Technical Specification Group Services and System Aspects *TSGS#25 (04)0632*
Meeting #25, Palm Springs, USA, 13-16 September 2004

Source: **TSG-SA WG4**

Title: **3GPP TS 26.346 version 1.0.0 "Multimedia Broadcast/Multicast Service; Protocols and Codecs" (Release 6)**

Agenda Item: **7.4.3**

Presentation of Specification to TSG SA Plenary

Presentation to: **TSG SA Meeting #25**

Document for presentation: **TS 26.346 "Multimedia Broadcast/Multicast Service; Protocols and Codecs", Version 1.0.0 (Release 6)**

Presented for: **Information**

Abstract of document:

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. This specification does not preclude the use of other delivery methods.

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

Changes since last presentation:

This specification is presented for the first time to TSG SA Plenary.

Outstanding Issues:

A number of Clauses need to be reviewed and /or completed, as this document is actually ~60% ready: to the purpose, a draft version of the specification including Editor's notes, highlights, comments, etc. is attached to this "clean" version. The reader interested to capture the still open issues may refer directly to the "working draft version".

Contentious Issues:

None.

Comment(s):

None.

3GPP TS 26.346 V1.0.0 (2004-09)

Technical Specification

3rd Generation Partnership Project;
Technical Specification Group Services and System Aspects;
Multimedia Broadcast/Multicast Service;
Protocols and Codecs
(Release 6)

Keywords
< UMTS, IP, packet mode, protocol, codec, MBMS>

3GPP

Postal address

3GPP support office address
650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet
http://www.3gpp.org

Contents

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. This 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 this document is the definition of a set of media codecs, formats and transport/application protocols to enable the deployment of MBMS user services. This specification 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. This specification 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: “MBMS User Services”.

[4] 3GPP TS 23.246: “MBMS Architecture and Functional description (Stage 2)”.

[5] 3GPP TS 25.346: “MBMS in the Radio Access Network; Stage 2”.

[6] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications", Schulzrinne H. et al., July 2003.

[7] IETF STD 0006: "User Datagram Protocol", Postel J., August 1980.

[8] IP [TBA]

[9] FLUTE – File Delivery over Unidirectional Transport, IETF Internet Draft, Work in progress, http://www.ietf.org/internet-drafts/draft-ietf-rmt-flute-07.txt.

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

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

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

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

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

[15] Session Description Protocol (SDP) Source Filters - IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdp-srcfilter-05.txt

[16] IETF RFC 3266: "Support for IPv6 in Session Description Protocol (SDP)", S. Olson, G. Camarillo, A. B. Roach, June 2002.

[17] IETF RFC 3048: "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby, January 2001.

[18] IETF RFC 2616: " Hypertext Transfer Protocol -- HTTP/1.1"

[19] IETF RFC 1738: "Uniform Resource Locators (URL)"

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the definitions in 3GPP TR 21.905 [1] as well as the following definitions apply.

Broadcast session: See TS 22.146: "Multimedia Broadcast/Multicast Service" [2].

Multicast joining: See TS 22.146: "Multimedia Broadcast/Multicast Service" [2].

Multicast session: See TS 22.146: "Multimedia Broadcast/Multicast Service" [2].

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

MBMS user services: See TS 22.246: "Multimedia Broadcast/Multicast Service: User Service" [3]. MBMS User Service may use more than one Multimedia Broadcast/Multicast Service (bearer service) and more than one Broadcast and/or Multicast session.

MBMS user service discovery/announcement: The 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. 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: A mechanism used by a MBMS user service to deliver content.

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

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

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

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

FLUTE channel: [TBA]

3.2 Abbreviations

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

ALC	Asynchronous Layered Coding
BM-SC	Broadcast-Multicast Service Centre
CC	Congestion Control
ERT	Expected Residual Time
ESI	Encoding Symbol ID
GGSN	Gateway GPRS Serving Node
GPRS	General Packet Radio Service
FDT	File Delivery Table
FEC	Forward Error Correction
FLUTE	File deLivery over Unidirectional Transport
IP	Internet Protocol
LCT	Layered Coding Transport
MBMS	Multimedia Broadcast/Multicast Service
MS	Mobile Station
RTP	Real-Time Transport Protocol

SBN	Source Block Number
SDP	Session Description Protocol
SCT	Sender Current TimeTOI Transport Object Identifier
TSI	Transport Session Identifier
UDP	User Datagram Protocol
UE	User Equipment
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.

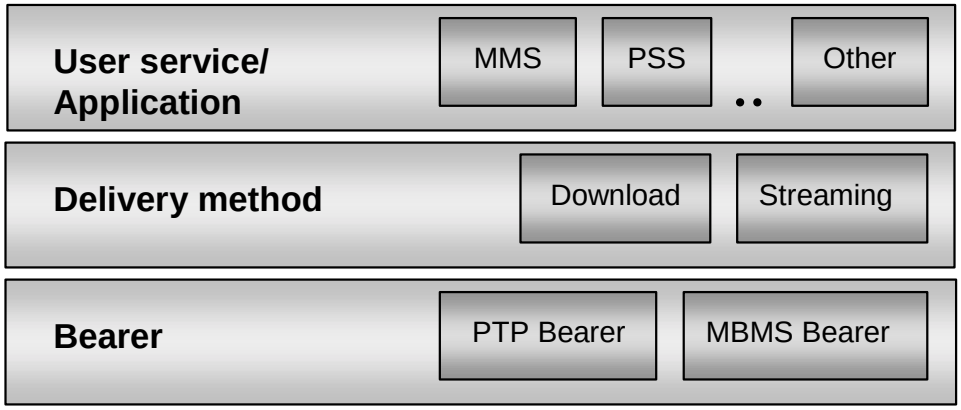


Figure 1: Functional Layers for MBMS User Service

- **Bearers**. Bearers provide the mechanism by which IP data is transported. MBMS bearers as defined in [4] (TS 23.246) and [3] (TS 22.146) 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 unicast post delivery supplementation, reception verification and support for inter-operator service profiles. Two delivery methods are defined, namely download and streaming. MBMS delivery may utilize both MBMS bearers and PTP bearers.

- **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.

4.2 MBMS User Service Entities

The figure below shows the MBMS user service entities and their inter-relations. Relation cardinality is depicted as well.

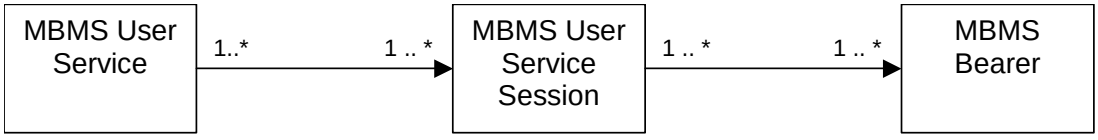


Figure 2: Entities and Relations

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.

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

4.3 MBMS Bearer Service Architecture

The MBMS Bearer Service Architecture is defined in [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 MBMS User Service Client description

This section describes the functional components of an MBMS User Service Client

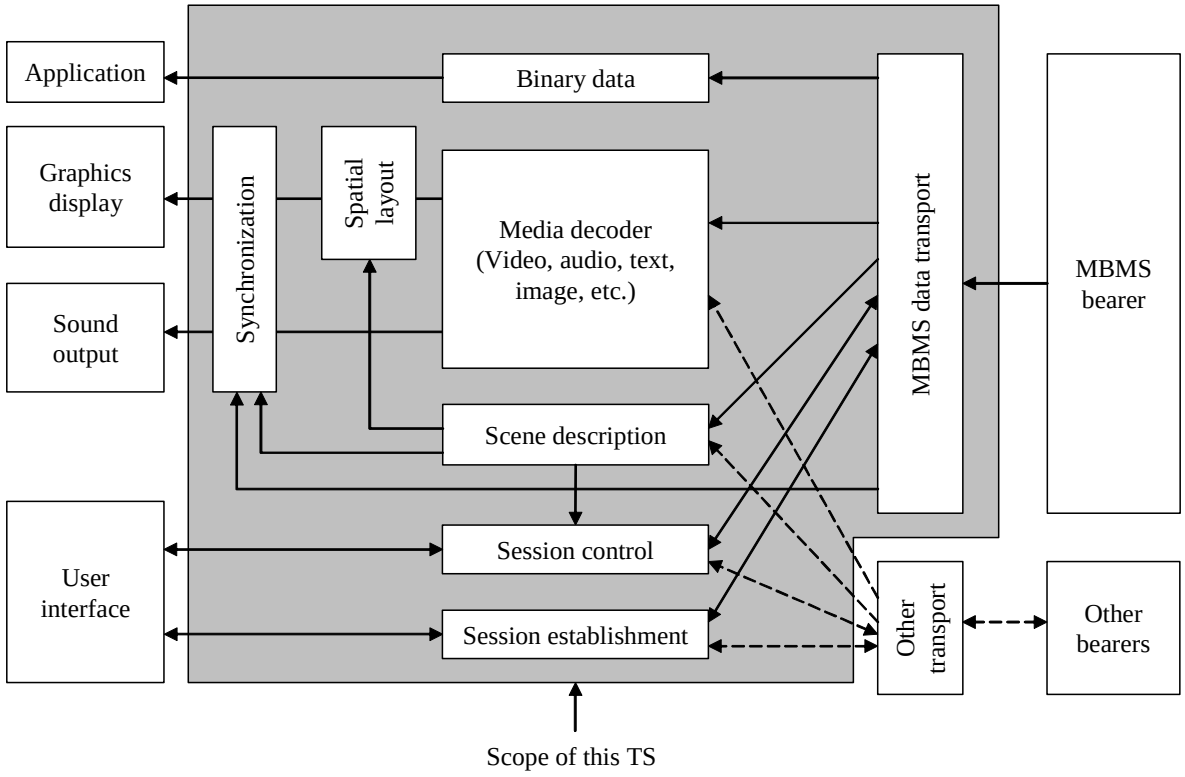


Figure 1: Functional components of an MBMS client

Figure 1 shows the functional components of an MBMS client. The components consist of session establishment, session control, scene description, media decoders, and MBMS data transport.

The session establishment refers to methods to invoke an MBMS session from a browser, or directly by tuning to a well-known session announcement channel. The session control deals with the set-up of the individual media streams between an MBMS client and MBMS servers.

The scene description consists of spatial layout and a description of the temporal relation between different media that is included in the media presentation. The first gives the layout of different media components on the screen and the latter controls the synchronisation of the different media (see clause 8).

The media decoders are the decoders for video, audio, text, still images, etc (see clause 7). The binary data that do not require synchronized presentation with other media is simply forwarded to the corresponding application.

The MBMS data transport provides delivery methods on the MBMS bearer. It may also provide MBMS Forward Error Correction (FEC) decoders.

Note that an MBMS client can utilize the other radio bearers with the MBMS bearer simultaneously or exclusively depending on its terminal capability. Thus, some of media objects, scene description, and presentation description file may be delivered by the other transports, e.g., HTTP/TCP, messaging, cell broadcast. However, these transports are out of scope of this document.

4.5 MBMS User Service Server description

The description of MBMS User Service Server is not in the scope of the standard.

5 Procedures and Protocol overview

This clause defines the procedures and protocols that the MBMS User Services should use.

5.1 Introduction

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 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, along with information on the user service, 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 section 5.3.

According to [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 section 5.2.2. Other user service announcement and discovery mechanisms by other means than the download delivery method are out of scope of the present specification.

User service announcement over a MBMS bearer

5.2.2.1 Introduction

MBMS Service Announcements are needed in order to advertise MBMS Streaming and MBMS Download User Services 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 section 6.1.

Service 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 [14].

The metadata consists of:

- a metadata envelope object(s) allowing the identification, versioning, update and temporal validity of a metadata fragment;
- a metadata fragment object(s) describing details of MBMS user services.

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.

5.2.2.2 Supported Metadata Syntaxes

The MBMS metadata syntax shall support 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).

5.2.2.3 Consistency control and Syntax Independence

The *Metadata Envelope* shall provide information to identify, version and expire metadata fragments. This shall be 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).

5.2.2.4 Metadata Envelope Definition

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

- *metadataURI*: A URI providing a unique identifier for the metadata fragment.
- *Version*: Current version number of the metadata fragment file. The version number is initially zero, and is increased by one whenever the metadata fragment version is changed.
- *validFrom*: The date and time from which the metadata fragment file is valid.
- *validUntil*: The date and time when the metadata fragment file expires.

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">
    <xs:complexType>
      <xs:sequence>
        <xs:any minOccurs="0" maxOccurs="unbounded"/>
      </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:anyAttribute processContents="skip"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

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.

5.2.2.5 Delivery of the Metadata Envelope

MBMS Service Announcement transports shall support delivery of the metadata envelope as a discrete object (XML file).

5.2.2.6 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).

5.2.2.7 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.

5.3 User service initiation/termination

5.3.1 Initiation

MBMS user service initiation refers to 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 user service activation consists of (not necessarily in this order):

- MBMS application initiation (this is outside the scope of this specification);
- MBMS service activation, as specified in CN1 specifications [Ref TBA];
- [Security Functions TBD].

The following metadata information are required to initiate the MBMS user service:

- [Service type: streaming, messaging etc. (to launch the right application in the terminal)];
- [broadcast or multicast mode];
- [security on/off and related parameters];
- [user service session start/stop time];
- [Port #, IP@, protocol];
- [media types and codecs];
- [QoS, data rates, UE MBMS bearer capability requirements, etc.];
- [FEC on/off, related parameters];
- [session identification];
- [content delivery verification on/off and related parameters]

5.3.2 Termination

5.4 Protocols

Figure 2 illustrates the protocol stack used by MBMS User services.

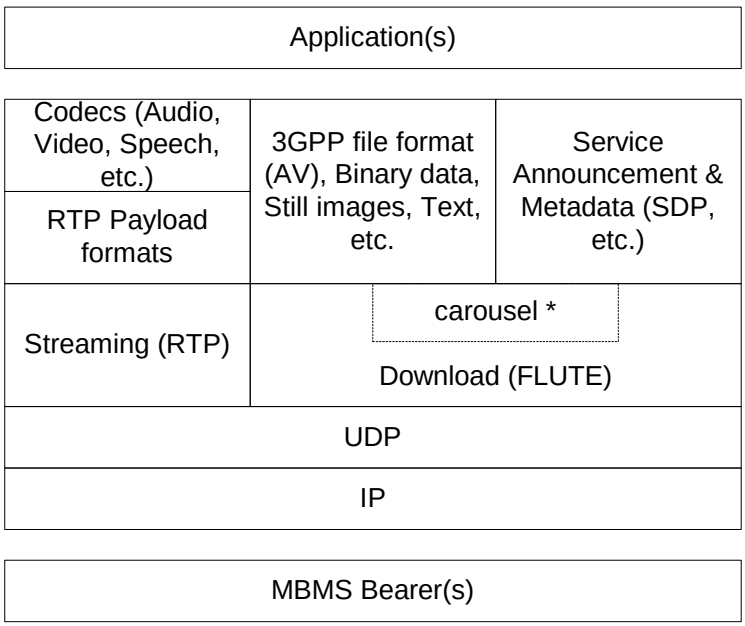


Figure 2: Protocol stack view of the MBMS User Services over MBMS bearer(s)

6 Delivery methods

6.1 Download delivery method

6.1.1 Introduction

MBMS download delivery method is based on the FLUTE protocol [9].

FLUTE is built on top of the Asynchronous Layered Coding (ALC) protocol instantiation [10]. ALC combines the Layered Coding Transport (LCT) building block [11], a congestion control building block and the Forward Error Correction (FEC) building block [12] to provide congestion controlled reliable asynchronous delivery of content to an unlimited number of concurrent receivers from a single sender. As mentioned in [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 3 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.

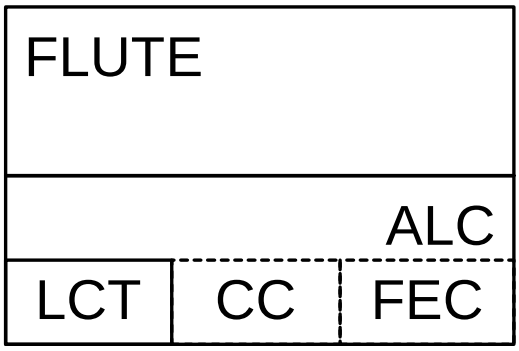


Figure 3: Building block structure of FLUTE

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.

6.1.2 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 this specification the term "file" is used for all objects carried by FLUTE (with the exception of the FDT Instances).

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

6.1.2.1 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).

Exactly one encoding symbol shall be carried in the payload of one FLUTE packet.

6.1.2.2 Symbol Encoding Algorithm

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

6.1.2.3 Blocking Algorithm

The "Algorithm for Computing Source Block Structure" described within the FLUTE specification [9] shall be used.

6.1.2.4 Congestion Control

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

6.1.2.5 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 payload ID shall be set according to [13] such that a 16 bit SBN (Source Block Number) and then the 16 bit ESI (Encoding Symbol ID) are given.

6.1.2.6 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.

6.1.2.7 Signalling of Parameters with FDT Instances

The FLUTE FDT Instance schema [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-OTI-Maximum-Source-Block-Length;
- FEC-OTI-Encoding-Symbol-Length;
- FEC-OTI-Max-Number-of-Encoding-Symbols;

NOTE: FLUTE [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).
- FEC-OTI-FEC-Instance-ID.

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

6.1.3 SDP for download delivery method

6.1.3.1 Introduction

The FLUTE specification [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 [ABNF reference].

The following download session parameters are required [9]:

- The sender IP address;
- The number of channels in the session;
- The destination IP address and port number for each channel in the session;
- The Transport Session Identifier (TSI) of the session;
- An indication of whether or not the session carries packets for more than one object. Note that no SDP attribute is needed for this (see clause 6.1.3.2.5)
- FEC Encoding ID and FEC Instance ID;

The following download session parameters are also proposed [9]:

- The start time and end time of the session;
- Some information that tells receiver, in the first place, that the session contains files that are of interest.

6.1.3.2 SDP Parameters for MBMS download session

The semantics of a Session Description of an MBMS download session shall include the 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;
- The Transport Session Identifier (TSI) of the session;
- The start time and end time of the session.

These shall be expressed in SDP [14, 15, 16] syntax according to the following clauses.

6.1.3.2.1 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:") [14, 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.

6.1.3.2.2 Number of channels

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

6.1.3.2.3 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 [14, 15]. The destination port number shall be defined according to the <port> sub-field of the media announcement field ("m=") of SDP [14, 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.

6.1.3.2.4 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 [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 [14]

6.1.3.2.5 Multiple objects transport indication

FLUTE [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.

6.1.3.2.6 Session Timing Parameters

A MBMS download session start and end times shall be defined according to the SDP timing field ("t=") [14, 15].

6.1.3.3 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=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9*
a=flute-tsi:3
a=flute-ch:1
m=application 12345 FLUTE/UDP 0
c=IN IP6 FF1E:03AD::7F2E:172A:1E24/1

6.2 Streaming delivery method

6.2.1 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.

6.2.2 The Data 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. The RTP payload formats and corresponding MIME types should be aligned to the ones defined in PSS Rel-6 [26.234].

RTP provides RTCP for feedback about the transmission quality. The transmission of RTCP packets in the downlink (sender reports) is allowed. In the context of MBMS 3GPP Rel-6, RTCP RR shall be turned off by SDP RR bandwidth modifiers [Ref]. Note that in the context of MBMS detection of link aliveness is not necessary.

6.2.3 Session description

SDP is provided to the MBMS client via a discovery/announcement procedure to describe the streaming delivery session.

6.2.3 Session control

6.3 Associated delivery procedures

6.3.1 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, the clauses 6.3.1 and 6.3.3 specify generic and extensible techniques for a potentially wide range of associated delivery procedures.

Clause 6.3.2 specifies the post-delivery procedures that are included in release 6, and are initiated only after an MBMS data transmission phase.

This specification describes these associated delivery procedures:

- File repair, for post-delivery repair of files initially delivered an MBMS download session

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.

When using a download delivery session to deliver download content, the UE shall support the file repair procedure.

6.3.1.1 The 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 shall take priority, such that configuration parameters received prior to, and out-of-band of, the download session they apply to are regarded as “global 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 a “procedure” element. All configuration parameters of one associated delivery procedure are contained as attributes of a “procedure” element. The “label”

attribute of a "procedure" element identifies the associated procedure to configure. The associated delivery procedure description is specified formally as an XML schema below.

6.3.2 Post-delivery Procedures

6.3.2.1 File Repair Procedure

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.
- 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 missing data from an MBMS download
2. Calculates a random *back-off time* and selects a server randomly out of a list
3. Sends a *repair request* message to the selected server at the calculated time.

Then the server

1. Responds with a *repair response* message either containing the requested data, or alternatively, describing an error case.

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.

6.3.2.1.1 Identification of Missing Data from an MBMS Download

6.3.2.1.2 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.

6.3.2.1.2.1 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.

6.3.2.1.2.2 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*.

6.3.2.1.2.3 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.

6.3.2.1.3 File Repair Server Selection

6.3.2.1.3.1 List of Base-URIs

A list of file repair servers is provided by a list of base-URIs as attributes of the associated delivery procedure description. These attributes and elements specify the base URIs of the point-to-point repair servers. Base-URIs may also be given as IP addresses, which may be used to avoid a requirement for DNS messaging. The base-URIs of a single file repair configuration file 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 "baseURI" elements, each of which shall be a child-element of the "procedure" element. The "baseURI" element provides the references to the file repair server via the "server" attribute. At least one "baseURI" element shall be present.

6.3.2.1.3.2 Selection from the Base-URI List

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

6.3.2.1.4 File Repair Request Message

Once missing file data is identified, the MBMS client sends one or more messages to a repair server requesting transmission of data that allows recovery of missing file data. All point-to-point repair requests and repair responses for a particular MBMS transmission shall take place in a single TCP session using the HTTP protocol [REFERENCE TO HTTP 1.1].

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

6.3.2.1.4.1 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. Each missing packet is uniquely identified by the encoding symbol 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). The (SBN, ESI) pair uniquely identifies a FLUTE encoding symbol which allows recovery of missing file data belonging to the above-mentioned file. 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 [1] request method GET. The (SBN, ESI) of requested encoding symbols are URL-encoded [19] and included in the HTTP GET request.

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 [1]:

```
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 )

sbn_range = ( sbnA [ "-" sbnZ ] ) / ( sbnA [ ";" esi_info] )

esi_info = *( "ESI=" esi_range *( "," esi_range ) )

esi_range = esiA [ "-" esiZ ]

sbnA = %d     ; the SBN, or the first of a range of SBNs

sbnZ = %d     ; the last SBN of a range of SBNs

esiA = %d     ; the ESI, or the first of a range of SBNs

esiZ = %d     ; the last ESI of a range of SBNs
```

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 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 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)

6.3.2.1.5 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.

The set of encoding symbols of a repair response message shall form the file repair response payload, which shall be an HTTP payload as specified in the following clause.

6.3.2.1.5.1 File Repair Response Messages Codes

In the case that the MBMS 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 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).

Other HTTP status codes [18] shall be used to support other cases. Other cases may include server errors, client errors (in the file repair request message), server overload and redirection to other MBMS file repair servers.

6.3.2.1.5.2 File Repair Response Message Format for 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 [18] may also be used but are not mandated by this mechanism.

Each encoding symbol of the file repair response payload shall be preceded by its FEC Payload ID. The FEC Payload ID is specified in below. The file repair response payload is constructed by including each FEC Payload ID and Encoding Symbol pair 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.

The UE and MBMS 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 exception of the last source encoding symbol of the last source block where symmetric FEC instance is used. All FEC Payload IDs are the same length for one file repair request-response as a single FEC Instance is used for a single file.

The FEC Payload ID shall consist of the encoding symbol SBN and ESI in big-endian order, with SBN occupying the most significant portion. The length, in bytes, of SBN and ESI field are specified by FEC Instance (or FEC Encoding and Instance combination) as, for example, in FEC Encoding ID 0 [13] for compact no-code FEC.

Figure 3 exemplifies the construction of the FEC Payload ID in the case of FEC Encoding ID 0.

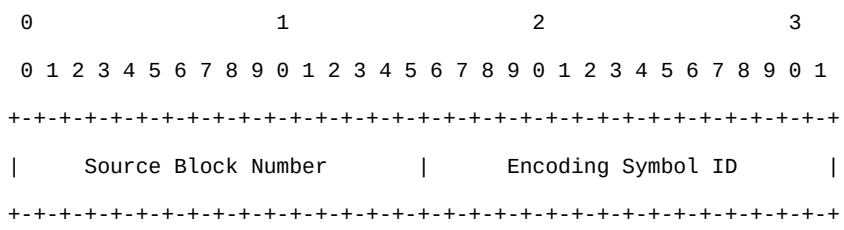


Figure 3 Example FEC Payload ID Format

Figure 4 illustrates the complete file repair response message format (box sizes are not indicative of the relative lengths of the labelled entities).

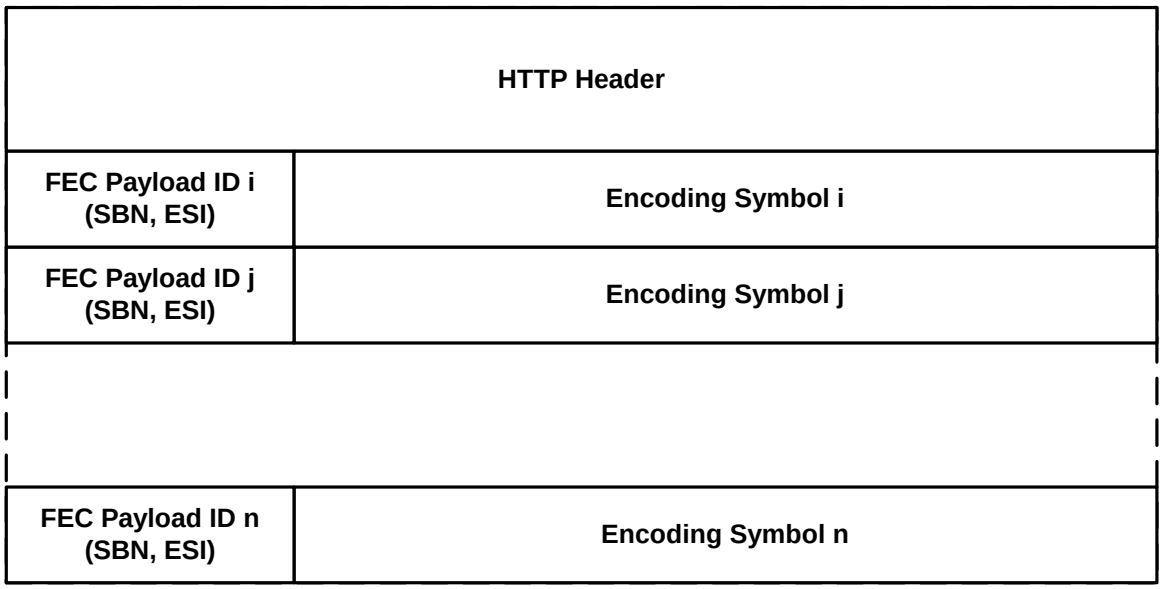


Figure 4 File Repair Response Message Format

6.3.2.1.6 Server Not Responsing Error Case

In the error case where a UE determines that the its selected repair server is not responding it shall return to the base-URI 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 base-uri 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 unrecognised message (not a recognisable HTTP response)
4. The server returns an HTTP server error status code (in the range 500-505)

6.3.2.2 Delivery confirmation procedure

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

The UE shall be able to confirm the reception and successful delivery of content. If the BM-SC provided parameters require delivery confirmation then the UE must confirm the content reception. The delivery confirmation procedure may be used for reception-based charging.

Using the parameters provide by the BM-SC, the UE randomises the time at which the delivery confirmation procedure is initiated. This can be used to spread the load of numerous confirmations from multiple MBMS receiving UEs.

If delivery confirmation is requested for statistical purposes the BM-SC may provide confirmation probability parameters setting the probability at which a UE would confirm reception. The BM-SC provides the probability range parameter and a confirmation probability parameter. The UE generates a random number within the probability range. If the generated random number is smaller than the confirmation probability than the UE confirms delivery. If the randomly generated number is larger than or equal to the confirmation probability than the UE does not confirm reception.

6.3.2.1 Signalling Delivery Confirmation Parameters

6.3.2.1 The Delivery Confirmation Procedure

6.3.3 Extensible xml-Schema for Associated Delivery Procedures

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="mbms-associated-procedure-description-instance">

      <xs:complexType>

         <xs:sequence>

            <xs:element name="procedure" minOccurs="1" maxOccurs="unbounded">
   <xs:complexType>

                  <xs:sequence>

                     <xs:element name="baseURI" minOccurs="1" maxOccurs="unbounded">

                        <xs:complexType>

                           <xs:attribute name="server" type="xs:anyURI" use="required"/>

                        </xs:complexType>

                     </xs:element>

                  </xs:sequence>

                  <xs:attribute name="label" type="xs:string" use="required"/>

                  <xs:attribute name="wait-time" type="xs:unsignedLong" use="required"/>

                  <xs:attribute name="max-back-off" type="xs:unsignedLong" use="required"/>

                  <xs:anyAttribute processContents="skip"/>

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

6.3.3.1 Enumerations

The “Server” attribute (type “xs:anyURI”) shall only take on base URI values and not give the URI resource part.

“Label” value = {“PostFileRepair”}

6.3.3.2 Example Associated Delivery Procedure Description Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<mbms-associated-procedure-description-instance

                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

                  xsi:schemaLocation="http://www.example.com/mbms-associated-descrition.xsd">
   <procedure label="PostFileRepair" offset-time="5" random-time-period="10">   <baseURI
server="http://mbmsrepair.operator.umts/"/> <baseURI server="http://mbmsrepair1.operator.umts/"/>
   <baseURI server="http://mbmsrepair2.operator.umts/"/></procedure>

</mbms-associated-procedure-description-instance>
```

7 Codecs

7.1 General

The set of media decoders that are supported by the MBMS Client to support a particular media type are defined below. These media decoders and media types are common to all MBMS User Services. E.g. common to Streaming and Download.

7.2 Speech

7.3 Audio

7.4 Video

7.5 Still images

7.6 Text

7.7 3GPP file format

8 Scene Description

The scene description consists of spatial layout and a description of the temporal relation between different media that is included in the media presentation. The first gives the layout of different media components on the screen and the latter controls the synchronisation of the different media (see clause 8).

9 Security aspects

Annex <A> (informative): QoS consideration

Annex <B> (normative): FLUTE Support Requirements

This clause provides a table representation of the requirement levels for different features in FLUTE. Table 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 [9]. The terms used in Table 1 are described underneath.

Table 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. Other FEC building blocks are undefined optional plug-ins.	Compact No-Code algorithm required. [TBD]	Compact No-Code algorithm is the default option. Other FEC building blocks are undefined optional plug-ins.	Compact No-Code algorithm is the default option. [TBD]
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	Not applicable	Optional	Not applicable
A flag active (header)	Required	Required	Optional	Optional
B flag active (header)	Required	Required	Optional	Optional
T flag active and SCT field (header)	Optional	Optional	Optional	Optional
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]

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 [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.

Annex C (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
09-2004	25				Version 1.0.0		1.0.0

3GPP TS 26.346 V0.0.5 (2004-09)

Technical Specification

Editor's Note: Information for the reader

3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs (Release 6)

TSG-SA4 PSM SWG internal working draft

The present document is the TSG-SA4 PSM SWG internal working draft of TS 26.346 Release 6.

- MS Word change marks are used to indicate changes between successive version of the draft (e.g.. the difference between version 0.0.0 and 0.1.0).
- Editor's notes are just for information during the drafting and will not be part of the final Release 6 of TS 26.346. They may, e.g., be used to describe the current status of the work in the PSM subgroup.
- Text highlighted with yellow is used to give important information to the reader. Yellow text will not be part of the final Release-6 document.

UMTS, IP, packet mode, protocol, codec, MBMS

Keywords

<keyword[, keyword]>

3GPP

Postal address

3GPP support office address

650 Route des Lucioles - Sophia Antipolis
Valbonne - FRANCE
Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Internet

http://www.3gpp.org

Contents

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. This 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 this document is the definition of a set of media codecs, formats and transport/application protocols to enable the deployment of MBMS user services. This specification 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. This specification 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: “MBMS User Services”.

[4] 3GPP TS 23.246: “MBMS Architecture and Functional description (Stage 2)”.

[5] 3GPP TS 25.346: “MBMS in the Radio Access Network; Stage 2”.

[6] IETF RFC 3550: "RTP: A Transport Protocol for Real-Time Applications", Schulzrinne H. et al., July 2003.

[7] IETF STD 0006: "User Datagram Protocol", Postel J., August 1980.

[8] IP [TBA]

[9] FLUTE – File Delivery over Unidirectional Transport, IETF Internet Draft, Work in progress, http://www.ietf.org/internet-drafts/draft-ietf-rmt-flute-07.txt.

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

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

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

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

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

[15] Session Description Protocol (SDP) Source Filters - IETF Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-mmusic-sdp-srcfilter-05.txt

[16] IETF RFC 3266: "Support for IPv6 in Session Description Protocol (SDP)", S. Olson, G. Camarillo, A. B. Roach, June 2002.

[17] IETF RFC 3048: "Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer", B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby, January 2001.

[18] IETF RFC 2616: " Hypertext Transfer Protocol -- HTTP/1.1"

[19] IETF RFC 1738: "Uniform Resource Locators (URL)"

[TBA…]

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the definitions in 3GPP TR 21.905 [1] as well as the following definitions apply.

Broadcast session: See TS 22.146: “Multimedia Broadcast/Multicast Service” [2].

Multicast joining: See TS 22.146: “Multimedia Broadcast/Multicast Service” [2].

Multicast session: See TS 22.146: “Multimedia Broadcast/Multicast Service” [2].

Multimedia Broadcast/Multicast Service (MBMS): See TS 22.146: “Multimedia Broadcast/Multicast Service” [2].

MBMS user services: See TS 22.246: “Multimedia Broadcast/Multicast Service: User Service” [3]. MBMS User Service may use more than one Multimedia Broadcast/Multicast Service (bearer service) and more than one Broadcast and/or Multicast session.

MBMS user service discovery/announcement: The 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. 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: A mechanism used by a MBMS user service to deliver content.

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

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

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

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

FLUTE channel: [TBA]

[Editor's note: SA4 terminology should be aligned with SA2 terminology]

3.2 Abbreviations

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

ALC	Asynchronous Layered Coding
BM-SC	Broadcast-Multicast Service Centre
CC	Congestion Control
ERT	Expected Residual Time
ESI	Encoding Symbol ID
GGSN	Gateway GPRS Serving Node
GPRS	General Packet Radio Service
FDT	File Delivery Table
FEC	Forward Error Correction
FLUTE	File deLivery over Unidirectional Transport
IP	Internet Protocol
LCT	Layered Coding Transport
MBMS	Multimedia Broadcast/Multicast Service
MS	Mobile Station
RTP	Real-Time Transport Protocol
SBN	Source Block Number
SDP	Session Description Protocol
SCT	Sender Current TimeTOI Transport Object Identifier
TSI	Transport Session Identifier
UDP	User Datagram Protocol
UE	User Equipment
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.

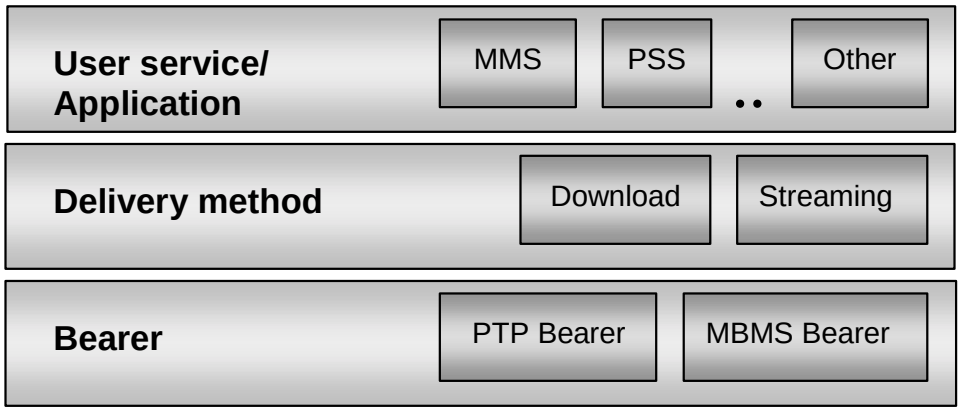


Figure 1: Functional Layers for MBMS User Service

- **Bearers**. Bearers provide the mechanism by which IP data is transported. MBMS bearers as defined in [4] (TS 23.246) and [3] (TS 22.146) 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 unicast post delivery supplementation, reception verification

and support for inter-operator service profiles. Two delivery methods are defined, namely download and streaming. MBMS delivery may utilize both MBMS bearers and PTP bearers.

– **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.

4.2 MBMS User Service Entities

The figure below 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.

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

4.3 MBMS Bearer Service Architecture

The MBMS Bearer Service Architecture is defined in [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.

[Editor's note: it is the intention of the PSM SWG to describe the MBMS User Service architecture in this clause]

4.4 MBMS User Service Client description

This section describes the functional components of an MBMS User Service Client

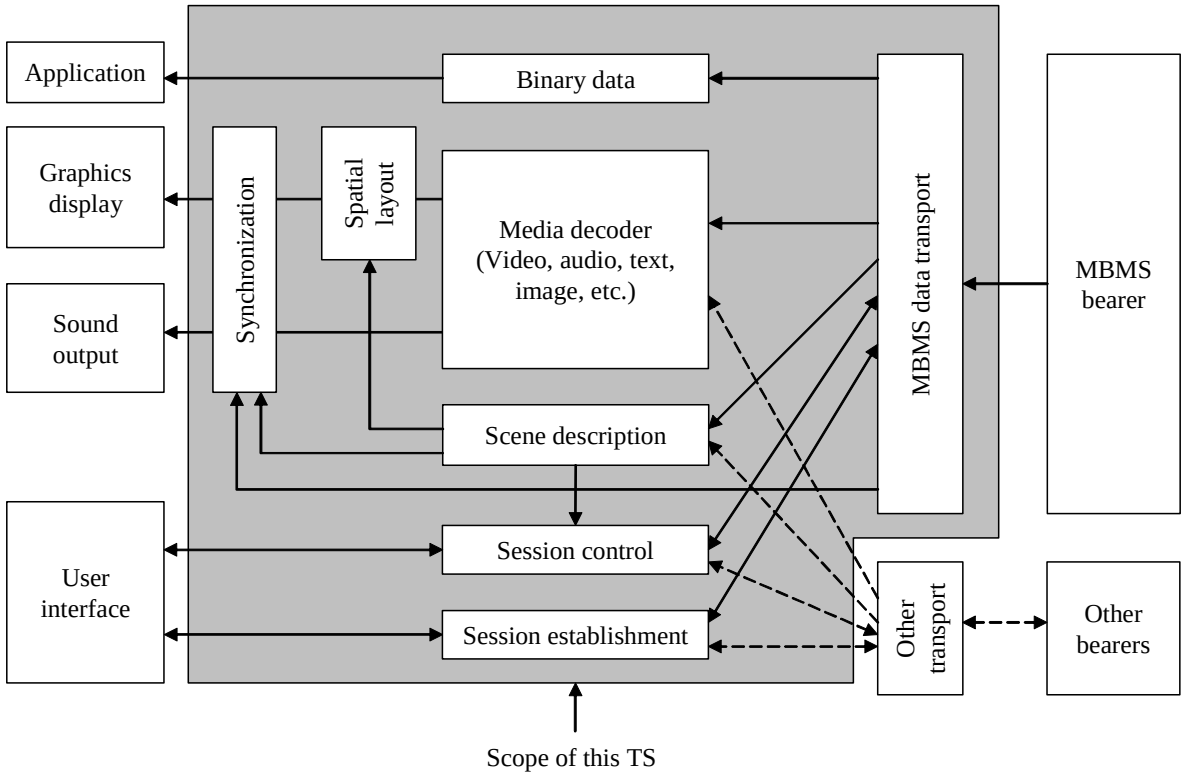


Figure 1: Functional components of an MBMS client

[Editor's note: this figure needs updating]

Figure 1 shows the functional components of an MBMS client. The components consist of session establishment, session control, scene description, media decoders, and MBMS data transport.

The session establishment refers to methods to invoke an MBMS session from a browser, or directly by tuning to a well-known session announcement channel. The session control deals with the set-up of the individual media streams between an MBMS client and MBMS servers.

The scene description consists of spatial layout and a description of the temporal relation between different media that is included in the media presentation. The first gives the layout of different media components on the screen and the latter controls the synchronisation of the different media (see clause 8).

The media decoders are the decoders for video, audio, text, still images, etc (see clause 7). The binary data that do not require synchronized presentation with other media is simply forwarded to the corresponding application.

The MBMS data transport provides delivery methods on the MBMS bearer. It may also provide MBMS Forward Error Correction (FEC) decoders.

Note that an MBMS client can utilize the other radio bearers with the MBMS bearer simultaneously or exclusively depending on its terminal capability. Thus, some of media objects, scene description, and presentation description file may be delivered by the other transports, e.g., HTTP/TCP, messaging, cell broadcast. However, these transports are out of scope of this document.

4.5 MBMS User Service Server description

The description of MBMS User Service Server is not in the scope of the standard.

5 Procedures and Protocol overview

This clause defines the procedures and protocols that the MBMS User Services should use.

[Editor's note: it is the intention of the editor that this clause should introduce the "stage 2" description of MBMS procedures and protocols while other clauses (e.g. section 6) would contain the "Stage 3" type of text relating to these procedures and protocols]

5.1 Introduction

[Editor's note: this section should introduce the MBMS 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 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, along with information on the user service, 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 section 5.3.

According to [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 section 5.2.2. Other user service announcement and discovery mechanisms by other means than the download delivery method are out of scope of the present specification.

[Editor's Note: Examples of other possible discovery/announcement mechanisms descriptions could be added in the MBMS TR and referenced in that section]

5.2.2 User service announcement over a MBMS bearer

5.2.2.1 Introduction

MBMS Service Announcements are needed in order to advertise MBMS Streaming and MBMS Download User Services 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 section 6.1.

Service 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 [14].

The metadata consists of:

- a metadata envelope object(s) allowing the identification, versioning, update and temporal validity of a metadata fragment;
- a metadata fragment object(s) describing details of MBMS user services.

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.

[Editor's note:" it **shall** be possible to support multiple metadata fragment syntaxes" i.e. the metadata envelope enables this]

5.2.2.2 Supported Metadata Syntaxes

The MBMS metadata syntax shall support 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).

5.2.2.3 Consistency control and Syntax Independence

The *Metadata Envelope* shall provide information to identify, version and expire metadata fragments. This shall be 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).

5.2.2.4 Metadata Envelope Definition

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

- *metadataURI*: A URI providing a unique identifier for the metadata fragment.
- *Version*: Current version number of the metadata fragment file. The version number is initially zero, and is increased by one whenever the metadata fragment version is changed.
- *validFrom*: The date and time from which the metadata fragment file is valid.
- *validUntil*: The date and time when the metadata fragment file expires.

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">
    <xs:complexType>
      <xs:sequence>
        <xs:any minOccurs="0" maxOccurs="unbounded"/>
      </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:anyAttribute processContents="skip"/>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

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.

5.2.2.5 Delivery of the Metadata Envelope

MBMS Service Announcement transports shall support delivery of the metadata envelope as a discrete object (XML file).

5.2.2.6 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).

5.2.2.7 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.

[Editor's note: clause 5.2.2 is still under discussion]

5.3 User service initiation/termination

5.3.1 Initiation

MBMS user service initiation refers to 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 user service activation consists of (not necessarily in this order):

- MBMS application initiation (this is outside the scope of this specification);
- MBMS service activation, as specified in CN1 specifications [Ref TBA];
- [Security Functions TBD].

[Editor's Note: Security functions may already be included in MBMS service activation procedures]

[Editor's Note: Some more information is required on the MBMS service subscription to finalize the MBMS user service initiation section]

The following metadata information are required to initiate the MBMS user service:

- [Service type: streaming, messaging etc. (to launch the right application in the terminal)];
- [broadcast or multicast mode];
- [security on/off and related parameters];
- [user service session start/stop time];
- [Port #, IP@, protocol];
- [media types and codecs];
- [QoS, data rates, UE MBMS bearer capability requirements, etc.];
- [FEC on/off, related parameters];
- [session identification];
- [content delivery verification on/off and related parameters]

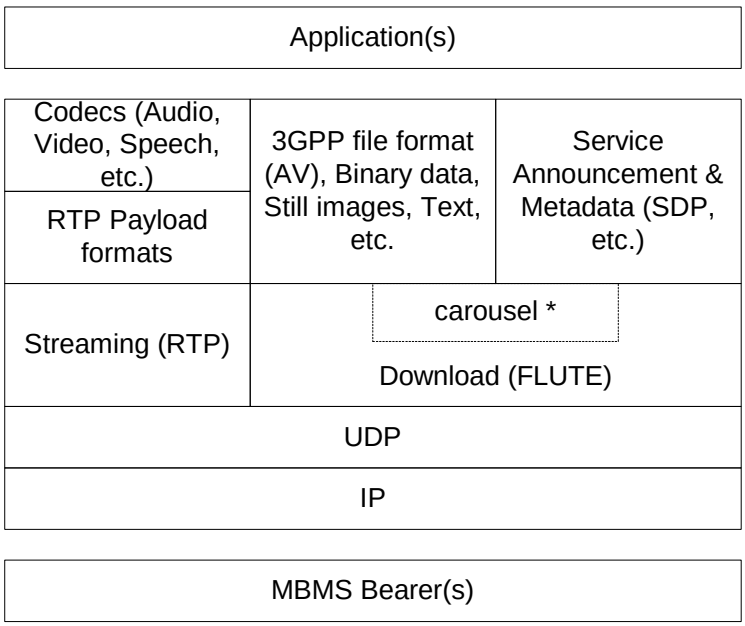
[Editor's Note: the above list is for further study]

5.3.2 Termination

[Editor's Note: this section should describe user service termination]

5.4 Protocols

Figure 2 illustrates the protocol stack used by MBMS User services.



[* Editor's Note: The inclusion of carousel is pending further contribution.]

Figure 2: Protocol stack view of the MBMS User Services over MBMS bearer(s)

6 Delivery methods

6.1 Download delivery method

6.1.1 Introduction

MBMS download delivery method is based on the FLUTE protocol [9].

FLUTE is built on top of the Asynchronous Layered Coding (ALC) protocol instantiation [10]. ALC combines the Layered Coding Transport (LCT) building block [11], a congestion control building block and the Forward Error Correction (FEC) building block [12] to provide congestion controlled reliable asynchronous delivery of content to an unlimited number of concurrent receivers from a single sender. As mentioned in [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 3 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.

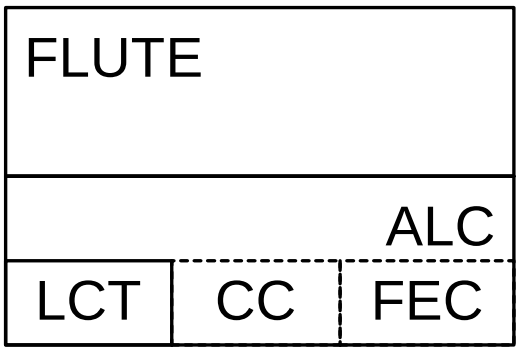


Figure 3: Building block structure of FLUTE

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.

6.1.2 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 this specification the term "file" is used for all objects carried by FLUTE (with the exception of the FDT Instances).

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

6.1.2.1 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).

Exactly one encoding symbol shall be carried in the payload of one FLUTE packet.

[Editor's note: The limitation of FLUTE packet payload length is still under discussion]

6.1.2.2 Symbol Encoding Algorithm

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

[Editor's note: the support of any other symbol encoding scheme is still under discussion]

6.1.2.3 Blocking Algorithm

The "Algorithm for Computing Source Block Structure" described within the FLUTE specification [9] shall be used.

6.1.2.4 Congestion Control

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

6.1.2.5 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 payload ID shall be set according to [13] such that a 16 bit SBN (Source Block Number) and then the 16 bit ESI (Encoding Symbol ID) are given.

[Editor's note: This section is still under discussion]

6.1.2.6 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.

[Editor's note: This section is still under discussion]

6.1.2.7 Signalling of Parameters with FDT Instances

The FLUTE FDT Instance schema [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-OTI-Maximum-Source-Block-Length;
- FEC-OTI-Encoding-Symbol-Length;
- FEC-OTI-Max-Number-of-Encoding-Symbols;

NOTE: FLUTE [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).
- FEC-OTI-FEC-Instance-ID.

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

[Editor's note: This section is still under discussion.]

[Editor's note: There may be other FDT extension parameters to include]

6.1.3 SDP for download delivery method

6.1.3.1 Introduction

The FLUTE specification [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 [ABNF reference].

The following download session parameters are required [9]:

- The sender IP address;
- The number of channels in the session;

- The destination IP address and port number for each channel in the session;
- The Transport Session Identifier (TSI) of the session;
- An indication of whether or not the session carries packets for more than one object. Note that no SDP attribute is needed for this (see clause 6.1.3.2.5)
- FEC Encoding ID and FEC Instance ID;

[Editor's note: The purpose of putting the "FEC Encoding ID and FEC Instance ID" is to indicate the required capabilities to the UE for the download delivery session. The exact parameters are still under discussion]

The following download session parameters are also proposed [9]:

- The start time and end time of the session;
- Some information that tells receiver, in the first place, that the session contains files that are of interest.

6.1.3.2 SDP Parameters for MBMS download session

The semantics of a Session Description of an MBMS download session shall include the 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;
- The Transport Session Identifier (TSI) of the session;
- The start time and end time of the session.

These shall be expressed in SDP [14, 15, 16] syntax according to the following clauses.

6.1.3.2.1 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:") [14, 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.

6.1.3.2.2 Number of channels

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

6.1.3.2.3 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 [14, 15]. The destination port number shall be defined according to the <port> sub-field of the media announcement field ("m=") of SDP [14, 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

[Editor's note: the FLUTE/UDP protocol id is still under discussion in IETF.]

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.

6.1.3.2.4 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 [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 [14]

6.1.3.2.5 Multiple objects transport indication

FLUTE [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.

6.1.3.2.6 Session Timing Parameters

A MBMS download session start and end times shall be defined according to the SDP timing field ("t=") [14, 15].

6.1.3.3 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=source-filter: incl IN IP6 * 2001:210:1:2:240:96FF:FE25:8EC9*
a=flute-tsi:3
a=flute-ch:1
m=application 12345 FLUTE/UDP 0
c=IN IP6 FF1E:03AD::7F2E:172A:1E24/1

6.2 Streaming delivery method

6.2.1 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.

[Editor's note: a figure should be added here describing the streaming delivery method protocol stack]

6.2.2 The Data 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. The RTP payload formats and corresponding MIME types should be aligned to the ones defined in PSS Rel-6 [26.234].

[Editor's note: FEC payload format is FFS. RTP for the transport of FEC blocks should be added when agreed. The use of SRTP is FFS and will be discussed with SA3]

RTP provides RTCP for feedback about the transmission quality. The transmission of RTCP packets in the downlink (sender reports) is allowed. In the context of MBMS 3GPP Rel-6, RTCP RR shall be turned off by SDP RR bandwidth modifiers [Ref]. Note that in the context of MBMS detection of link aliveness is not necessary.

[Editor's note: for Rel-6 it was agreed that RTCP feedback could be useful in future releases]

6.2.3 Session description

SDP is provided to the MBMS client via a discovery/announcement procedure to describe the streaming delivery session.

6.2.3 Session control

[Editor's note: the need or not for a session control (like e.g. RTSP) is currently not agreed]

[Editor's note: reporting of QoE metrics is felt useful by PSM and is FFS]

6.3 Associated delivery procedures

[Editor's note: This clause should specify the associated delivery procedures e.g. post-delivery procedures like ptp repair and content reception verification/reporting]

6.3.1 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, the clauses 6.3.1 and 6.3.3 specify generic and extensible techniques for a potentially wide range of associated delivery procedures.

Clause 6.3.2 specifies the post-delivery procedures that are included in release 6, and are initiated only after an MBMS data transmission phase.

This specification describes these associated delivery procedures:

- File repair, for post-delivery repair of files initially delivered an MBMS download session
- [Editor's note: Content reception verification/reporting is to be discussed in SA4]

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.

When using a download delivery session to deliver download content, the UE shall support the file repair procedure.

6.3.1.1 The 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 shall take priority, such that configuration parameters received prior to, and out-of-band of, the download session they apply to are regarded as "global 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 a "procedure" element. All configuration parameters of one associated delivery procedure are contained as attributes of a "procedure" element. The "label" attribute of a "procedure" element identifies the associated procedure to configure. The associated delivery procedure description is specified formally as an XML schema below.

6.3.2 Post-delivery Procedures

6.3.2.1 File Repair Procedure

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.
- 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 missing data from an MBMS download
2. Calculates a random *back-off time* and selects a server randomly out of a list
3. Sends a *repair request* message to the selected server at the calculated time.

Then the server

1. Responds with a *repair response* message either containing the requested data, or alternatively, describing an error case.

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.

6.3.2.1.1 Identification of Missing Data from an MBMS Download

[Editor's note: Some short text needed about identification of which symbols, blocks and files are corrupted and which it wants to fill the gaps. This should include the distinction between missing file data, missing encoding symbols and encoding symbols transmitted with repair data.]

6.3.2.1.2 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.

6.3.2.1.2.1 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.

6.3.2.1.2.2 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*.

6.3.2.1.2.3 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.

6.3.2.1.3 File Repair Server Selection

6.3.2.1.3.1 List of Base-URIs

A list of file repair servers is provided by a list of base-URIs as attributes of the associated delivery procedure description. These attributes and elements specify the base URIs of the point-to-point repair servers. Base-URIs may also be given as IP addresses, which may be used to avoid a requirement for DNS messaging. The base-URIs of a single file repair configuration file 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 “baseURI” elements, each of which shall be a child-element of the “procedure” element. The “baseURI” element provides the references to the file repair server via the “server” attribute. At least one “baseURI” element shall be present.

6.3.2.1.3.2 Selection from the Base-URI List

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

6.3.2.1.4 File Repair Request Message

Once missing file data is identified, the MBMS client sends one or more messages to a repair server requesting transmission of data that allows recovery of missing file data. All point-to-point repair requests and repair responses for a particular MBMS transmission shall take place in a single TCP session using the HTTP protocol [REFERENCE TO HTTP 1.1].

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

6.3.2.1.4.1 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. Each missing packet is uniquely identified by the encoding symbol 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). The (SBN, ESI) pair uniquely identifies a FLUTE encoding symbol which allows recovery of missing file data belonging to the above-mentioned file. 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 [1] request method GET. The (SBN, ESI) of requested encoding symbols are URL-encoded [19] and included in the HTTP GET request.

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 [1]:

```
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 )

sbn_range = ( sbnA [ "-" sbnZ ] ) / ( sbnA [ ";" esi_info] )

esi_info = *( "ESI=" esi_range *( "," esi_range ) )

esi_range = esiA [ "-" esiZ ]

sbnA = %d     ; the SBN, or the first of a range of SBNs

sbnZ = %d     ; the last SBN of a range of SBNs

esiA = %d     ; the ESI, or the first of a range of SBNs

esiZ = %d     ; the last ESI of a range of SBNs
```

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 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 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)

[Editor's note: future applications may define a new syntax for the query]

6.3.2.1.5 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.

The set of encoding symbols of a repair response message shall form the file repair response payload, which shall be an HTTP payload as specified in the following clause.

6.3.2.1.5.1 File Repair Response Messages Codes

In the case that the MBMS 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 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).

Other HTTP status codes [18] shall be used to support other cases. Other cases may include server errors, client errors (in the file repair request message), server overload and redirection to other MBMS file repair servers.

6.3.2.1.5.2 File Repair Response Message Format for 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.

[Editor's note: this Content-Type shall be IANA registered and subsequently also referenced]

This header is as follows:

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

Note, other HTTP headers [18] may also be used but are not mandated by this mechanism.

Each encoding symbol of the file repair response payload shall be preceded by its FEC Payload ID. The FEC Payload ID is specified in below. The file repair response payload is constructed by including each FEC Payload ID and Encoding Symbol pair 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.

The UE and MBMS 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 exception of the last source encoding symbol of the last source block where symmetric FEC instance is used. All FEC Payload IDs are the same length for one file repair request-response as a single FEC Instance is used for a single file.

The FEC Payload ID shall consist of the encoding symbol SBN and ESI in big-endian order, with SBN occupying the most significant portion. The length, in bytes, of SBN and ESI field are specified by FEC Instance (or FEC Encoding and Instance combination) as, for example, in FEC Encoding ID 0 [13] for compact no-code FEC.

Figure 3 exemplifies the construction of the FEC Payload ID in the case of FEC Encoding ID 0.

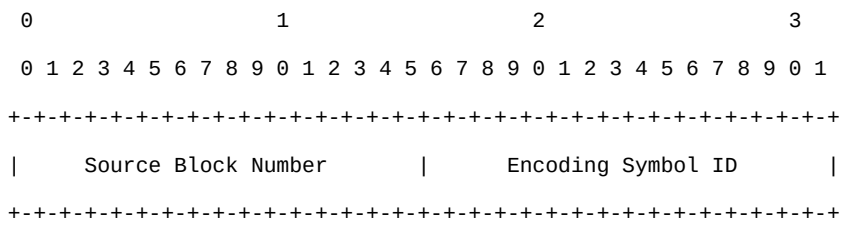


Figure 3 Example FEC Payload ID Format

Figure 4 illustrates the complete file repair response message format (box sizes are not indicative of the relative lengths of the labelled entities).

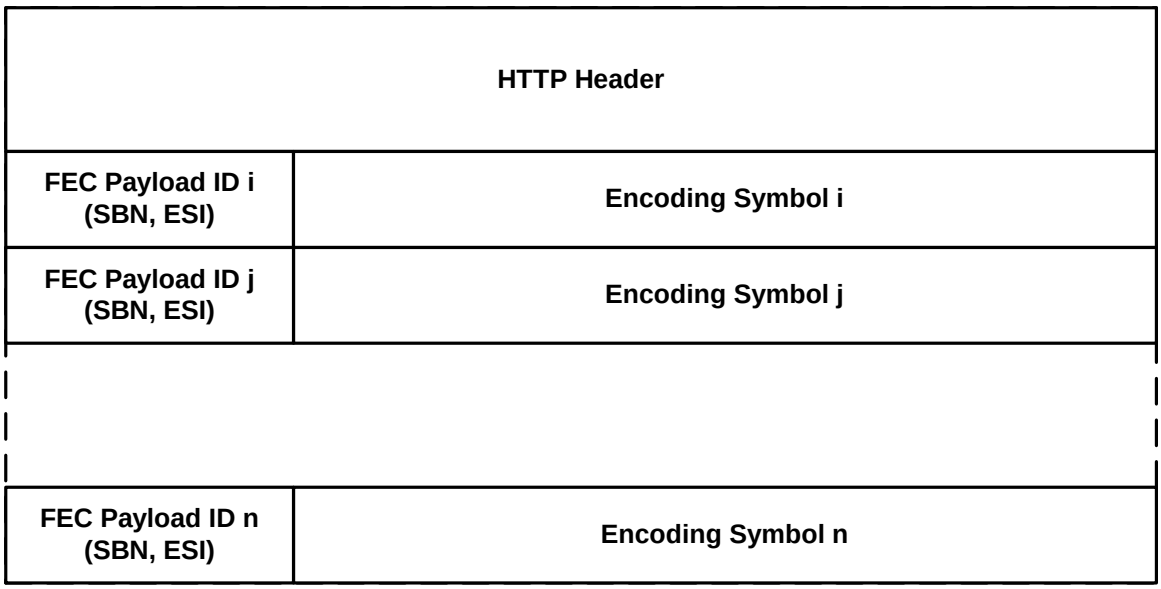


Figure 4 File Repair Response Message Format

6.3.2.1.6 Server Not Responsing Error Case

In the error case where a UE determines that the its selected repair server is not responding it shall return to the base-URI 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 base-uri 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 unrecognised message (not a recognisable HTTP response)
4. The server returns an HTTP server error status code (in the range 500-505)

6.3.2.2 Delivery confirmation procedure

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

The UE shall be able to confirm the reception and successful delivery of content. If the BM-SC provided parameters require delivery confirmation then the UE must confirm the content reception. The delivery confirmation procedure may be used for reception-based charging.

Using the parameters provide by the BM-SC, the UE randomises the time at which the delivery confirmation procedure is initiated. This can be used to spread the load of numerous confirmations from multiple MBMS receiving UEs.

If delivery confirmation is requested for statistical purposes the BM-SC may provide confirmation probability parameters setting the probability at which a UE would confirm reception. The BM-SC provides the probability range parameter and a confirmation probability parameter. The UE generates a random number within the probability range. If the generated random number is smaller than the confirmation probability than the UE confirms delivery. If the randomly generated number is larger than or equal to the confirmation probability than the UE does not confirm reception.

6.3.2.1 Signalling Delivery Confirmation Parameters

6.3.2.1 The Delivery Confirmation Procedure

6.3.3 Extensible xml-Schema for Associated Delivery Procedures

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="mbms-associated-procedure-description-instance">

      <xs:complexType>

         <xs:sequence>

            <xs:element name="procedure" minOccurs="1" maxOccurs="unbounded">
   <xs:complexType>

                  <xs:sequence>

                     <xs:element name="baseURI" minOccurs="1" maxOccurs="unbounded">

                        <xs:complexType>

                           <xs:attribute name="server" type="xs:anyURI" use="required"/>
```

```
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
                <xs:attribute name="label" type="xs:string" use="required"/>
                <xs:attribute name="wait-time" type="xs:unsignedLong" use="required"/>
                <xs:attribute name="max-back-off" type="xs:unsignedLong" use="required"/>
                <xs:anyAttribute processContents="skip"/>
              </xs:complexType>         </xs:element>       </xs:sequence>    </xs:complexType>
  </xs:element></xs:schema>
```

6.3.3.1 Enumerations

The “Server” attribute (type “xs:anyURI”) shall only take on base URI values and not give the URI resource part.

“Label” value = {“PostFileRepair”}

[Editor’s note: for release 6, label is expected to include at least {“PostFileRepair”, “ReceptionReport”}

6.3.3.2 Example Associated Delivery Procedure Description Instance

```
<?xml version="1.0" encoding="UTF-8"?>
<mbms-associated-procedure-description-instance

                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

                xsi:schemaLocation="http://www.example.com/mbms-associated-descrition.xsd">
   <procedure label="PostFileRepair" offset-time="5" random-time-period="10">   <baseURI
server="http://mbmsrepair.operator.umts/"/> <baseURI server="http://mbmsrepair1.operator.umts/"/>
   <baseURI server="http://mbmsrepair2.operator.umts/"/></procedure>

</mbms-associated-procedure-description-instance>
```

7 Codecs

[Editor’s note: This clause writes individual codec specifications.]

7.1 General

The set of media decoders that are supported by the MBMS Client to support a particular media type are defined below. These media decoders and media types are common to all MBMS User Services. E.g. common to Streaming and Download.

7.2 Speech

7.3 Audio

7.4 Video

7.5 Still images

7.6 Text

7.7 3GPP file format

8 Scene Description

[Editor's note: This clause writes on SMIL issues. Some additions and restrictions to 3GPP SMIL profile (e.g delivery timing, URL of media objects) may be necessary.]

The scene description consists of spatial layout and a description of the temporal relation between different media that is included in the media presentation. The first gives the layout of different media components on the screen and the latter controls the synchronisation of the different media (see clause 8).

9 Security aspects

[Editor's note: This section should describe the MBMS user service security aspects. It should mainly refer to the relevant specifications (e.g. 3GPP SA3 TSs, OMA DRM Specifications)]

Annex <A> (informative): QoS consideration

[Editor's note: This clause writes QoS guidelines.]

Annex <B> (normative): FLUTE Support Requirements

This clause provides a table representation of the requirement levels for different features in FLUTE. Table 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 [9]. The terms used in Table 1 are described underneath.

Table 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. Other FEC building blocks are undefined optional plug-ins.	Compact No-Code algorithm required. [TBD]	Compact No-Code algorithm is the default option. Other FEC building blocks are undefined optional plug-ins.	Compact No-Code algorithm is the default option. [TBD]
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	Not applicable	Optional	Not applicable
A flag active (header)	Required	Required	Optional	Optional
B flag active (header)	Required	Required	Optional	Optional
T flag active and SCT field (header)	Optional	Optional	Optional	Optional
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]

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 [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.

Annex <X> (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2003					Initial draft for SA4 #28		0.0.0
Mar 2004					Second draft for SA4 PSM SWG#05		0.0.1
Apr 2004					Third draft : result of the MBMS drafting session during PSM#05 (Nortel, Vidiator, Siemens, Ericsson, Nokia, and the editor (NEC))	0.0.1	0.0.2
May 2004					Draft for SA4 #31	0.0.2	0.0.3
May 2004					Changes agreed during the PSM SWG of SA4#31	0.0.3	0.0.4
August 2004					Changes agreed during the PSM SWG of SA4#32: Tdoc S4-040395 with modifications Tdoc S4-040411 section 4.1 and 4.2 with modifications Tdoc S4-040412 with modifications Tdoc S4-040413 with modifications (6.3.2.2 + support for ptp) Tdoc S4-040414 Tdoc S4-040524 Tdoc S4-040523 with modifications Tdoc S4-040528 Tdoc S4-040396 Tdoc S4-040530 Drafting session on streaming delivery method Raised FLUTE from working assumption to agreement	0.0.4	0.0.5