

3GPP2 C.S0070-0
Version 1.0
January 2011



3RD GENERATION
PARTNERSHIP
PROJECT 2
"3GPP2"

BCMCS Codecs and Transport Protocols

© 2011 3GPP2

3GPP2 and its Organizational Partners claim copyright in this document and individual Organizational Partners may copyright and issue documents or standards publications in individual Organizational Partner's name based on this document. Requests for reproduction of this document should be directed to the 3GPP2 Secretariat at secretariat@3gpp2.org. Requests to reproduce individual Organizational Partner's documents should be directed to that Organizational Partner. See www.3gpp2.org for more information.

Revision History

Revision	Description of Changes	Date
Rev 0 v1.0	Initial publication	January 2011

Table of Contents

1	1. Introduction.....	1
2	1.1 Scope.....	1
3	1.2 Document Convention	1
4	2. References.....	1
5	2.1 Normative References.....	1
6	2.2 Informative References.....	4
7	3. Definitions, Symbols and Abbreviations.....	4
8	3.1 Definitions.....	4
9	3.2 Symbols and Abbreviations	4
10	4. BCMCS Media Types (CODECS).....	5
11	4.1 General requirements	5
12	4.2 Speech.....	5
13	4.3 Video.....	5
14	4.4 Audio.....	6
15	4.5 Text	6
16	4.6 Timed Text.....	7
17	4.7 Synthetic Audio	7
18	4.8 Still Image.....	7
19	4.9 Bitmap Graphics	7
20	4.10 Vector Graphics	7
21	4.11 3GPP2 File Formats.....	7
22	5. BCMCS Transport Protocols	8
23	5.1 Protocol Reference Model	8
24	5.2 Overview.....	8
25	5.2.1. BCMCS Download Delivery Method.....	8
26	5.2.2. BCMCS Streaming Delivery Method.....	9
27	6. Service and Session Description	9
28	6.1 XML Schema	10
29	7. Download Delivery.....	10
30	7.1 Base protocol – ALC, FLUTE.....	10

Table of Contents

1	7.1.1. Content Encoding.....	10
2	7.1.2. File Descriptions	11
3	7.1.3. File Versioning.....	11
4	7.1.4. Indication of End of File and End of Session	11
5	7.1.4.1. Determining End of File Delivery	11
6	7.1.4.2. Signaling End of File Delivery Session	12
7	7.1.5. Signaling of Parameters	12
8	7.1.5.1. Signaling of Parameters with Basic ALC/FLUTE Headers	12
9	7.1.5.2. Signaling of Parameters with LCT Extension Header	12
10	7.1.5.3. Signaling of Parameters with FLUTE Extension Headers	13
11	7.1.5.4. Signaling of Parameters with FDT Instances.....	13
12	7.1.6. FLUTE FDT Instance XML Schema.....	14
13	7.1.6.1. XML Schema (Normative)	14
14	7.1.6.2. XML Instance Example (Informative).....	14
15	7.2 FEC	14
16	8. Streaming delivery	15
17	8.1 Transport Protocol	15
18	8.1.1. Speech.....	16
19	8.1.2. Video.....	16
20	8.1.3. Audio.....	16
21	8.1.4. Timed Text.....	16
22	8.1.5. Synthetic Audio	16
23	Annex A FLUTE FDT XML Schema (normative)	17

1	<u>Figures</u>	
2	Figure 1 BCMCS Protocol Stack in the MS and the BCMCS Content Server	8
3	Figure 2 Building Block Structure of File Delivery in BCMCS	9

1 Foreward

2 (This foreword is not part of this specification.)

3 This technical specification recommends codecs as well as protocols for Broadcast Multicast
4 Services (BCMCS).

1 1. Introduction

2 1.1 Scope

3 This specification recommends media types and protocols to support delivery of broadcast
4 multicast services.

5 1.2 Document Convention

6 “Shall” and “shall not” identify requirements to be followed strictly to conform to this document
7 and from which no deviation is permitted. “Should” and “should not” indicate that one of several
8 possibilities is recommended as particularly suitable, without mentioning or excluding others,
9 that a certain course of action is preferred but not necessarily required, or that (in the negative
10 form) a certain possibility or course of action is discouraged but not prohibited. “May” and “need
11 not” indicate a course of action permissible within the limits of the document. “Can” and
12 “cannot” are used for statements of possibility and capability, whether material, physical or
13 causal.

14 2. References

15 The following standards are referenced in this text. At the time of publication, the editions
16 indicated were valid. All standards are subject to revision, and parties to agreements based upon
17 this document are encouraged to investigate the possibility of applying the most recent editions
18 of the standards indicated below. ANSI and TIA maintain registers of currently valid national
19 standards published by them.

20 2.1 Normative References

- 21 [1] 3GPP2: C.S0014-D "Enhanced Variable Rate Codec, Speech Service Options 3, 68, 70,
22 and 73 for Wideband Spread Spectrum Digital Systems", January 2010.
- 23 [2] ITU-T: Recommendation H.264 "Advanced video coding for generic audiovisual
24 services", March 2005. also available as ISO/IEC 14496-10: "Information technology -
25 Coding of audio-visual objects - Part 10: Advanced Video Coding", 2009.
- 26 [3] ITU-T: Recommendation H.263: Video coding for low bit rate communications, January
27 2005.
- 28 [4] ITU-T: Recommendation H.263 Annex X: Profiles and Levels Definition, April 2001.
- 29 [5] IETF: RFC 3984, S. Wenger et al., "[RTP Payload Format for H.264 Video](#)", Feb 2005.
- 30 [6] 3GPP TS 26.401: "General audio codec audio processing functions; Enhanced aacPlus
31 general audio codec; General description".
- 32 [7] W3C: "XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition) ",
33 August 2002, <http://www.w3.org/TR/2002/REC-xhtml1-20020801/>

- 1 [8] The Unicode Consortium: "The Unicode Standard", Version 3.0 Reading, MA, Addison-
2 Wesley Developers Press, 2000, ISBN 0-201-61633-5.
- 3 [9] ISO/IEC: 10646:2003 "Information technology - Universal Multiple-Octet Coded
4 Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", 2003.
- 5 [10] 3GPP: TS 26.245: "Transparent end-to-end Packet switched Streaming Service (PSS);
6 Timed text format".
- 7 [11] 3GPP2: C.S0050-B: 3GPP2 File formats for multimedia services, June 2007.
- 8 [12] MIDI Manufacturers Association: "Scalable Polyphony MIDI Specification", Version
9 1.0, RP-34, Los Angeles, CA, February 2002.
- 10 [13] MIDI Manufacturers Association: "Scalable Polyphony MIDI Device 5-to-24 Note
11 Profile for 3GPP", Version 1.0, RP-35, Los Angeles, CA, February 2002.
- 12 [14] MIDI Manufacturers Association: "Standard MIDI Files 1.0", RP-001, in "The Complete
13 MIDI 1.0 Detailed Specification, Document Version 96.1", Los Angeles, CA, USA,
14 February 1996.
- 15 [15] MIDI Manufacturers Association: Mobile DLS, MMA specification v1.0, RP-41 Los
16 Angeles, CA, USA. 2004.
- 17 [16] MIDI Manufacturers Association: Mobile XMF Content Format Specification, MMA
18 specification v1.0, RP-42, Los Angeles, CA, USA. 2004.
- 19 [17] ITU-T: Recommendation T.81 (1992) | ISO/IEC 10918-1:1993: "Information technology
20 - Digital compression and coding of continuous-tone still images - Requirements and
21 guidelines".
- 22 [18] Eric Hamilton, C-Cube Microsystems: "JPEG File Interchange Format", Version 1.02,
23 September 1992. Available at: <http://www.jpeg.org/public/jfif.pdf>
- 24 [19] CompuServe: "GIF Graphics Interchange Format: A Standard defining a mechanism for
25 the storage and transmission of raster-based graphics information", CompuServe
26 Incorporated, Columbus, OH, USA, 1987. See at
27 <http://www.dcs.ed.ac.uk/home/mxr/gfx/2d/GIF87a.txt>.
- 28 [20] CompuServe: "Graphics Interchange Format: Version 89a", CompuServe Incorporated
29 Columbus, OH, USA, 1990.
- 30 [21] IETF: RFC 2083, T. Boutell, "[PNG \(Portable Networks Graphics\) Specification Version
31 1.0](#)", March 1997.
- 32 [22] W3C: April 2005: "Scalable Vector Graphics (SVG) Full 1.2 Specification",
33 <http://www.w3.org/TR/SVG12/>.
- 34 [23] W3C: December 2009: "Scalable Vector Graphics (SVG) Tiny 1.2 Specification",
35 <http://www.w3.org/TR/SVGTiny12/>.
- 36 [24] ECMA: Standard ECMA-327 (June 2001): "ECMAScript 3rd Edition Compact Profile".
- 37 [25] IETF: RFC 1952, P. Deutsch, "[GZIP file format specification version 4.3](#)", May 1996

- 1 [26] 3GPP2: X.S0022-A, "Broadcast and Multicast Service in cdma2000 Wireless IP
2 Network, February 2007.
- 3 [27] IETF: RFC 5775, M. Luby et al., "[Asynchronous Layered Coding \(ALC\) Protocol
4 Instantiation](#)", April 2010.
- 5 [28] IETF: RFC 3926, T. Paila et al., "[FLUTE - File Delivery over Unidirectional Transport](#)",
6 October 2004.
- 7 [29] IETF: RFC 5052, M. Watson et al., "[Forward Error Correction \(FEC\) Building Block](#)",
8 August 2007.
- 9 [30] IETF: RFC 5651, M. Luby et al., "[Layered Coding Transport \(LCT\) Building Block](#)",
10 October 2009.
- 11 [31] IETF: I-D, M. Luby et al., "[RaptorQ Forward Error Correction Scheme for Object
12 Delivery](#)", draft-ietf-rmt-bb-fec-raptorq, August 2010.
- 13 [Editor Note: The above document is a work in progress and should not be referenced unless and
14 until it is approved and published. Until such time as this Editor's Note is removed, the inclusion
15 of the above document is for informational purposes only.]
- 16 [32] IETF: RFC 5445, M. Watson, "[Basic Forward Error Correction \(FEC\) Schemes](#)", March
17 2009.
- 18 [33] IETF: RFC 4566, M. Handley et al., "[SDP: Session Description Protocol](#)", July 2006.
- 19 [34] OMA BCAST: "[File and Stream Distribution for Mobile Broadcast Services](#)", version
20 1.0, February 2009.
- 21 [35] OMA BCAST: "[Service Guide for Mobile Broadcast Services](#)", version 1.0, February
22 2009.
- 23 [36] OMA BCAST: "[Broadcast Distribution System Adaptation – 3GPP2/BCMCS](#)", version
24 1.0, February 2009.
- 25 [37] IETF: RFC 3550, H. Schulzrinne et al., "[RTP: A Transport Protocol for Real-Time
26 Applications](#)", July 2003.
- 27 [38] IETF: RFC 5188, H. Desineni and Q. Xie, "[RTP Payload format for Enhanced variable
28 Rate wideband codec \[EVRC-WB\] and media subtype updates for EVRC-B Codec](#)", Feb
29 2008.
- 30 [39] IETF: Z. Fang, "[RTP payload format for Enhanced Variable Rate Narrowband-Wideband
31 Codec](#)". draft-zfang-avt-rtp-evrc-nw-02, November 2010.
- 32 [Editor Note: The above document is a work in progress and should not be referenced unless and
33 until it is approved and published. Until such time as this Editor's Note is removed, the inclusion
34 of the above document is for informational purposes only.]
- 35 [40] IETF: RFC 4629, J. Ott et al., "[RTP Payload format for ITU-T Rec. H.263](#)". January
36 2007.
- 37 [41] IETF: RFC 3640, F. de Bont et al., "[RTP Payload format for transport of MPEG-4
38 Elementary Streams](#)", November 2003.

- 1 [42] IETF: RFC 4396, J. Rey and Y. Matsui, "[RTP payload format for 3GPP2 Timed Text](#)",
2 February 2006.
- 3 [43] IETF: RFC 4695, J. Lazzaro and J. Wawrzynek, "[RTP Payload format for MIDI](#)",
4 November 2006.

5 **2.2 Informative References**

6 This section provides references to other documents that may be useful for the reader of this
7 document.

- 8 [44] ETSI: 100 974 (v6.2.0) “Digital cellular telecommunication system (Phase 2+); Mobile
9 Application Part (MAP) specification (GSM 09.02 Version 6.2.0 Release 1997)”,
10 November 1998.

11 **3. Definitions, Symbols and Abbreviations**

12 This section contains definitions, symbols and abbreviations that are used throughout the
13 document.

14 **3.1 Definitions**

15 *Fountain Code* – In a fountain FEC code, as many encoding symbols as needed can be
16 generated by the encoder on-the-fly from the source symbols of a block.

17 *RaptorQ* – A systematic FEC fountain code with superior flexibility, support for larger source
18 block sizes, and better coding efficiency than Raptor codes in RFC5053.

19 **3.2 Symbols and Abbreviations**

20	3GPP	3rd Generation Partnership Project
21	3GPP2	3rd Generation Partnership Project 2
22	ALC	Asynchronous Layered Coding
23	AVC	Advanced Video Coding
24	BCMCS	Broadcast Multicast Services
25	DCT	Discrete Cosine Transform
26	EVRC-NW	Enhanced Variable Rate Codec Narrowband-Wideband
27	EVRC-WB	Enhanced Variable Rate Codec Wideband
28	FDT	File Delivery Table
29	FEC	Forward Error Correction
30	FLUTE	File Delivery over Unidirectional Transport
31	GIF	Graphics Interchange Format

1	IDR	Instantaneous Decoding Refresh
2	IETF	Internet Engineering Task Force
3	ITU	International Telecommunications Union
4	LCT	Layered Coding Transport
5	MIME	Multipurpose Internet Mail Extensions
6	OMA	Open Mobile Alliance
7	RFC	Request for Comments
8	RTCP	Real-time Transport Control Protocol
9	RTP	Real-time Transport Protocol
10	SDP	Session Description Protocol
11	SEI	Supplemental Enhancement Information (RTP message type)
12	SP-MIDI	Scalable Polyphony Musical Instrument Digital Interface
13	UDP	User Datagram Protocol
14	XHTML	Extensible Hypertext Markup Language
15	XML	Extensible Markup Language

16 **4. BCMCS Media Types (CODECS)**

17 This section describes the default codecs for each media category.

18 **4.1 General requirements**

19 The set of media decoders that are supported by the BCMCS Terminal to support a particular
 20 media type are defined below. Speech, Audio, Video, Timed Text and Scene description media
 21 decoders are relevant for both BCMCS Download and Streaming delivery. Other media decoders
 22 are only relevant for BCMCS Download delivery.

23 **4.2 Speech**

24 If speech is supported, the BCMCS Terminal shall support EVRC-NW [1] and EVRC-WB [1]
 25 decoders.

26 **4.3 Video**

27 If video is supported, the BCMCS terminal shall support:

- 28 • H.264 Constrained Baseline Profile level 1.3 video decoder (ITU-T Recommendation
 29 H.264|ISO/IEC 14496-10 [2]) without requirements on output timing conformance (annex C
 30 of [2]).
- 31 • H.263 profile 0 level 45 decoder [3] and [4]

1 Note that BCMCS does not offer dynamic negotiation of media codecs.

2 When H.264 (AVC) is in use in the BCMCS streaming delivery method, it is recommended to
3 transmit H.264 (AVC) parameter sets within the SDP description of a stream (using sprop-
4 parameter-sets MIME/SDP parameter - RFC3984 [5]), and it is not recommended to transmit
5 parameter sets within the RTP stream. Moreover, it is not recommended to reuse any parameter
6 set identifier value that appeared previously in the SDP description or in the RTP stream.

7 However, if a sequence parameter set is taken into use or updated within the RTP stream, it shall
8 be contained at least in each IDR access unit and each access unit including a recovery point SEI
9 message in which the sequence parameter set is used in the decoding process. If a picture
10 parameter set is taken into use or updated within the RTP stream, it shall be contained at the
11 latest in the first such access unit in each entry sequence that uses the picture parameter set in the
12 decoding process, in which an entry sequence is defined as the access units between an IDR
13 access unit or an access unit containing a recovery point SEI message, inclusive, and the next
14 access unit, exclusive, in decoding order, which is either an IDR access unit or contains a
15 recovery point SEI message.

16 There are no requirements on output timing conformance [2] for BCMCS Terminals.

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

25 **4.4 Audio**

26 If audio is supported, the BCMCS terminal shall support Enhanced AACPlus [6] decoder.

27 **4.5 Text**

28 The text decoder is intended to enable formatted text in a Synchronized Multimedia Integration
29 Language (SMIL) presentation. If text is supported, a BCMCS terminal shall support

- 30 • text formatted according to XHTML Mobile Profile [7];
- 31 • rendering a SMIL presentation where text is referenced with the SMIL 2.0 "text" element
32 together with the SMIL 2.0 "src" attribute.

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

- 34 • UTF-8, [8];
- 35 • UTF-16, [9].

1 **4.6 Timed Text**

2 If timed text is supported, BCMCS terminal shall support [10]. Timed text may be transported
3 over RTP or downloaded in 3GPP2 [11] files using Basic profile.

4 **4.7 Synthetic Audio**

5 If synthetic audio is supported, the BCMCS terminal shall support the Scalable Polyphony
6 Musical Instrument Digital Interface (SP-MIDI) content format defined in Scalable Polyphony
7 MIDI Specification [12] and the device requirements defined in Scalable Polyphony MIDI
8 Device 5-to-24 Note Profile for 3GPP [13].

9 SP-MIDI content is delivered in the structure specified in Standard MIDI Files 1.0 [14], either in
10 format 0 or format 1.

11 In addition the Mobile DLS instrument format defined in [15] and the Mobile XMF content
12 format defined in [16] should be supported.

13 **4.8 Still Image**

14 If still images are supported, ISO/IEC JPEG [17] together with JFIF [18] decoders shall be
15 supported by the BCMCS terminal. The support for ISO/IEC JPEG only applies to the following
16 two modes:

- 17 • baseline DCT, non-differential, Huffman coding
- 18 • progressive DCT, non-differential, Huffman coding.

19 **4.9 Bitmap Graphics**

20 If bitmap graphics is supported, the following bitmap graphics decoders shall be supported:

- 21 • GIF89a, [20];

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

- 23 • GIF87a, [19];
- 24 • PNG, [21].

25 **4.10 Vector Graphics**

26 If vector graphics is supported, SVG Tiny 1.2 [22], [23] and ECMAScript [24] shall be
27 supported.

28 NOTE 1: The compression format for SVG content is GZIP [25], in accordance with the SVG
29 specification [22].

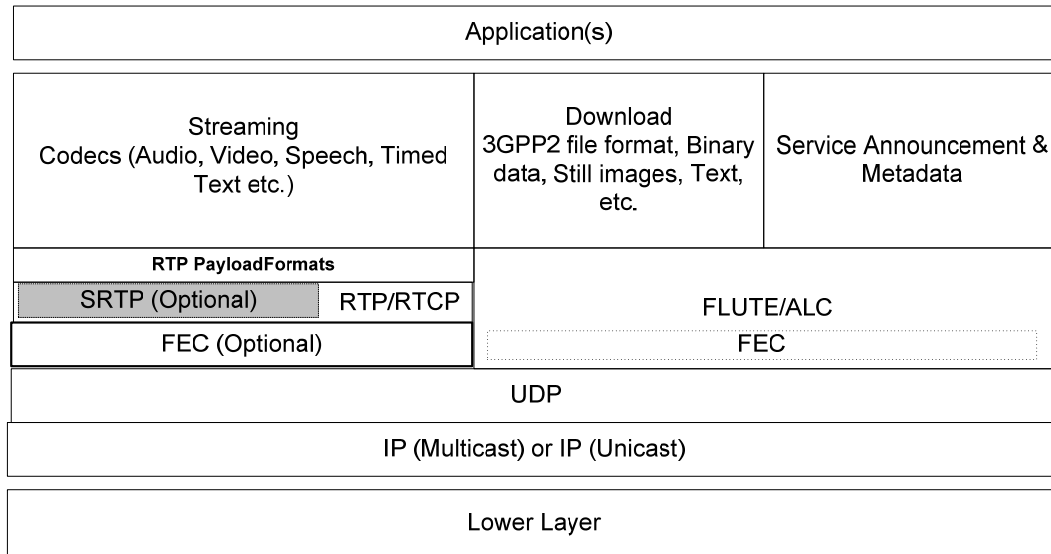
30 **4.11 3GPP2 File Formats**

31 A BCMCS terminal shall support the Basic profile and the Extended presentation profile of the
32 3GPP2 file format defined in C.S0050-B [11].

1 5. BCMCS Transport Protocols

2 5.1 Protocol Reference Model

3 Figure 1 illustrates the protocol stack used by BCMCS in the MS and BCMCS Content Server
4 (see [26]).



5
6 **Figure 1 BCMCS Protocol Stack in the MS and the BCMCS Content Server**

7 5.2 Overview

8 Two delivery methods are defined in BCMCS in this specification, the download delivery
9 method and the streaming delivery method.

10 5.2.1. BCMCS Download Delivery Method

11 BCMCS download delivery method uses ALC [27] when delivering content over BCMCS
12 broadcast channel. FLUTE [28] is built on the Asynchronous Layered Coding (ALC) protocol
13 instantiation. ALC combines the Layered Coding Transport (LCT) building block [30], a
14 congestion control building block, and the Forward Error Correction (FEC) building block [29]
15 to provide congestion controlled reliable asynchronous delivery of content to an unlimited
16 number of concurrent receivers from a single sender. As mentioned in [27], congestion control is
17 not appropriate in the type of environment that BCMCS download delivery is provided, and thus
18 congestion control is not used for BCMCS download delivery. See Figure 2 for an illustration of
19 file delivery building block structure used for BCMCS. FLUTE/ALC is carried over UDP/IP,
20 and is independent of the IP version and the underlying link layers used.

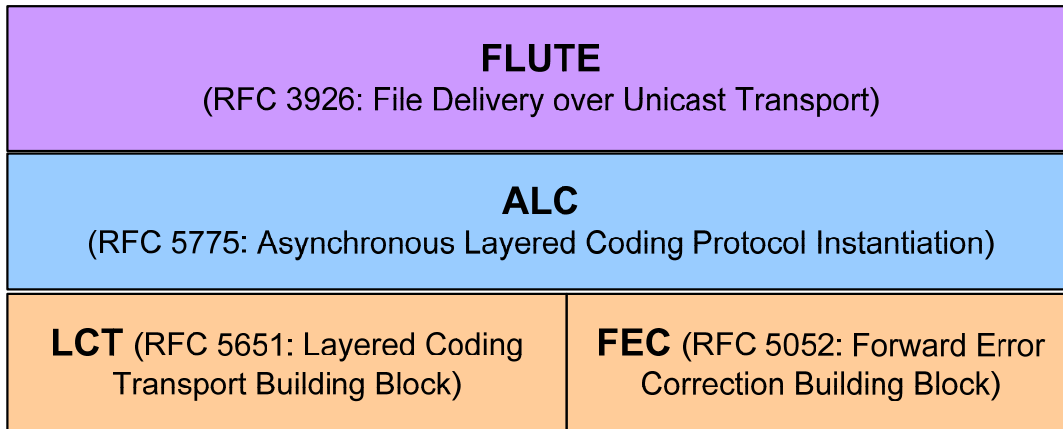


Figure 2 Building Block Structure of File Delivery in BCMCS

ALC uses the LCT building block to provide in-band session management functionality. The LCT building block provides transport level support for reliable content delivery and stream delivery protocols. 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. In this specification, RaptorQ FEC as specified in [32] is used, which includes its specific usage that is equivalent to no FEC coding (see section 7.3 of [32]).

5.2.2. BCMCS Streaming Delivery Method

BCMCS streaming delivery method is used to deliver multimedia content (e.g., speech, audio, video etc) over a BCMCS broadcast channel. The streaming delivery method is particularly useful for multicast and broadcast of scheduled streaming content. BCMCS streaming delivery method uses RTP as a transport protocol. RTP provides a means for sending real-time or streaming data over UDP. RTP provides RTCP for feedback about the transmission quality. The transmission of RTCP packets in the downlink (sender reports) is allowed and in the reverse link (receiver reports) is disabled. The RTP payload format is specified in Section 4. The use of FEC for BCMCS streaming by the sender is optional. In the case where the FEC is not used by the sender, the FEC Layer should not be used (i.e., RTP is mapped onto UDP directly).

6. Service and Session Description

BCMCS includes Service Discovery/Announcement, content subscription, BCMCS Information Acquisition, content availability determination, BCMCS registration, BCMCS content delivery, and BCMCS deregistration as specified in Section 5 of [26].

The BCMCS Service Discovery/Announcement using interactive announcement function via BCMCS Information Acquisition procedures is specified in Section 5.1 and Section 5.3 of [26].

The MS may support the OMA BCAST Service Guide as specified in [35]. The OMA BCAST Service Guide enables the service and content providers to describe the services and content they make available, or offer for subscription or purchase, as mobile broadcast services either over the broadcast channel or over the interaction channel. The OMA BCAST Service Guide delivery procedures for BCMCS are specified in Section 6.3 of [36].

1 The download delivery method of Section 7 is used for BCMCS Service
2 Discovery/Announcement over a broadcast channel.

3 For BCMCS download and streaming delivery, session description information provides the
4 associated media details, transport addresses, and other session description metadata, in the SDP
5 format as defined in [33]. BCMCS session description information can be provided to the MS
6 either via BCMCS Information Acquisition mechanisms as specified in [26], or in the BCAST
7 Service Guide [35] as profiled by [36], if the MS supports the BCAST Service Guide (SG).

8 If the MS uses BCMCS Information Acquisition procedures for session description information,
9 the MS shall use the HTTP-based Information Acquisition Request and Information Acquisition
10 Response messages as specified in [26]. Specifically, the session description information is
11 returned in the <SDP> element of the “<BCMCS><Response>” message, in response to the
12 HTTP-based “<BCMCS><Request>” message with the “<RequestTypeVal>” set to ‘AppInfo’
13 or ‘All’. If the MS uses BCAST SG for session description information, the MS shall follow the
14 procedures as specified in Section 6.3.4 of [36].

15 **6.1 XML Schema**

16 XML schema for BCMCS Information Acquisition is specified in Section 7 of [26].

17 **7. Download Delivery**

18 **7.1 Base protocol – ALC, FLUTE**

19 The BCMCS download delivery method shall employ the FLUTE protocol as specified in [28],
20 and described in Section 5, to transmit file content from the BCMCS Content Server to the MS.

21 FLUTE uses ALC [27] which uses the FEC building block [29] to provide reliability for the
22 broadcast channel. The FEC building block allows the choice of an appropriate FEC code to be
23 used within ALC. The BCMCS Content Server and the MS shall use RaptorQ FEC as defined in
24 [31]. The implementation can correspond to either full FEC encoding which includes repair
25 symbols, or trivial FEC encoding (no repair symbols sent) equivalent to the Compact No-Code
26 FEC scheme as specified in [32]. RaptorQ (with FEC Encoding ID of 6 (TBC)), a fully-
27 specified FEC scheme, is a fountain code in which as many encoding symbols as needed can be
28 generated by the encoder on-the-fly from the source symbols of a source block of data. The
29 decoder is able to recover the source block from any set of encoding symbols only slightly more
30 in number than the number of source symbols. The RaptorQ code as specified is a systematic
31 code, meaning that all the source symbols are among the encoding symbols that can be
32 generated.

33 **7.1.1. Content Encoding**

34 Files may be content encoded using the generic GZIP algorithm [25]. Terminals shall support
35 GZIP content decoding of files. For GZIP-encoded files, the “Content-Encoding” attribute of the
36 file description, carried in the FDT in FLUTE file delivery, shall be assigned the value “gzip”.

1 **7.1.2. File Descriptions**

2 The content file object in BCMCS download delivery is declared by file description information.
 3 Each set of file descriptions comprises, at the minimum, information that establishes a mapping
 4 between the file's identification in the form of a URI, and a Transmission Object Identifier (TOI)
 5 identifying the specific transmission object in the file delivery session (or FLUTE session). In
 6 FLUTE, the file description information is logically referred to as the File Delivery Table (FDT),
 7 and is conveyed in any FDT Instance delivered in the session that describes that file, i.e., the
 8 FDT Instance contains a 'File' element with 'Content-Location' attribute set to the file URI. For
 9 such file descriptions, its version is the wrap-around adjusted value of the FDT Instance ID¹ (sent
 10 as the LCT Extension Header 'EXT_FDT', also referred to as the FDT Instance Header), and the
 11 expiry time is the 'Expires' attribute value of the FDT Instance.

12 **7.1.3. File Versioning**

13 A BCMCS content file (as identified by its URI) may be associated with multiple transmission
 14 objects (i.e., with multiple TOI values) during the lifetime of the file distribution session. In such
 15 event, for the BCMCS Content Server, the transmission object declared by the file description
 16 with the highest version number shall represent the latest version of the file. For an FDT
 17 Instance containing a new set of data elements (e.g., complementary to the previously sent ones),
 18 the TOI associated with a given file may be left unchanged, which means that the version of the
 19 file did not change.

20 Note: The MS should not send post-reception file repair requests for an old version of a file once
 21 a file description declaring a newer version of the file is received.

22 **7.1.4. Indication of End of File and End of Session**

23 **7.1.4.1. Determining End of File Delivery**

24 The MS shall conclude that the distribution of a file has ended when one of the following events
 25 occur:

- 26 • It determines that the file delivery session has ended, as specified in Section 7.1.4.2.
- 27 • It receives an end-of-object packet (ALC packet with B-flag in LCT header set to 'true') for
 28 the transmission object representing the latest version of the file. The MS shall not determine
 29 that the delivery has ended for this file in case the end-of-object packet belongs to an older
 30 version of the file.

31 The MS should also conclude that the delivery of a file has ended upon detecting, among the set
 32 of file descriptions describing the latest version of the file, that the last file description to expire
 33 has expired. The expiry time of this file description, when interpreted as the anticipated end time
 34 of delivery for this file, is therefore updated when the MS receives a file description describing a
 35 newer version of the file, or a file description describing the latest version of the file and expiring
 36 later than the formerly signaled expiry time.

¹ 'wrap-around adjusted value' refers to taking into consideration counter wrap-around, as well as assuming only those values for which the previous FDT Instances have expired.

1 **7.1.4.2. Signaling End of File Delivery Session**

2 A file delivery session is considered complete when one of the following events occurs:

- 3 • The file delivery session is declared by an SDP-formatted session description, the stop time
4 provided for this session is bounded (i.e., second sub-field of the “t=” field is not null), and
5 this stop time is reached;
- 6 • The MS receives an end-of-session packet (ALC packet with A-flag in the LCT header set to
7 true);
- 8 • The MS decides to exit the session.

9 **7.1.5. Signaling of Parameters**

10 **7.1.5.1. Signaling of Parameters with Basic ALC/FLUTE Headers**

11 The default LCT header fields shall be as specified in [30], [27] and [28] with the following
12 additional specializations and changes:

- 13 • The Congestion Control Information (CCI) field shall not be included in the header. The
14 Congestion control flag (C) in the header, indicating the length of the CCI field, shall be
15 ignored.
- 16 • The Transport Object Identifier (TOI) field should be of length 16 bits (O=0, H=1) or 32 bits
17 (O=1, H=0). The maximum length of Transmission Object Identifier (TOI) field should be 32
18 bits.
- 19 • The Transport Session Identifier (TSI) field length shall be equal to the Transport Object
20 Identifier (TOI) field length (S=O).
- 21 • The terminal shall support a TOI field length up to 112 bits.
- 22 • Only TOI ‘0’ (zero) shall be used for FDT Instances.
- 23 • The following features may be used for signaling the end of session and the end of object
24 transmission to the receiver:
 - 25 ○ The Close Session flag (A) for indicating the end of a session.
 - 26 ○ The Close Object flag (B) for indicating the end of an object.

27 In FLUTE the following applies:

- 28 • EXT_TIME header extension of LCT is not supported
- 29 • The LCT header length (HDR_LEN) shall be set to the total length of the LCT header in
30 units of 32-bit words.
- 31 • The FEC Payload ID shall be set according to [31], such that an 8-bit SBN (Source Block
32 Number) and then the 24-bit ESI (Encoding Symbol ID) are assigned.

33 **7.1.5.2. Signaling of Parameters with LCT Extension Header**

34 In order to provide timing information related to an ALC/FLUTE session:

- 35 • the BCMCS Content Server may use the EXT_TIME LCT extension header, and
- 36 • the MS may support the EXT_TIME LCT extension header.

1 **7.1.5.3. Signaling of Parameters with FLUTE Extension Headers**

2 LCT Header Extension fields EXT_FTI, EXT_FDT and EXT_CENC (the latter two as specified
3 in FLUTE [28]) shall be constrained as follows:

- 4 • EXT_FTI shall be included in every FLUTE packet carrying FEC Object Transmission
5 Information symbols belonging to any FDT Instance.
- 6 • FDT Instances shall not be content encoded and therefore EXT_CENC shall not be used.

7 Furthermore, the following applies:

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

10 **7.1.5.4. Signaling of Parameters with FDT Instances**

11 The FLUTE FDT Instance schema defined in Section 7.1.6 shall be used. In addition, the
12 following applies to both the FDT-Instance level information and all files of a FLUTE session.

13 The inclusion of the following FDT Instance data elements is mandatory according to the
14 FLUTE specification [28].

- 15 • Content-Location (URI of a file);
- 16 • TOI (Transmission Object Identifier of a file instance);
- 17 • Expires (expiry data for the FDT Instance).

18 The inclusion of the following FDT Instance data elements is mandatory in accordance with
19 [31]:

- 20 • Transfer-Length
- 21 • FEC-OTI-FEC-Encoding-ID;
- 22 • FEC-OTI-Encoding-Symbol-Length;
- 23 • FEC-OTI-Scheme-Specific-Info.

24 Because [31] represents a fully-specified FEC scheme for which the FEC Instance ID is not
25 applicable, the following FDT Instance data element shall not be included:

- 26 • FEC-OTI-FEC-Instance-ID.

27 These optional FDT Instance data elements may be included for download delivery in BCMCS:

- 28 • Complete (the signaling that an FDT Instance provides a complete, and subsequently
29 unmodifiable, set of file parameters for a FLUTE session may be performed according to this
30 method);
- 31 • Content-Length (source file length in bytes);
- 32 • Content-Type (content MIME type);
- 33 • Content-Encoding;
- 34 • Content-MD5 (It is recommended to indicate the MD5 hash value whenever multiple
35 versions of the file, i.e., distinct file objects identified by the same Content-Location, are
36 anticipated for the download session(s). Note that the MD5 hash value is calculated over the
37 file as transported by FLUTE, i.e., after any gzip compression as indicated by Content-
38 Encoding, or before decompression.)

1 7.1.6. FLUTE FDT Instance XML Schema

2 7.1.6.1. XML Schema (Normative)

3 The syntax of the BCMCS FLUTE FDT Instance is given by the XML schema shown in
4 Appendix A.

5 7.1.6.2. XML Instance Example (Informative)

6 The following XML document provides an example of a FDT Instance for BCMCS download
7 delivery:

```
8
9 <?xml version="1.0" encoding="UTF-8"?>
10 <FDT-Instance
11   xmlns="urn:3gpp2:xml:bcmcs:fd:fdt:1.0"
12   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
13   xmlns:foo="urn:foo"
14   xsi:schemaLocation="http://www.3gpp2.org/tech/profiles/bcmcs_fd_fdt-v1_0.xsd"
15   Complete="true"
16   Content-Encoding="gzip"
17   Content-Type="video/3gpp2"
18   FEC-OTI-Encoding-Symbol-Length="512"
19   Expires="331129600"
20   FEC-OTI-FEC-Encoding-ID="0"
21   xsi:type="FDT-InstanceType-Bcmcs">
22   <File
23     xsi:type="FileType-Bcmcs"
24     Content-Type="application/sdp"
25     Content-Length="7543"
26     Transfer-Length="4294"
27     Version-ID-Length="8"
28     TOI="2"
29     FEC-OTI-Encoding-Symbol-Length="16"
30     Content-Location="http://www.example.com/fancy-session/main.sdp">
31     <foo:yousay>goodbye</foo:yousay>
32   </File>
33   <File
34     xsi:type="FileType-Bcmcs"
35     Content-Length="161934"
36     Transfer-Length="157821"
37     TOI="3"
38     FEC-OTI-FEC-Encoding-ID="1"
39     FEC-OTI-Encoding-Symbol-Length="512"
40     FEC-OTI-Scheme-Specific-Info="AAEBBA=="
41     Content-Location="http://www.example.com/fancy-session/trailer.3gp2"
42     foo:myattribute="myvalue">
43   </File>
44   <foo:andisay>hello</foo:andisay>
45 </FDT-Instance>
```

46 7.2 FEC

47 The MS shall support the “RaptorQ FEC scheme” [31] (FEC Encoding ID 6(TBC)). This
48 includes support for:

- 49 • a decoder for the RaptorQ FEC scheme such that it complies with the Requirements of a
50 Compliant Decoder described in [31]. ”.

- 1 • Decoding of a sub-block with maximum size of 262,144 bytes,
- 2 • the FEC Payload ID format as defined in [31], and
- 3 • the FEC Object Transmission Information format as defined in [31].

4 The BCMCS Content Server shall support the procedures specified in [31] (FEC Encoding ID
5 6(TBC)). Note that this implies that for sending FDT Instances, FEC Encoding ID 6 (TBC) shall
6 be used.

7 In case the BCMCS Content Server sends both source and repair symbols, then it shall not send
8 any encoding symbol twice before sending all possible encoding symbols at least once. In
9 addition, the Example Parameter Derivation Algorithm in [31], should be used for the derivation
10 of the transport parameters:

- 11 • T, the symbol size in bytes,
- 12 • Z, the number of source blocks,
- 13 • N, the number of sub-blocks in each source block.

14 The previous three parameters are derived from the following input parameters:

- 15 • F, the transfer length of the object, in bytes,
- 16 • WS, a target for the sub-block size, in bytes, shall be set to $WS=262,144$ bytes,
- 17 • Al, the symbol alignment parameter, in bytes, should be set to $Al=4$,
- 18 • P', the maximum packet payload size, in bytes, which should be a multiple of Al,
- 19 • SS, a parameter such that the desired lower bound on the sub-symbol size is $SS*Al$. SS
20 should be set to $SS=8$, and shall be chosen such that $SS*Al$ is at least P',
- 21 • K'_max, the maximum number of source symbols per source block, shall be set to
22 $K'_max=56,404$.

23 In case the BCMCS Content Server only sends source symbols, but no repair symbols, the
24 BCMCS Content Server shall not use sub-blocking. In this case the size of each source block
25 shall not exceed WS, i.e., the product of K'_max and T shall not exceed WS, and the BCMCS
26 Content Server may send duplicated source symbols.

27 **8. Streaming delivery**

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

32 **8.1 Transport Protocol**

33 RTP [37] is the transport protocol for BCMCS streaming delivery. RTP provides a means for
34 sending real-time or streaming data over UDP. RTP provides RTCP for feedback about the
35 transmission quality. The transmission of RTCP packets in the downlink (sender reports) is
36 allowed. In this version of the specification, RTCP RR shall be turned off by SDP RR bandwidth
37 modifiers.

38 The RTP payload formats are described below for each of the media types.

1 **8.1.1. Speech**

2 For EVRC-WB [2], the payload format defined in [38] shall be supported.

3 For EVRC-NW [2], the payload format defined in [39] shall be supported.

4 **8.1.2. Video**

5 For H.264 (AVC) [2], the payload format defined in [5] shall be supported.

6 A BCMCS Terminal supporting H.264 (AVC) is required to support all three packetization
7 modes: single NAL unit mode, non-interleaved mode and interleaved mode. For the interleaved
8 packetization mode, a BCMCS Terminal shall support streams for which the value of the "sprop-
9 deint-buf-req" MIME parameter is less than or equal to $\text{MaxCPB} * 1000 / 8$, inclusive, in which
10 "MaxCPB" is the value for Video Coding Layer (VCL) parameters of the H.264 (AVC) profile
11 and level in use, as specified in [2].

12 For H.263 media type [3], the payload format defined in [40] shall be supported.

13 **8.1.3. Audio**

14 For enhanced AACPlus [6] audio media type, the payload format defined in [41] shall be
15 supported.

16 **8.1.4. Timed Text**

17 For timed text [10], the payload format defined in [42] shall be supported.

18 **8.1.5. Synthetic Audio**

19 For SP-MIDI [12], the payload format defined in [43] shall be supported.

1 Annex A FLUTE FDT XML Schema (normative)

```

2
3 <xs:schema xmlns="http://www.3gpp2.org/XMLSchema/BCMCS/v1.0/fd/fdt/1.0"
4 xmlns:xs="http://www.w3.org/2001/XMLSchema"
5 targetNamespace="http://www.3gpp2.org/XMLSchema/BCMCS/v1.0/fd/fdt/1.0"
6 elementFormDefault="qualified">
7 <!-- IETF elements and datatypes -->
8 <xs:element name="FDT-Instance" type="FDT-InstanceType"/>
9 <xs:complexType name="FDT-InstanceType">
10 <xs:sequence>
11 <!-- IETF elements -->
12 <xs:element name="File" type="FileType" maxOccurs="unbounded"/>
13 <!-- other elements -->
14 <xs:any namespace="##other" processContents="skip" minOccurs="0"
15 maxOccurs="unbounded"/>
16 </xs:sequence>
17 <!-- IETF attributes -->
18 <xs:attribute name="Expires" type="xs:string" use="required"/>
19 <xs:attribute name="Complete" type="xs:boolean" use="optional"/>
20 <xs:attribute name="Content-Type" type="xs:string" use="optional"/>
21 <xs:attribute name="Content-Encoding" type="xs:string" use="optional"/>
22 <xs:attribute name="FEC-OTI-FEC-Encoding-ID" type="xs:unsignedByte" use="optional"/>
23 <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length" type="xs:unsignedLong"
24 use="optional"/>
25 <xs:attribute name="FEC-OTI-Encoding-Symbol-Length" type="xs:unsignedLong"
26 use="optional"/>
27 <xs:attribute name="FEC-OTI-Max-Number-of-Encoding-Symbols" type="xs:unsignedLong"
28 use="optional"/>
29 <xs:attribute name="FEC-OTI-Scheme-Specific-Info" type="xs:base64Binary"
30 use="optional"/>
31 <!-- BCMCS attributes -->
32 <xs:attribute name="FullFDT" type="xs:boolean" use="optional" default="false"/>
33 <xs:attribute name="Version-ID-Length" type="xs:unsignedLong" use="optional"/>
34 <!-- other attributes -->
35 <xs:anyAttribute namespace="##other" processContents="skip"/>
36 </xs:complexType>
37 <xs:complexType name="FileType">
38 <xs:sequence>
39 <!-- other elements -->
40 <xs:any namespace="##other" processContents="skip" minOccurs="0"
41 maxOccurs="unbounded"/>
42 </xs:sequence>
43 <!-- IETF attributes -->
44 <xs:attribute name="Content-Location" type="xs:anyURI" use="required"/>
45 <xs:attribute name="TOI" type="xs:positiveInteger" use="required"/>
46 <xs:attribute name="Content-Length" type="xs:unsignedLong" use="optional"/>
47 <xs:attribute name="Transfer-Length" type="xs:unsignedLong" use="optional"/>
48 <xs:attribute name="Content-Type" type="xs:string" use="optional"/>
49 <xs:attribute name="Content-Encoding" type="xs:string" use="optional"/>
50 <xs:attribute name="Content-MD5" type="xs:base64Binary" use="optional"/>
51 <xs:attribute name="FEC-OTI-FEC-Encoding-ID" type="xs:unsignedByte" use="optional"/>
52 <xs:attribute name="FEC-OTI-Maximum-Source-Block-Length" type="xs:unsignedLong"
53 use="optional"/>
54 <xs:attribute name="FEC-OTI-Encoding-Symbol-Length" type="xs:unsignedLong"
55 use="optional"/>
56 <xs:attribute name="FEC-OTI-Max-Number-of-Encoding-Symbols" type="xs:unsignedLong"
57 use="optional"/>
58 <xs:attribute name="FEC-OTI-Scheme-Specific-Info" type="xs:base64Binary"
59 use="optional"/>

```

```
1 <!-- BCMCS attributes -->
2 <xs:attribute name="Version-ID-Length" type="xs:unsignedLong" use="optional"/>
3 <!-- other attributes -->
4 <xs:anyAttribute namespace="##other" processContents="skip"/>
5 </xs:complexType>
6 <!-- BCMCS restrictions datatypes -->
7 <!-- BCMCS : Transfer-Length, Content-Type, FEC-OTI-FEC-Encoding-ID, FEC-OTI-Encoding-
8 Symbol-Length & FEC-OTI-Scheme-Specific-Info required -->
9 <xs:complexType name="FDT-InstanceType-BCMCS">
10 <xs:complexContent>
11 <xs:restriction base="FDT-InstanceType">
12 <xs:sequence>
13 <!-- unchanged elements to be listed again -->
14 <xs:element name="File" type="FileType" maxOccurs="unbounded"/>
15 <xs:any namespace="##other" processContents="skip" minOccurs="0"
16 maxOccurs="unbounded"/>
17 </xs:sequence>
18 <!-- required attributes for BCMCS -->
19 <xs:attribute name="Transfer-Length" type="xs:unsignedLong" use="required"/>
20 <xs:attribute name="Content-Type" type="xs:string" use="required"/>
21 <xs:attribute name="FEC-OTI-FEC-Encoding-ID" type="xs:unsignedByte" use="required"/>
22 <xs:attribute name="FEC-OTI-Encoding-Symbol-Length" type="xs:unsignedLong"
23 use="required"/>
24 <xs:attribute name="FEC-OTI-Scheme-Specific-Info" type="xs:base64Binary"
25 use="required"/>
26 <xs:anyAttribute namespace="##other" processContents="skip"/>
27 </xs:restriction>
28 </xs:complexContent>
29 </xs:complexType>
30 </xs:schema>
```