf{U}_{\text{upper}}$.

### B.8.2.5 Fourth Phase

For each of the first *i* rows of **A**, for each group of 8 columns in the $\mathbf{U}_{\text{upper}}$submatrix of this row, if the set of 8 column entries in $\mathbf{U}_{\text{upper}}$ are not all zero then the row of the precomputation matrix **U″** that matches the pattern in the 8 columns is exclusive-ORed into the row, thus zeroing out those 8 columns in the row at the cost of exclusive-oring one row of **U″** into the row.

After this phase **A** is the *L* by *L* identity matrix and a complete decoding schedule has been successfully formed. Then, as explained in Section C.2.1, the corresponding decoding consisting of exclusive-ORing known encoding symbols can be executed to recover the intermediate symbols based on the decoding schedule.

The triples associated with all source symbols are computed according to B.5.2.2.The triples for received source symbols are used in the decoding. The triples for missing source symbols are used to determine which intermediate symbols need to be exclusive-ORed to recover the missing source symbols.

# Annex C (informative): IANA registration

This annex provides the required IANA registration.

## C.1 Registration of SDP Protocol Identifiers for Source packet

This specification defines two new SDP protocol identificators for source packets. As the registration rules requires these to be registered by an RFC, there will be an RFC referencing the definitions here.

Protocol identifier 'UDP/MBMS-FEC/RTP/AVP' identifies a protocol combination of UDP[7], FEC source packets (see section 8.2.2.3), RTP [6] using the AVP profile [78]. This protocol identifier shall use the FMT space rules that are used for RTP/AVP.

Protocol identifier 'UDP/MBMS-FEC/RTP/SAVP' identifies a protocol combination of UDP [7], FEC source packets (see section 8.2.2.3), and RTP [6] using the SAVP profile [77]. This protocol identifier shall use the FMT space rules that are used for RTP/AVP.

## C.2 Registration of SDP Protocol identifier for repair packets

This specification defines one new SDP protocol identificator for FEC repair packets. As the registration rules requires these to be registered by an RFC, there will be an RFC referencing the definitions within this specification.

Protocol identifier 'UDP/MBMS-REPAIR' identifies a protocol combination of UDP [7], FEC repair packets (see section 8.2.2.4). The FMT string is not used and shall be set to '*'.

## C.3 Registration of MIME type "application/simpleSymbolContainer"

The MIME Type "application/simpleSymbolContainer" denotes that the message body is a simple container of encoding symbols for the file repair procedure (clause 9.3.5.2 - File Repair Response Message Format for Carriage of Repair Data).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.4 Registration of MIME type "application/mbms-user-service-description-parameter"

The MIME Type "application/mbms-user-service-description-parameter" denotes that the message body is a user service description instance (see clause 5.2.4).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.5 Registration of MIME type "application/mbms-envelope"

The MIME Type "application/mbms-envelope" denotes that the message body is a metadata envelope (see clause 5.2.4).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.6 Registration of MIME type "application/mbms-protection-description"

The MIME-Type "application/mbms-protection-description" denotes that the message body is an MBMS protection description parameter (see clause 5.2.2.3).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.7 Registration of MIME type "application/mbms-associated-procedure-parameter"

The MIME-Type "application/mbms-associated-procedure-parameter" denotes that the message body contains the associated procedure parameters (see clause 9.2).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.8 Registration of MIME type "application/vnd.3gpp.mbms-msk+xml"

The MIME-Type "application/vnd.3gpp.mbms-msk+xml" denotes that the message body contains the MSK request parameters (see 3GPP TS 33.246 [20]).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.9 Registration of MIME type "application/vnd.3gpp.mbms-register+xml"

The MIME-Type "application/vnd.3gpp.mbms-register+xml" denotes that the message body contains the MBMS User Service Registration parameters (see 3GPP TS 33.246 [20]).

NOTE: The detailed IANA registration information of this content type is tbd.

## C.10 Registration of MIME type "application/vnd.3gpp.mbms-deregister+xml"

The MIME-Type "application/vnd.3gpp.mbms-deregister+xml" denotes that the message body contains the MBMS User Service Deregistration parameters (see 3GPP TS 33.246 [20]).

NOTE: The detailed IANA registration information of this content type is tbd

# Annex D (informative): RTP packetization guidelines

This annex provides guidelines for MBMS senders to minimize initial buffering delay between starting of the reception and starting of rendering of media data in MBMS receivers: When H.264 (AVC) video is in use, an MBMS sender should form FEC source blocks in which the first H.264 (AVC) access unit in decoding order is an IDR access unit.

MBMS senders should transmit all application data units for a given H.264 (AVC) access unit, or audio frame within one FEC source block.

MBMS senders should set the min-buffer-time MIME/SDP parameter and the minimum buffering delay elements included in FEC source blocks to values that are sufficient to cover any required de-interleaving of application data units, such as H.264 (AVC) NAL units and coded audio frames, from their transmission order to decoding order.

When RTP timestamps are converted to the wallclock time of the MBMS receiver, the smallest RTP timestamp among the FEC source packets of a FEC source block of a stream should be equal or close to the smallest RTP timestamp among the FEC source packets of a FEC source block of any other stream of the same MBMS streaming session.

When RTP timestamps are converted to the wallclock time of the MBMS receiver, the greatest RTP timestamp among the FEC source packets of a FEC source block of a stream should be equal or close to the greatest RTP timestamp among the FEC source packets of a FEC source block of any other stream of the same MBMS streaming session.

# Annex E (informative): Change history

| Change history | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Date** | **TSG SA#** | **TSG Doc.** | **CR** | **Rev** | **Subject/Comment** | **Old** | **New** |
| 2005-03 | 27 | SP-050082 | | | Approved at TSG SA#27 Plenary | 2.0.0 | 6.0.0 |
| 2005-06 | 28 | SP-050250 | 001 | 1 | Corrections to QoE metrics specification for MBMS | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 002 | 1 | Using two TMGIs | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 003 | | MBMS Service Descriptions over HTTP | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 004 | 1 | Corrections to the specification of Associated Delivery Procedures for MBMS | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 005 | 2 | Usage of MBMS Session Identity | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 010 | 1 | MBMS user service announcement via point-to-point push bearers | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 011 | | Removal of obsolete note | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 013 | | Specification of Raptor Forward Error Correction and Streaming User Service bundling | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 015 | | Clarification of Associated Delivery Procedure | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 016 | | Corrections of FLUTE Support Requirements | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 017 | | Corrections of the reference list | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 018 | | Definition of RTP Session | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 019 | | Corrections and editorial modifications to chapter 4 | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 020 | | MBMS Repair | 6.0.0 | 6.1.0 |
| 2005-06 | 28 | SP-050250 | 021 | | MBMS Media Codec Support | 6.0.0 | 6.1.0 |
| | | | | | | | |

# History

| Document history | | |
|---|---|---|
| V6.0.0 | March 2005 | Publication |
| V6.1.0 | June 2005 | Publication |
| | | |
| | | |
| | | |