Technical Specification Group Services and System Aspects ***TSGS#26(04)0832***

Meeting #26, Athens, Greece

Source: **TSG-SA WG4**

Title: 3GPP TS 26.346 version 1.5.0 "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 #26**

Document for presentation: **3GPP TS 26.346 "Multimedia Broadcast/Multicast Service; Protocols and Codecs", Version 1.5.0 (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 second time to TSG SA Plenary, for information: to the purpose, a version of the specification including revision marks allows to detect the differences from the version 1.0.0 provided at TSG SA#25. Note that since the specification was completely restructured, the revision marks are difficult to read; therefore, a "clean" version of the specification is attached as well for the reader's convenience.

Outstanding Issues:

A number of Clauses need to be reviewed and /or completed. The reader interested to capture the still open issues may refer directly to the "working draft version", which includes Editor's notes, highlights, comments, etc..

Contentious Issues:

None.

Comment(s):

None.

3GPP TS 26.346 ~~V1.0.0 (2004-09)~~V1.5.0 (2004-

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)

- 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> MBMS, FLUTE, RTP, FEC, streaming, download, broadcast, multicast

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:

ï Broadcast Mode

ï 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: ì Vocabulary for 3GPP Specificationsî .

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

[3] 3GPP TS 22.246: ì MBMS User Servicesî .

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

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

[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 ñ 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: ì Asynchronous Layered Coding (ALC) Protocol Instantiationî , M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft, December 2002.

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

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

[13] IETF RFC 3695: ì Compact Forward Error Correction (FEC) Schemesî , 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)"

[20] 3GPP TS 33.246 [TBA].

[21] UML [TBA].

[22] XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004

[23] ABNF reference [TBA]

[24] 3GPP TS 26.290: ì Extended AMR Wideband codec; Transcoding functionsî

[25] 3GPP TS 26.304: ì ANSI-C code for the Floating-point; Extended AMR Wideband codecî

[26] 3GPP TS 26.273: ì ANSI-C code for the Fixed-point; Extended AMR Wideband codecî

[27] Real-Time Transport Protocol (RTP) Payload Format for Extended AMR Wideband (AMR-WB+) Audio Codec, draft-ietf-avt-rtp-amrwbplus-01.txt

[28] 3GPP TS 26.401: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; General description".

[29] 3GPP TS 26.410: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; Floating-point ANSI-C code".

[30] 3GPP TS 26.411: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; Fixed-point ANSI-C code".

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: ì Multimedia Broadcast/Multicast Serviceî [2].

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

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

Multicast joining: See TS 22.146: ì Multimedia Broadcast/Multicast Serviceî [2].

Multicast session: See TS 22.146: ì Multimedia Broadcast/Multicast Serviceî [2].

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

MBMS user services: See TS 22.246: ì Multimedia Broadcast/Multicast Service: User Serviceî [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. An MBMS delivery method uses MBMS bearers in delivering content and may make use of associated procedures.

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 ñ Bearers, Delivery method and User service. Figure 1 depicts these layers with examples of bearer types, delivery methods and applications.

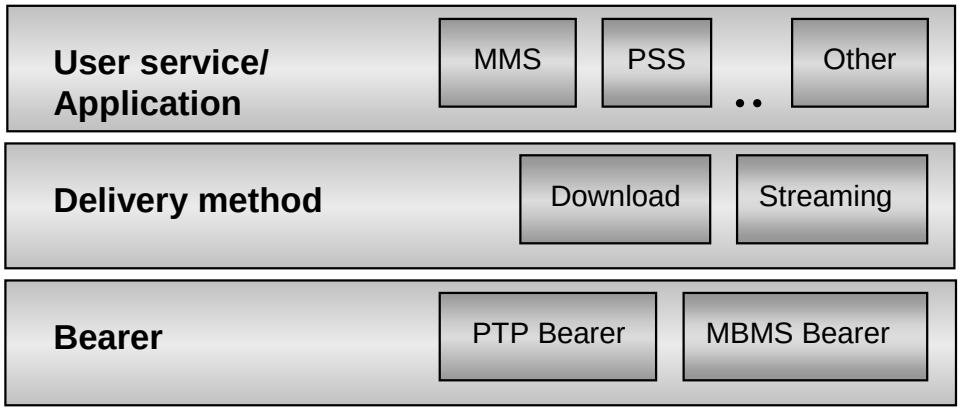


Figure ~~1:~~1: Functional Layers for MBMS User Service

– **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.~~associated delivery procedures such as file-repair, delivery verification. Two delivery methods are defined, namely download and ~~streaming. MBMS delivery may utilize both~~streaming. Delivery methods may be added beyond release 6. Delivery methods may use MBMS bearers and ~~PTP bearers.~~may make use of point-to-point bearers through a set of MBMS associated procedures.

– **User service**. The MBMS User service enables applications. Different application impose different requirements when delivering content to MBMS subscribers and may use different MBMS delivery methods. As an example a messaging application such as MMS would use the download delivery method while a streaming application such as PSS would use the streaming delivery method.

4.2 MBMS User Service Entities

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

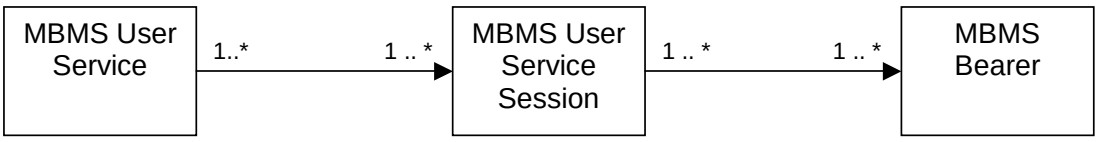


Figure ~~2:~~2: Entities and Relations

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~~clause 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

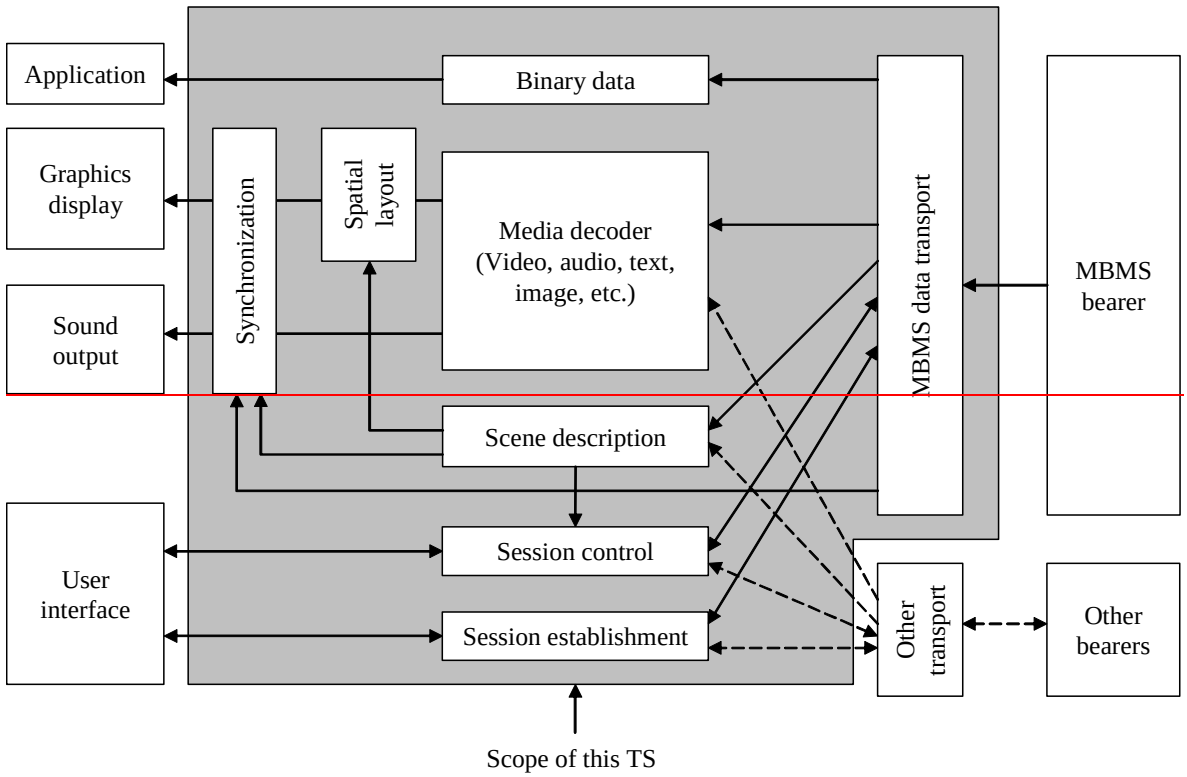
- The BM-SC;
- The GGSN;
- The UE.

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.~~Functional Entities to support MBMS User Services

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

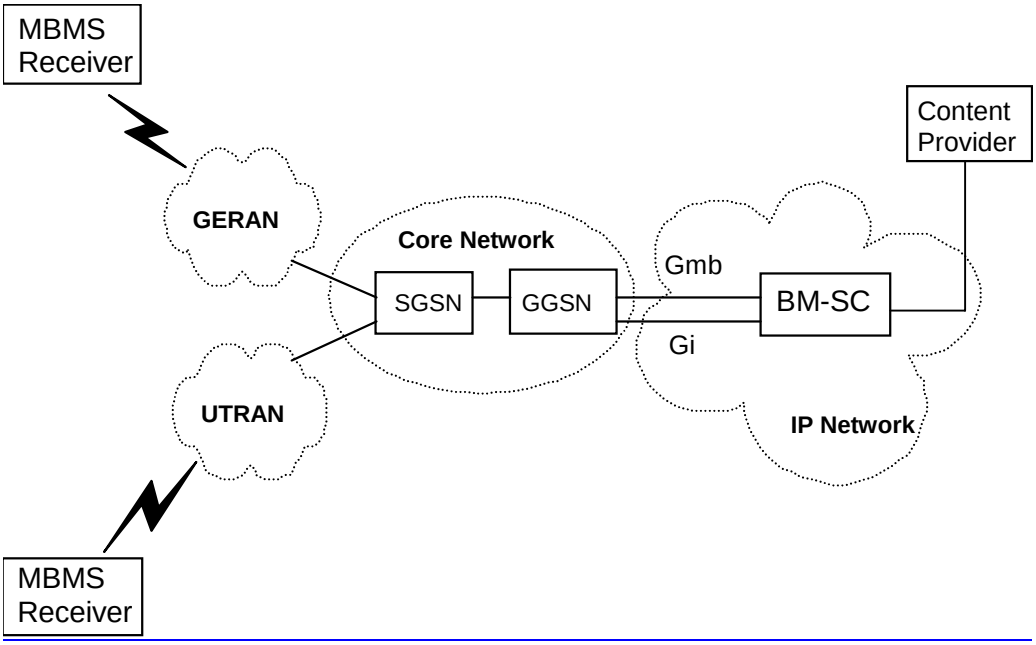


Figure 3: MBMS network architecture model

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

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

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

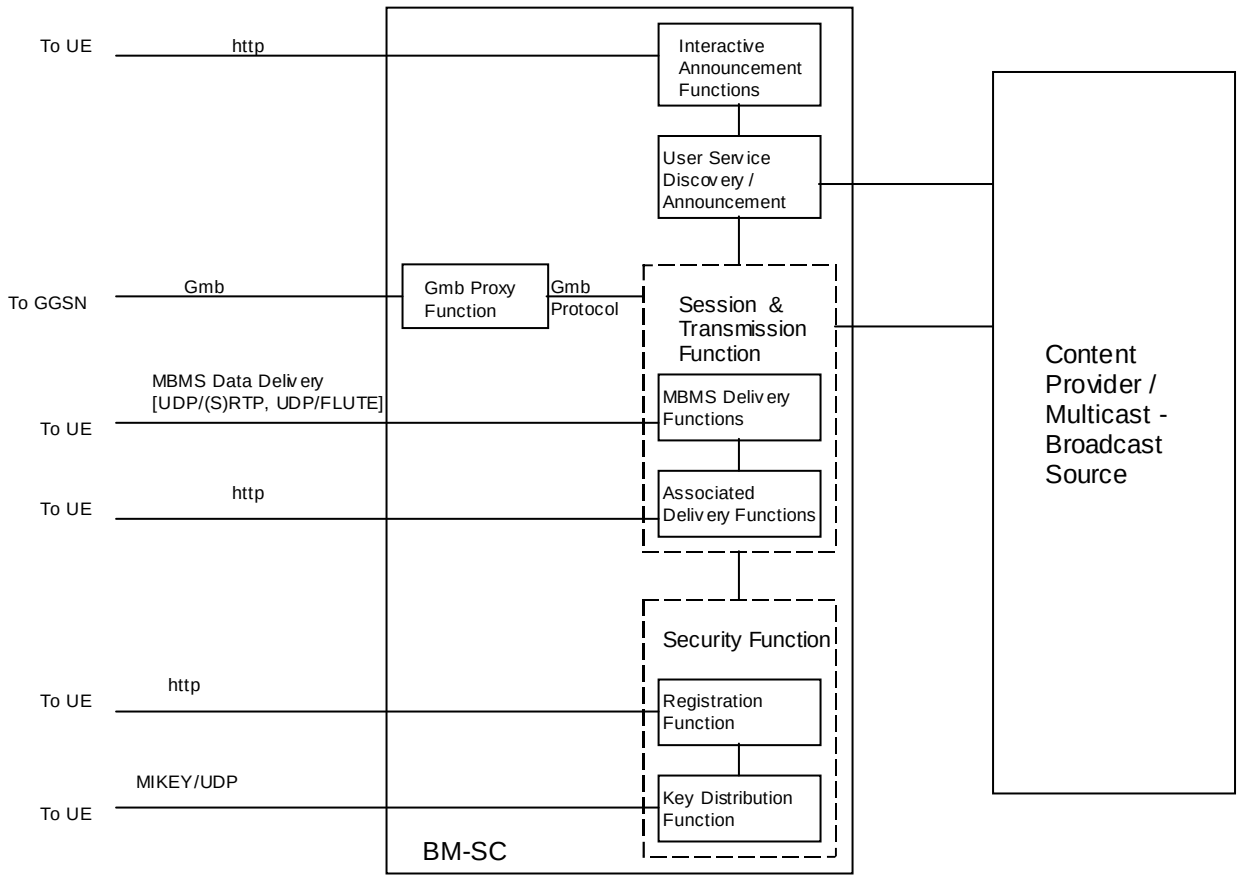


Figure 4: BM-SC sub-functional structure

[Editorís note: The http interaction between the MBMS UE and the User Service Discovery / Announcement function in is outside the scope of this specification]

[Editorís note: note that this picture highlights those BM-SC functional elements which are relevant for the SA4 work. For instance, the Security Function depicted here is part of the more general Membership function found in Fig. 5a of TS 23.246. Similar, the User Service Discovery/Announcement and Interactive Announcement functions are part of the more general Service Announcement function found in Fig. 5a of TS 23.246. Better consistency with Fig. 5a of TS 23.246 will be achieved during the final cleanup of this specification]

The Gmb-Proxy function relays the Gmb protocol between the Session function and the GGSN.

[Editorís note: SA2 is currently discussing the Gmb Proxy in the architecture. It is currently unclear whether the Gmb Proxy should be a BM-SC sub-function or belongs to an entity between a GGSN and a number of BM-SC functions. It is also unclear whether the Gmb Proxy would be an optional functionality]

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

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

- User Service Discovery / Announcement providing service description material to be presented to the end-user as well as application parameters used in providing service content to the end-user
- MBMS-based delivery of data/content (optionally confidentiality and/or integrity protected) from the BM-SC to the UE over IP multicast.
- Key Request and Registration procedure for receiving keys and key updates.

- Key distribution procedures whereby the BM-SC distributes key material required to access service data and delivered content.

- Associated Delivery functions are invoked by the UE in relation to the MBMS data transmission. The following associated delivery functions are available:

    - Point-to-point repair for download delivery method used to complement missing data using point-to-point sessions.

    - Delivery verification and reception statistics collection procedures

The interfaces between internal BM-SC functions are outside the scope of this specification.

4.4.1 Content Provider / Multicast Broadcast Source

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

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

4.4.2 MBMS Security Function

MBMS user services may use security functions for integrity and/or confidentiality protection of MBMS data. The MBMS security function is used for distributing MBMS keys (Key Distribution Function) to authorized UEs. UEs request the initial keys from the BM-SC and register (using the Registration Function) to receive key updates by key management procedures. Detailed description of the security functions is provided in [20] (TS 33.246).

[Editorís Note: The MBMS Security function is to be confirmed by SA3]

4.4.3 MBMS Session and Transmission Function

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

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

The BM-SC Session and Transmission function is further described in later clauses of this specification as well as in [4] (TS 23.246).

If security functions are activated for the MBMS User Service, the confidentiality and/or integrity protection is applied by the BM-SC to outgoing MBMS data transmissions. The traffic protection is applied between the BM-SC and the UEs. The security is based on symmetric keys, which are shared between the BM-SC and the UEs accessing the service. For further details on traffic protection see [20] (TS 33.246).

[Editorís Note: The MBMS Security function is to be confirmed by SA3].

4.4.4 Gmb Proxy function

The Gmb Proxy function relays the Gmb protocol and may fill in the MBMS bearer service oriented attributes. The BM-SC Proxy function is described in [4] (TS 23.246).

[Editorís note: SA2 is currently discussing the Gmb Proxy in the architecture. It is currently unclear whether the Gmb Proxy should be a BM-SC sub-function or belongs to an entity between a GGSN and a number of BM-SC functions. It is also unclear whether the Gmb-Proxy would be a mandatory functionality]

4.4.5 User Service Discovery / Announcement function

The User Service Discovery / Announcement provides service description information, which may be delivered via an MBMS bearer or via the interactive announcement function.

[Editorís note: Description of the User Service Description / Announcement function in this clause is ffs]

4.4.6 Interactive Announcement Function.

~~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~~An Interactive Announcement Function may offer an alternative means to provide service descriptions to the UE, e.g. using HTTP. The specification of this function is 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~~4.4.7 MBMS UE

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

In case the MBMS user service is secured, the UE needs one or more cryptographic MBMS service keys, therefore the UE requests the relevant cryptographic MBMS service keys using the MBMS security functions. The received keys are then used for securing the current session.

[Editorís Note: The MBMS Security function is to be confirmed by SA3].

5 Procedures and Protocol

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

5.1 Introduction

~~5.2 User service discovery/announcement~~This clause specifies the MBMS User service procedures and protocols.

5.2 User Service Discovery/Announcement

5.2.1 Introduction

User service discovery refers to methods for the UE to obtain a list of available MBMS user services 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.~~clause 0.

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.~~clause 5.2.3. Other user service announcement and discovery mechanisms by other means than the download delivery method are out of scope of the present specification.

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

~~User service announcement over a MBMS bearer~~

5.2.2 ~~5.2.2.1 Introduction~~MBMS User Service Description metadata fragments

MBMS ~~Service Announcements are~~User Service Discovery/ Announcement is 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.~~clause 7 or using interactive announcement functions.

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

The metadata consists of:

- a metadata fragment object describing details of MBMS user services.
- ~~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.~~service sessions
- a metadata fragment object(s) describing details of Associated delivery methods
- a metadata fragment object(s) describing details of service protection

Metadata management information consists of:

~~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

5.4Protocols

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

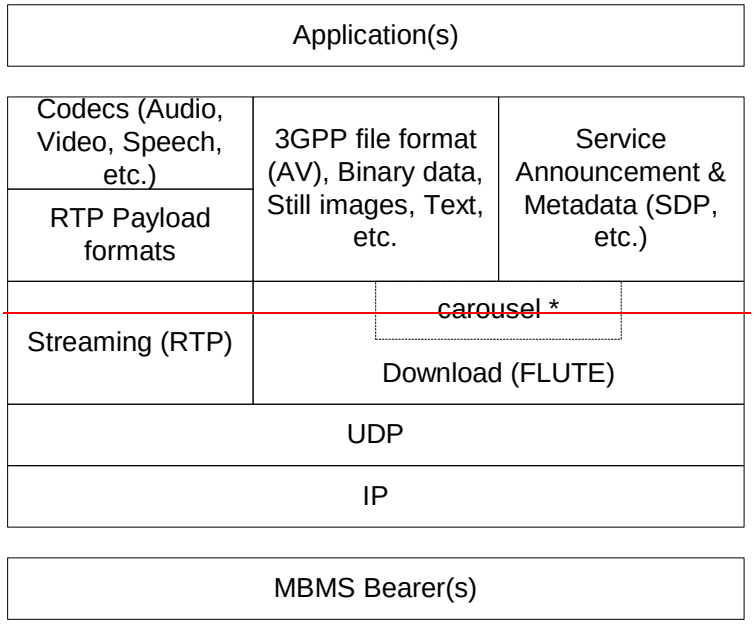


Figure 2: Protocol stack view of the MBMS User Services over MBMS bearer(s)

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.

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

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

Figure 5: Simple Description Data Model

Figure 5 illustrates the simple data model relation between these description instances using UML [21] for a single User Service Description (note, ì Nî means any number in each instance). One MBMS User Service Description instance shall include at least one delivery method description instance. The delivery method description shall refer to one session description instance.

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

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

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

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

5.2.2.1 Session Description

One or more session descriptions are contained in one session description object. The session description instance shall be formatted according to the session description protocol (SDP). Each session description instance shall contain only descriptions for only one RTP Streaming session or FLUTE Download session. One session description may describe both one download and one streaming session. The *sessionDescriptionURI* references the session description object. The session description is specified in clause 7.3for the MBMS download delivery method and in clause 8.3.16.2.3.1 for the MBMS streaming delivery method.

5.2.2.2 Associated Delivery Procedure Description

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

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

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

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

5.2.2.3 Service Protection Description

[Editorís note: this clause will be completed when the overall security functions are clear]

5.2.2.4 XML-Schema for MBMS User Service Description

The root element of the MBMS user service description is the *userServiceDescription* element. The element is of type *userServiceDescriptionType*.

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

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

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

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

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

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

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema elementFormDefault="qualified"

   targetNamespace="urn:3gpp:metadata:2004:userservicedescription"

   xmlns="urn:3gpp:metadata:2004:userservicedescription"

   xmlns:xs="http://www.w3.org/2001/XMLSchema">


   <xs:element name="userServiceDescription" type="userServiceDescriptionType"/>


   <xs:complexType name="userServiceDescriptionType">

      <xs:sequence>

         <xs:element name="name" type="nameType" minOccurs="0"

                     maxOccurs="unbounded"/>

         <xs:element name="serviceLanguage" type="xs:language" minOccurs="0"

                     maxOccurs="unbounded"/>

         <xs:element name="deliveryMethod" type="deliveryMethodType"
```

```
                    maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="serviceId" type="xs:anyURI" use="required"/>
    </xs:complexType>

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

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

</xs:schema>
```

5.2.2.5 Example MBMS User Service Description Instances

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

```
<?xml version="1.0" encoding="UTF-8"?>
<userServiceDescription
    xmlns="www.example.com/3gppUserServiceDescription"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    userServiceId="urn:3gpp:0010120123hotdog">
      <deliveryMethod
          sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>

</userServiceDescription>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<userServiceDescription
  xmlns="www.example.com/3gppUserServiceDescription"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  serviceId="urn:3gpp:1234567890coolcat">
    <name lang="EN">something in english</name>
    <name lang="DE">something in german</name>
    <name lang="FR">something in french</name>
    <name lang="FI">something in finnish</Name>
    <serviceLanguage>EN</serviceLanguage>
    <serviceLanguage>DE</serviceLanguage>
    <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>
    <deliveryMethod
      sessionDescriptionURI=http://www.example.com/3gpp/mbms/session2.sdp
        associatedProcedureDescriptionURI="http://www.example.com/3gpp/mbms/procedureX.xml"/>
    <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session3.sdp"
        associatedProcedureDescriptionURI="http://www.example.com/3gpp/mbms/procedureY.xml"/>
    <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session4.sdp"
</userServiceDescription>
```

5.2.3 User service announcement over a MBMS bearer

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

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

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

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

5.2.3.1 Supported Metadata Syntaxes

The MBMS metadata syntax supports the following set of features:

- Support of carriage of SDP descriptions, and SDP is expected to sufficiently describe at least: MBMS Streaming sessions and, MBMS download sessions;

- Support for multiple metadata syntaxes, such that the delivery and use of more than one metadata syntax is possible;
- Consistency control of metadata versions, between senders and receivers, independent of the transport and bearer use for delivery;
- Metadata fragments are identified, versioned and time-limited (expiry described) in a metadata fragment syntax-independent manner (which is a consequence of the previous two features).

5.2.3.2 Consistency control and Syntax Independence

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

5.2.3.3 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.

5.2.3.4 Delivery of the Metadata Envelope

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

5.2.3.5 Metadata Envelope Transport

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

5.2.3.6 Metadata Envelope and Metadata Fragment Association with FLUTE

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

5.3 User Service Initiation/Termination

5.3.1 Initiation

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

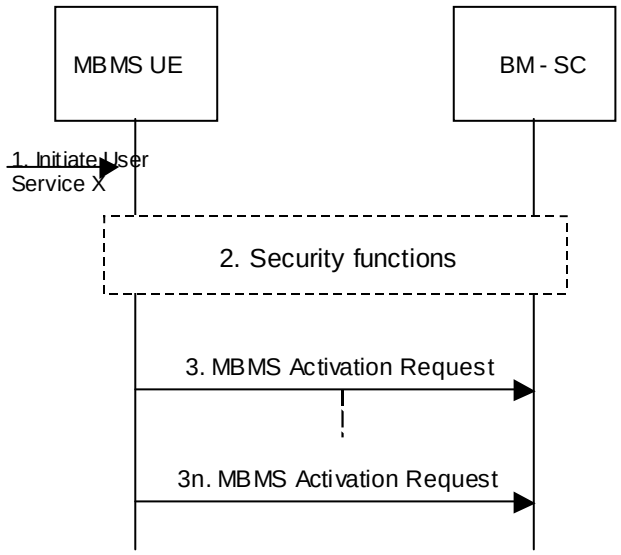


Figure 6: Initiation of an MBMS User Service

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

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

[Editorís Note: The MBMS Security function is to be confirmed by SA3].

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

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

The MBMS user service activation consists of (not necessarily in this order):

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

[Editor's Note: : Details on the security parameters are FFS]

[Editor's Note: Some more information is required on the MBMS User Service Description to finalize the MBMS user service initiation clause]

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

- Unique Service Identifier: MBMS operator specific
- Service Name: language tagged concise name - which may be human readable. (Optional - zero or more).
- Service Language: list of those provided for the service. (Optional - zero or more).
- One or more delivery method elements, each with:
    - Session Description Pointer: URI identifier of the session description (SDP file)
    - Associated Delivery Procedure Description Pointer: URI identifier of the relevant ìassociated delivery procedure descriptionî instance for this session. (More than one procedure may be listed from within the identified description, including file repair and reception reporting procedures). (Optional) If this is not used, associated delivery procedures are ìoffî for the related session
    - Security Description: URI identifier of the security description (SDP file). (Optional) If this is not used, the security procedures are ìoffî for the related session
- Session start/stop time: start time, end time (SDP)
- Session Identifier: (Source IP address + Transport Session Identifier for FLUTE) (SDP)
- Data rates: bandwidth and modifiers (SDP)
- IP addressing and ports: Destination IP address, UDP destination port (SDP)
- Protocol ID: RTP/UDP for streaming, FLUTE/UDP for download (SDP)
- Media types and codecs: media type and fmt-list (SDP)
- Bearer mode: Multicast/Broadcast bearer mode per media (SDP)

- Component language: language per media (SDP)

- FEC on/off, related parameters: FEC instance numbers as capability requirement (SDP)

NOTE: The list of languages that the service is named in (Service Name) may differ for the list of service languages the service is provided in (Service Language).

[Editor's note: RTP session Id is To Be Decided]

5.3.2 MBMS User Service termination procedure

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

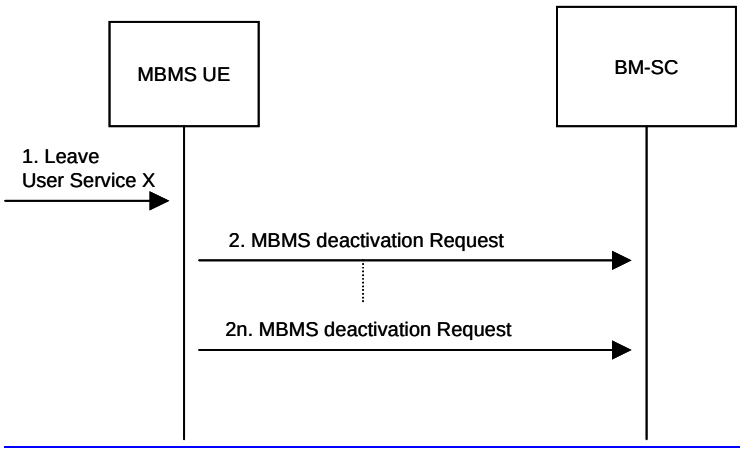


Figure 7: Termination of an MBMS user service

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

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

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

5.4 MBMS Data Transfer Procedure

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

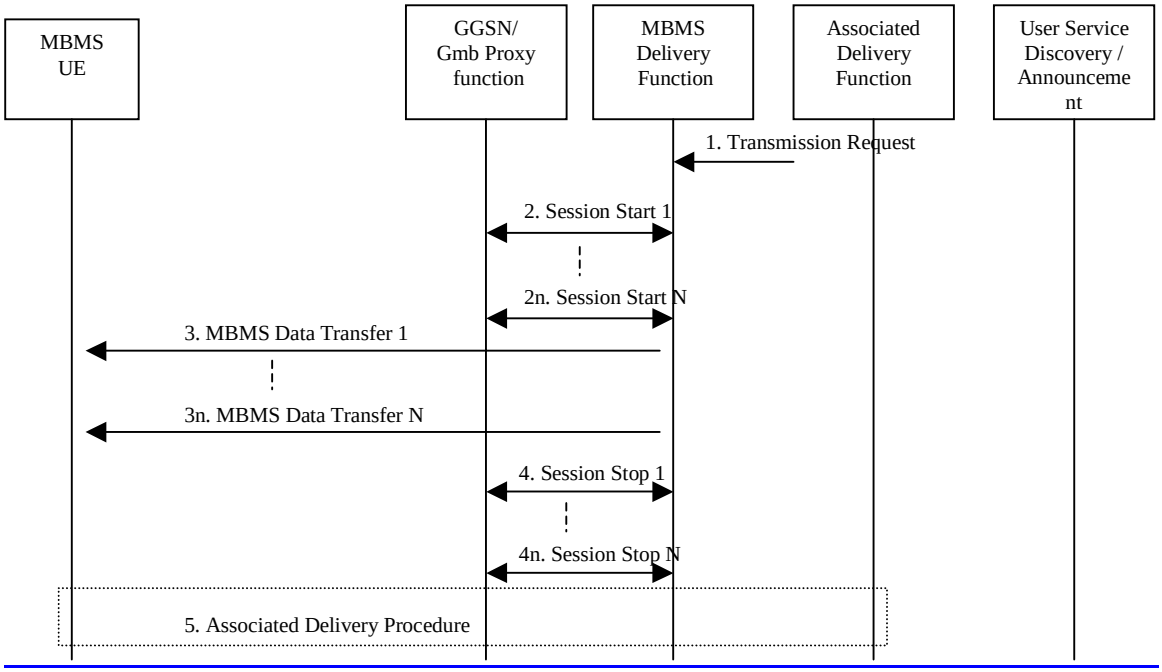


Figure 8: Procedure of MBMS Data Transfer

[Editorís Note: security related interactions are not yet depicted in the sequence]

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

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

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

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

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

[Editorís Note: The MBMS Security function is to be confirmed by SA3]

5.5 MBMS Protocols

Figure 9 illustrates the protocol stack used by MBMS User services.

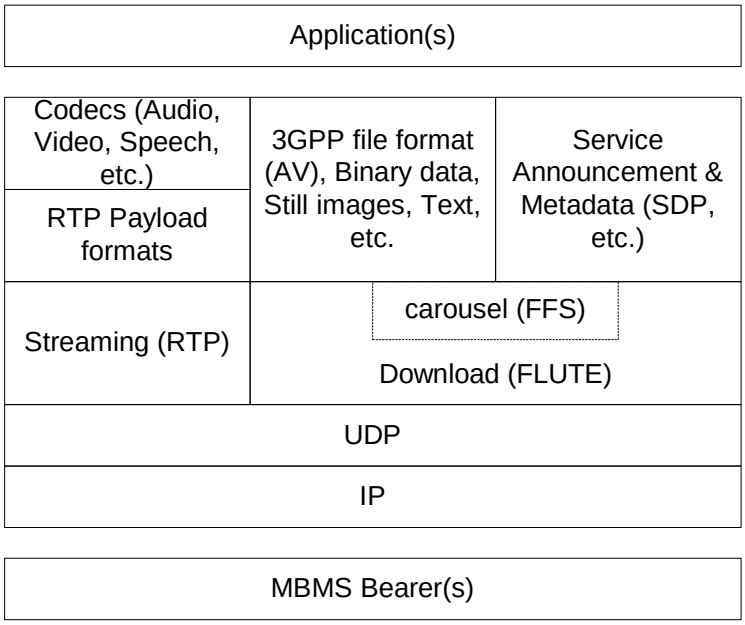


Figure ~~3: Building block structure of FLUTE~~

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

6 Introduction on Delivery Methods

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

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

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

7 Download Delivery Method

7.1 Introduction

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

FLUTE is built on top of the Asynchronous Layered Coding (ALC) protocol instantiation [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 ~~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 ñ the File Description Table (FDT) Instances ñ 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 ì Sender Current Time (SCT)î field (when T=1).
- The R flag shall indicate the use of the optional ì Expected Residual Time (ERT)î 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.

6.2.2.1.1 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 (ìa=source-filter:î) [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.

6.2.2.1.2 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.

6.2.2.1.3 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 ìconnection dataî field (ìc=î) of SDP [14, 15]. The destination port number shall be defined according to the <port> sub-field of the media announcement field (ìm=î) of SDP [14, 15].

The presence of a FLUTE session on a certain channel shall be indicated by using the ëm lineí 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.

6.2.2.1.4 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 = 1*aî 1 =î 1 flute-tsiî 1 :î 1 integer CRLF*

integer = as defined in [14]

6.2.2.1.5 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.

6.2.2.1.6 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 (ît=î) [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 Sessiondescription

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 deliveredan 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 ìassociated procedure descriptionî 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 ìglobal defaultsî, 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 îapplication/mbms-associated-procedure-parameterî identifies associated delivery procedure description instances (configuration files).

In XML, each associated delivery procedure entry shall be configured using a ìprocedureî element. All configuration parameters of one associated delivery procedure are contained as attributes of a ìprocedureî element. The ìlabelî attribute of a ìprocedureî 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 clientsí 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.

6.2.2.1.7 6.3.2.1.1 Identification of Missing Data from an MBMS Download

6.2.2.1.8 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.

6.2.2.1.9 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 ìoffset-timeî attribute.

6.2.2.1.10 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 ìrandom-time-periodî attribute.

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

6.2.2.1.11 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.

6.2.2.1.12 6.3.2.1.3 File Repair Server Selection

6.2.2.1.13 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 theassociated delivery procedure

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 10 for an illustration of FLUTE building block structure. FLUTE is carried over UDP/IP, and is independent of the IP version and the underlying link layers used.

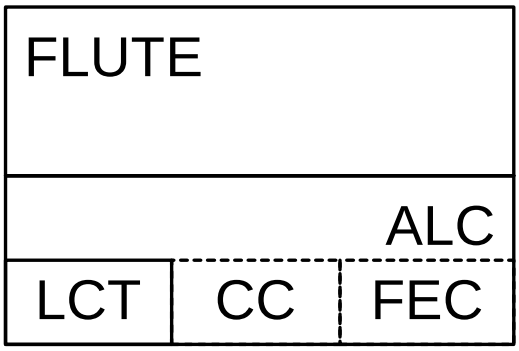


Figure 10: Building block structure of FLUTE

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

7.2 FLUTE usage for MBMS download

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

In 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.

7.2.1 Fragmentation of Files

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

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

[Editorís note: The limitation of FLUTE packet payload length is still under discussion]

7.2.2 Symbol Encoding Algorithm

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

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

7.2.3 Blocking Algorithm

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

7.2.4 Congestion Control

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

7.2.5 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 ì Sender Current Time (SCT)î field (when T=1).
- The R flag shall indicate the use of the optional ì Expected Residual Time (ERT)î 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.

[Editorís note: This clause is still under discussion]

7.2.6 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.

[Editorís note: This clause is still under discussion]

7.2.7 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.

[Editorís note: This clause is still under discussion.]

[Editorís note: There may be other FDT extension parameters to include]

7.3 SDP for Download Delivery Method

7.3.1 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 [23] (ABNF reference).

7.3.2 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 per media;
- The Transport Session Identifier (TSI) of the session;
- The start time and end time of the session;
- The protocol ID (i.e. FLUTE/UDP);
- Media type(s) and fmt-list;
- Data rate using existing SDP bandwidth modifiers;
- Mode of MBMS bearer per media;
- FEC capabilities and related parameters;
- Service-language(s) per media.

This list includes the parameters required by FLUTE [9]

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

7.3.2.1 Sender IP address

There shall be exactly one IP sender address per MBMS download session, and thus there shall be exactly one IP source address per complete MBMS download session SDP description. The IP source address shall be defined according to the source-filter attribute (ì a=source-filter:î) [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.

7.3.2.2 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.

7.3.2.3 Destination IP address and port number for channels

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

- IP destination address
- Destination port number.

The IP destination address shall be defined according to the ì connection dataî field (ì c=î) of SDP [14, 15]. The destination port number shall be defined according to the <port> sub-field of the media announcement field (ì m=î) of SDP [14, 15].

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

m=application 12345 FLUTE/UDP 0

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

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

7.3.2.4 Transport Session Identifier (TSI) of the session

The combination of the TSI and the IP source address identifies the FLUTE session. Each TSI shall uniquely identify a FLUTE session for a given IP source address during the time that the session is active, and also for a large time before and after the active session time (this is also an LCT requirement [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 = ì aî ì =î ì flute-tsi*î î :î integer CRLF*

integer = as defined in [14]

7.3.2.5 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.

7.3.2.6 Session Timing Parameters

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

7.3.2.7 Mode of MBMS bearer per media

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

a=mbms-mode:broadcast 1234

Where any multicast mode media are described, the MBMS bearer mode declaration attribute shall not be used at session level and shall not be used at media level for multicast mode media.

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

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

tmgi = 1*DIGIT

7.3.2.8 FEC capabilities and related parameters

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

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

This can be session-level (and so the first instance (fec-ref=0) becomes the default for all media) and media-level to specify differences between media. This is optional as the information will be available elsewhere (e.g. FLUTE FDT Instances). If this attribute is not used, and no other FEC-OTI information is signaled to the UE by other means, the UE may assume that support for FEC id 0 is sufficient capability to enter the session.

A new FEC-declaration attribute shall be defined which results in, e.g.;

a=FEC:0

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

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

sdp-fec-declaration-line = "a=FEC-declaration:" fec-ref SP fec-enc-id ";" [SP fec-inst-id] CRLF

fec-ref = 1*DIGIT (value is the SDP-internal identifier for FEC-declaration).

fec-enc-id = "encoding-id=" enc-id

end-id = 1*DIGIT (value is the FEC Encoding ID used).

fec-inst-id = "instance-id=" inst-id

inst-id = 1*DIGIT (value is the FEC Instance ID used).

sdp-fec-line = "a=FEC:" fec-ref CRLF

fec-ref = 1*DIGIT (value is the FEC-declaration identifier).

7.3.2.9 Service-language(s) per media

The existing SDP attribute ìa=langî is used to label the language of any language-specific media. The values are taken from RFC 3066 which in turn takes language and (optionally) country tags from ISO639 and 3661 (e.g. "a=lang:EN-US"). These are the same tags used in the User Service Description XML.

7.3.3 SDP Examples for FLUTE Session

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

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

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

8 Streaming delivery method

8.1 Introduction

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

8.2 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.

[Editorís note: it was agreed that RTCP feedback could be useful in future releases]

8.2.1 FEC mechanism for RTP

[Editor's note: this scheme is intended to be generic. If the chosen FEC scheme(s) doesn't fit, it can be modified]

The generic mechanism for systematic FEC of RTP streams consists of two RTP payload formats, one for FEC source packets, one for FEC protection data, and their related signaling. In addition a FEC algorithm definition is needed including source block construction, definition of one or more FEC payload ID (FPID) and Object Transmission Information (OTI). A receiver supporting this delivery method shall support the FEC payload format for source RTP packets and shall support the payload format for FEC symbols. The two RTP payload formats interact to create the FEC mechanism. The concept of the protection mechanism is briefly discussed below. More details can be found in the following sub-clauses.

Figure 11 depicts the interaction between the different protocol implementations of a sender and receiver. On the sender side, an original RTP packet has been created. The original packet consists of an RTP payload and the RTP header including the RTP header fields specified by the payload (timestamp, payload type, marker bit). An original packet has been processed to generate the RTP header fields, i.e. assigned SSRC and sequence number, CRSRC list, etc. If the packet were sent without FEC protection it would be forwarded for RTP processing. RTP processing is defined here as the activities that the RTP stack performs to gather RTCP statistics (bytes and packet sent) and optionally perform padding, i.e. operations that are dependent on the packet size. For original RTP packets that are going to be FEC protected, the RTP packet is encapsulated into the FEC payload format for source RTP packets. This encapsulation consists of the addition of a FEC payload ID field prior to the original payload. The RTP header is maintained with the exception of the substitution of the original RTP payload type (indicating the type of media) to the FEC source payload type, so to identify the use of the FEC source payload format. The FEC payload ID data is provided by the FEC encoder. The completed FEC source RTP packet is forwarded for RTP processing thus ensuring correct RTCP statistics.

In addition of being scheduled for transmission, a copy of the complete RTP packet is also processed by the FEC encoder. The FEC encoder starts with mirroring the de-capsulation operation that will be performed by the receiver after the reception of a source packet. The FEC de-capsulation implements the steps necessary to reconstruct the original RTP packet that would have been sent had it not been FEC protected. More precisely, the de-capsulation substitutes the payload type value of the FEC payload format with the payload type value of the source payload format, and removes the FEC payload ID field. The FEC decoder takes this reconstructed original RTP packet and places it into its encoding source block in accordance with the FEC payload ID data, see clause 8.2.1.7.

After processing all original RTP packets to be protected, the FEC encoder generates the desired amount of FEC protection data, i.e. encoding symbols. These encoding symbols are packetized in the FEC protection data payload format and sent to the receiver as RTP packets. The protection data RTP packets shall be sent using an SSRC different from the source RTP packetís SSRC, but within the same RTP session. Doing so avoids non-continuous sequence numbering spaces for both the FEC protection and the source media RTP packet streams.

The receiver forwards the received source packets directly to the media receiver buffer, after reception and de-capsulation of the FEC source payload format as discussed above. A copy of (or reference to) the reconstructed original RTP packet is also processed by the FEC decoder. The FEC decoder places the original RTP packet into the source block in accordance with the source packet's FEC payload ID. When receiving protection data packets, the receiver depacketizes the encoding symbols and uses them if necessary to decode missing portions of the source block. If any source RTP packets have been lost, and sufficient source and protection data packets have been received, FEC decoding can be performed. If complete original RTP packets are recovered, they are forwarded to the media receiver buffer, and for further media processing.

Note that the RTP receiver buffer, must have enough depth to buffer all the original packets and allow time for the protection packets to arrive and decode missing packets before any of these packets are consumed by the media decoder.

The FEC payload ID (Source and Protection) are used to associate the source RTP packets and protection RTP packets, respectively, to a source block. The FEC payload ID are part of the definition of the FEC scheme and is identified by the FEC encoding ID and instance ID for underspecified FEC encoding IDs. One FEC encoding ID for the streaming delivery method is specified in clause 8.2.1.6. Any FEC encoding IDs using the defined RTP payload formats shall be a systematic FEC code and may use two different FEC payload IDs. If two FEC payload ID is used, the FEC Encoding ID shall define which FEC payload ID format shall be used in the source and protection data RTP payload format respectively.

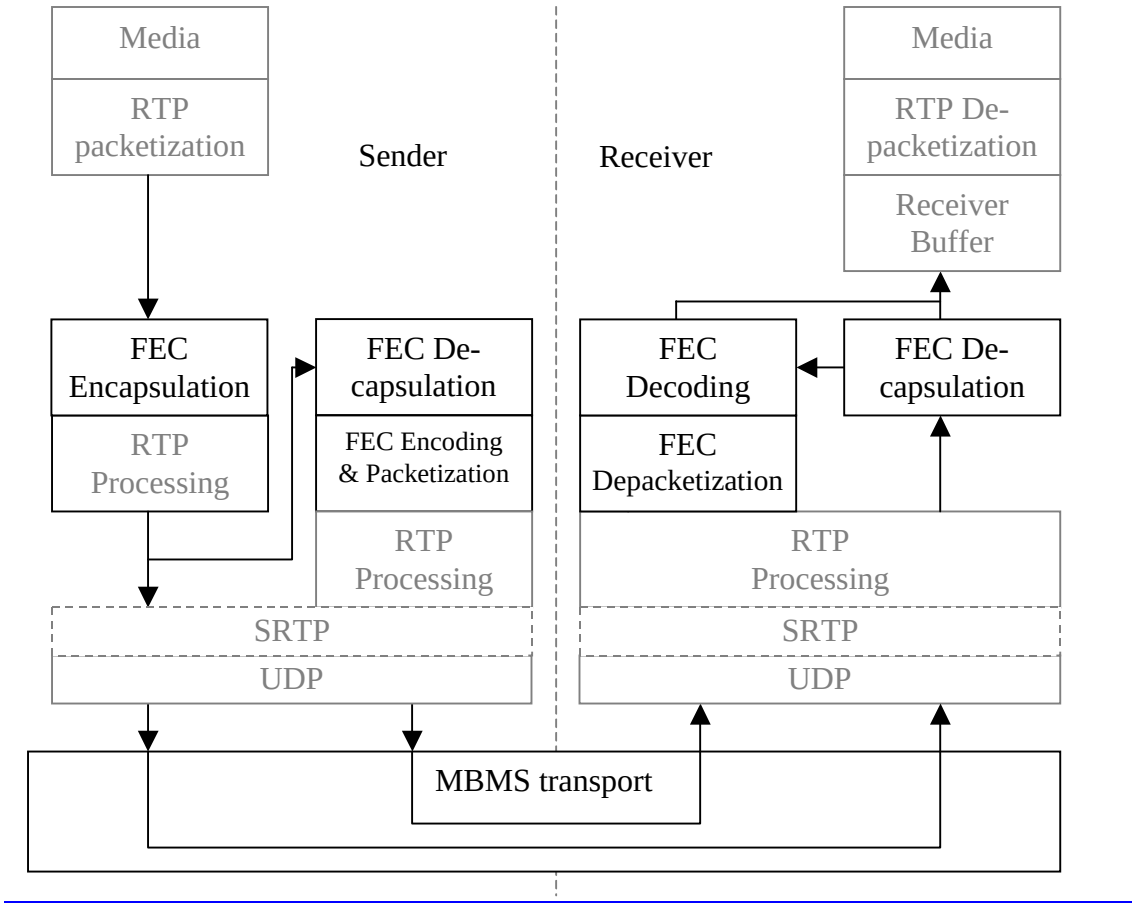


Figure 11: FEC mechanism for RTP interaction diagram

8.2.1.1 Sending Terminal Operation

A sender may send RTP packets that are not FEC protected (original RTP packets), and shall then use the original PT assigned to payload format and bypass the below defined operation. Original RTP packets that are going to be FEC protected shall use the below procedure.

A sender performs the following operations:

1. Start with an original RTP packet complete with RTP header and payload, including profile specific extensions, and variable sized fields as CSRC list and extension headers.
2. The sender reserves the position and the amount of space needed for the original RTP packet information in the FEC code's source block. In doing so, the FPID information can be determined. See Clause 8.2.1for details.
3. The source RTP packet is created by inserting the FPID after the RTP header and before the payload header.
4. To allow the receiver to identify the original RTP payload type and indicate that the FPID is present, the RTP PT is changed from its original value to the value that indicate the combination of original payload format and the inserted FPID.
5. The source RTP packet generated is forwarded for RTP processing where RTCP statistics are gathered.
6. After RTP processing, a copy of the packet being sent to the receiver(s) is processed to reconstruct the original RTP packet. In particular, the FPID is removed, and the RTP payload type (that indicated the insertion of the FPID in the original RTP packet) is replaced with the original PT.
7. The resulting RTP packet is placed in the correct source block and in the position determined in step 2 above.

8. When a source block is complete, the FEC encoder generates encoding symbols and places these symbols into protection RTP packets, to be conveyed using normal RTP mechanisms. The SSRC for the protection packets are used in these transmissions.

8.2.1.2 Receiving Terminal Operation

A receiver operates as follows, when receiving a packet:

1. If an RTP packet is not FEC encapsulated as indicated by the payload type, it is forwarded to the receiver buffer and the operation ends.
2. If a source RTP packet with a payload type that indicates that it is FEC encapsulated is received:
    a. The FPID is removed from the RTP packet and stored.
    b. The RTP payload type is replaced with the original PT based on the received PT.
    c. The original packet is forwarded to RTP receiving entity, in Figure 11 the receiver buffer. A copy is forwarded together with the FPID to the FEC decoder. As an alternative, a reference to the receiver buffer together with the FPID may be used.
    d. FEC decoder places the RTP packet into the source block according to the FPID.
3. If a protection RTP packet is received as indicated by the payload type, the encoding symbols it contains are depacketized and passed on to the FEC decoder.
4. If all source packets have been received no FEC decoding is necessary.
5. If some source RTP packets are missing as detected by the gaps in the sequence number space, then the FEC decoder determines if it has received enough source and protection packets to decode and performs the decoding operation.
6. Any missing source RTP packets that were reconstructed during the decoding operation can be forwarded directly to the RTP receiving entity, i.e. receiver buffer.

It should be understood that the RTP receiving entity will require some form of buffering and packet re-ordering so to insert any reconstructed packets into their appropriate sequencing. The amount of buffering depends primarily on the media decoder implementation, and its tolerance to do decoding on reordered packets, and the (maximum) size of the source block. For media requiring decoding in order, the minimal delay can be directly derived as the maximum time duration represented by the RTP packets allowed in each source block. To allow receivers to determine the minimal buffering requirement, the repair RTP packets payload formats has a parameter "min-buffer-time" as defined in 8.2.1.11.

8.2.1.3 RTP Payload format for source RTP packets

The FEC payload format for source RTP packets shall be used to add a FEC payload ID field to an original RTP packet that is going to be FEC protected. This enables the FEC sender and receiver to correctly place the source (original) information into its FEC encoding and decoding process respectively. The FEC algorithm specific addressing of any source information is provided in the FEC payload ID. This payload format is identified by the media type defined in clause 8.2.1.13.

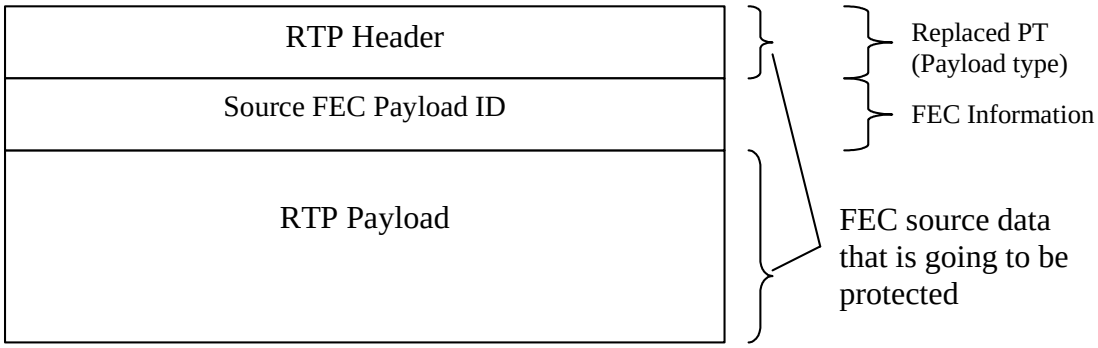


Figure 12: Principle of FEC payload format for source RTP packets

Figure 12 depicts the layout of a FEC payload packet. Into the original RTP packet and between the RTP header and the RTP payload (including any payload specific headers), the FEC payload ID is inserted. In addition, the Payload Type field in the RTP header shall be changed to indicate the presence of the Source FEC Payload ID. No other changes to the RTP header shall be done. This forms the RTP packet to be sent over the wire. For the FEC calculation, however, the reconstructed original RTP packet, is used in the source block as source information.

The reconstruction of the original RTP packet is done in two simple steps:

1. Replace the RTP header payload type (PT) field with the original payload type value as indicated by the "opt" parameters value for the actually present PT.

2. Remove the source FEC payload ID by placing the RTP payload part directly after the RTP header.

The precise structure and length of the Source FEC payload ID depends on the FEC scheme employed. The FEC scheme and its necessary information is defined in Annex <B> and indicated by out-of-band methods as defined in clause 8.2.1.14.

8.2.1.4 RTP Payload Format for Protection Data

The RTP payload format for protection data carries encoding symbols generated by the FEC encoding process as payload. The basic format for a protection RTP packet is depicted in Figure 13. The RTP header is followed by the Repair FEC Payload ID, which are the RTP payload header for one or more encoding symbols. Those encoding symbols fill the rest of the packet. The RTP payload format is identified by the media type (application/rtp-mbms-fec-symbols) defined in Clause 8.1.2.13.

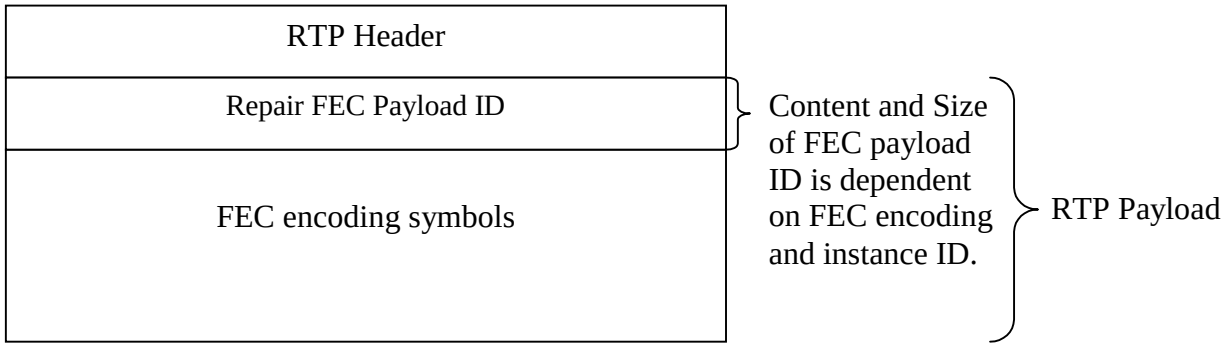


Figure 13: RTP payload structure for repair RTP packets

The RTP header information for the protection RTP packet is set as follows:

Marker bit: The marker bit shall be set 1 for the last protection RTP packet sent for each source block, and otherwise set to 0.

Timestamp: The timestamp rate shall be 10 kHz and shall be set to a time corresponding to the packet's transmission time. The timestamp value has no use in the actual FEC protection process and is only set to a value to produce reasonable resolution for arrival measuring and jitter calculation.

Sequence number: Is set in accordance with RFC 3550 [6]. The sequence number is primarily used to detect losses of the protection RTP packets. All protection RTP packets for a source block shall be sent in consecutive order, and be complete before starting the transmission of the next source block.

Payload type (PT): Is dynamically allocated using an out-of-band signalling mechanism. The PT is used to determine the FEC parameters for the payload. The FEC encoding ID, FEC instance ID, and any additional FEC object transmission information is all determined through the PT.

SSRC: One SSRC is used per source SSRC. The SSRC used by the protection payload format shall be different the one used by the source RTP packets. The binding of the source SSRC to the repair SSRC shall be performed using the RTCP SDES CNAME, which shall be identical for the two SSRCs.

The FEC Payload ID is of a fixed in size for a given PT, as the PT defines the FEC scheme employed and its parameters (if any). A FEC encoding ID and its FEC payload ID is defined in 8.2.1.

8.2.1.5 A Structure of the FEC source block

Editor's Note: Other methods may be defined by the selected FEC code.

This clause defines the method for forming the FEC source block that is utilized by the FEC encoding ID specified in the Clause 8.2.1.6.

How to reconstruct the original RTP packets that are to be included in a source block from source RTP packets are described in Clause 8.1.2.3. Let L be the length in bytes of an original RTP packet that is to be added to the source block. The source symbol size is T bytes and shall be whole bytes. When the original RTP packet is placed into the source block the value of L is first written as a two-byte value, followed by the packet. If the 2 byte RTP packet length indicator and source RTP packets with size L do not align with a encoding symbol boundary, then padding up to the next boundary shall be done using the value 0 in each byte. As long as any source RTP packets remain to be placed, the procedure is repeated starting writing the RTP packet size in the next encoding symbol.

An example of forming a source block follows, where the symbol size T is 16 bytes. Initially the source block contains K = 0 source symbols. If the first source RTP packet is 26 bytes in length then 26 is written into the first two bytes of source symbol K = 0 of the source block followed by the 26 bytes of the RTP packet. The packet then spans U = ceil((2+26)/16) = 2 source symbols. The last 4 bytes of source symbol K+Uñ1 = 1 are filled with zeroes to the next source symbol boundary, and the number of source symbols K in the source block is updated to K+U = 2. If the second original RTP packet is 52 bytes in length then 52 is written into the first two bytes of source symbol K = 2 of the source block followed by the 52 bytes of the RTP packet. The packet then spans U = ceil((2+52)/16) = 4 source symbols. The last 10 bytes of source symbol K+Uñ1 = 5 are filled with zeroes to the next source symbol boundary, and K is updated to K+U = 6. If the third original RTP packet is 103 bytes in length then 103 is written into the first two bytes of source symbol K = 6 of the source block followed by the 103 bytes of the RTP packet. The packet then spans U = ceil((2+103)/16) = 7 source symbols. The last 7 bytes source symbol K+Uñ1 = 12 are filled with zeroes to the next source symbol boundary, and K is updated to K+U = 13. If the third original RTP packet is the last of the source block then the number of source symbols in the source block is K = 13. The source block for this example is illustrated in Figure 14, where each entry is a byte and the rows correspond to the source symbols and are numbered from 0 to 12.

26		$B_{0,0}$	$B_{0,1}$	$B_{0,2}$	$B_{0,3}$	$B_{0,4}$	$B_{0,5}$	$B_{0,6}$	$B_{0,7}$	$B_{0,8}$	$B_{0,9}$	$B_{0,10}$	$B_{0,11}$	$B_{0,12}$	$B_{0,13}$
$B_{0,14}$	$B_{0,15}$	$B_{0,16}$	$B_{0,17}$	$B_{0,18}$	$B_{0,19}$	$B_{0,20}$	$B_{0,21}$	$B_{0,22}$	$B_{0,23}$	$B_{0,24}$	$B_{0,25}$	0	0	0	0
52		$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$	$B_{1,5}$	$B_{1,6}$	$B_{1,7}$	$B_{1,8}$	$B_{1,9}$	$B_{1,10}$	$B_{1,11}$	$B_{1,12}$	$B_{1,13}$
$B_{1,14}$	$B_{1,15}$	$B_{1,16}$	$B_{1,17}$	$B_{1,18}$	$B_{1,19}$	$B_{1,20}$	$B_{1,21}$	$B_{1,22}$	$B_{1,23}$	$B_{1,24}$	$B_{1,25}$	$B_{1,26}$	$B_{1,27}$	$B_{1,28}$	$B_{1,29}$

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

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

8.2.1.6 FEC Encoding ID definition

This clause defines a FEC encoding scheme and is identified by the FEC encoding ID [TBA]. It utilized the method for forming FEC source block as defined in Clause 8.2.1.3. It defines two different FEC payload IDs, one for source RTP packets and another for FEC encoding symbols that are used with the corresponding RTP payload formats.

Editorís Note: This clause requires some rewording after the final decision on the FEC scheme(s) to be supported.

8.2.1.7 FEC Payload ID for Source symbols

The Source FEC payload ID is composed as follows:

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

Figure 1: Source FEC Payload ID

Source Block Number (SBN), (8 or 16 bits): The I-D of the source block the media packet belongs to.

Encoding Symbol ID (ESI), (8 or 16 bits): The starting symbol index of the source packet in the source block.

Editorís Note: The Source Block number could be 1 or 2 bytes and is independent of the FEC scheme ñ its length is a tradeoff decision between wrap-round resilience and overhead. It would also be possible to allow both lengths and distinguish by SDP. This needs to be decided.

Editorís Note: For your information, as far as we understand the various FEC proposals, the length of SBN and ESI is as follows:

	DF	Siemens	NEC	Bamboo	Nokia
SBN	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2
ESI	2	2	2	2	1

8.2.1.8 FEC payload ID for Repair symbols

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

Source Block Number SBN

Encoding Symbol ID ESI
Source block length SBL
Encoding block length EBL
Symbol Length T

Figure 2: Repair FEC Payload ID

Source Block Number (SBN), (8 or 16 bits): The I-D of the source block the media packet belongs to.

Encoding Symbol ID (ESI), (8 or 16 bits): The starting symbol index of the source packet in the source block.

Source Block Length (SBL), (8, 16, or 32 bits): The number of source symbols in the source block.

Encoding Block Length (EBL), (not present, 8, 16, or 32 bits): The total number of symbols (source symbols of the source block plus encoding symbols generated from the source block) that are sent for the source block.

Symbol Length (T), (not present, 8 or 16 bits): The length of a symbol in bytes.

Editorís note: The rest of this clause requires rewording after the FEC schemes supported have been selected.

Editorís note: For your information, as far as we understand the various FEC proposals, the length of the various FEC payload ID fields is as follows:

	DF	Siemens	NEC	Bamboo	Nokia
SBN	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2
ESI	2	2	2	2	1
SBL	2	4	2	2	1
EBL	0	4 (*)	2 (*)	2 (*)	1
T	2 (*)	2 (*)	2 (*)	2 (*)	1

Editorís note: the (*) in the above table denotes fields that could probably be of zero length here, and conveyed in SDP.

8.2.1.9 FEC encoding procedures

Editor's note: Here the exact procedures of the FEC scheme is to be defined. Information that needs IANA registration also needs to be indicated.

8.2.1.10 Signalling

The two different RTP payload formats requires each a media type to identify them and define their respective set of parameters.

8.2.1.11 Registration of media type application/rtp-mbms-fec-symbols

This media type represents the RTP payload format defined in Clause 8.2.1.4.

Type name: application

Subtype name: rtp-mbms-fec-symbols

Required parameters:

FEID: The FEC Encoding ID used in this instantiation. Expressed either as an integer or a string without white space.

min-buffer-time: The minimum RTP receiver buffer time need to ensure that FEC repair has time to happen. The value is in milliseconds. The min-buffer-time shall be the maximum time allowed for the sender between sending the first and the last packet, source or repair, belonging to the same source block.

Optional parameters:

FIID: The FEC Instance ID used in this instantiation. Expressed either as an integer or a string without white space. Parameter shall be present for all FEC encoding IDs that are not fully specified.

FOTI: The additional FEC object transmission information if any that the FEC instantiation uses as defined by the FEID and FIID. The content is expected to be binary data but is not necessarily so, however it shall always be BASE64 encoded in the parameter value.

Encoding considerations:

The binary parameter FOTI shall be encoded using BASE 64. The RTP payload format is a binary one, however as it is restricted to usage over RTP, no special considerations are needed.

Restrictions on usage:

This format is only defined for transfer over RTP RFC3550.

Security considerations:

This format carries protection data and instructions used in FEC decoding. Thus alteration or insertion of packets can cause wide spread corruption of recovered data. Thus authentication and integrity protection of the format is appropriate. See also clause 7 of RFC 3452.

Interoperability considerations:

The greatest interoperability issue with this format is that it virtually supports any systematic FEC encoding scheme. Thus the issue is to ensure that both sender and receiver are capable of using the same FEC codes.

Published specification:

3GPP Technical specification TS 26.346

Applications which use this media type:

MBMS terminals capable of receiving the MBMS streaming delivery method.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh File Type Code(s): N/A

Person & email address to contact for further information:

Magnus Westerlund (magnus.westerlund@ericsson.com)

Intended usage:

COMMON

Author:

3GPP SA4

Change controller:

3GPP TSG SA

8.2.1.13 Registration of media type audio, video, or text/rtp-mbms-fec-tag

This media type represents the RTP payload format defined in Clause 8.1.2.4.

Type name: audio, video, or text

Subtype name: rtp-mbms-fec-tag

Required parameters:

opt: The original payload type (OPT) of the media that is tagged by this instantiation of the format. This is an unsigned integer which range depends on the RTP profile in use, but commonly 0-127.

rate: An integer value, equal to the RTP timestamp rate value for the payload type specified by "opt".

FEID: The FEC Encoding ID used in this instantiation. Expressed either as an integer or a string without white space.

Optional parameters:

FIID: The FEC Instance ID used in this instantiation. Expressed either as an integer or a string without white space. Parameter shall be present for all FEC encoding IDs that are not fully specified.

FOTI: The additional FEC object transmission information if any that the FEC instantiation uses as defined by the FEID and FIID. The content is expected to be binary data but is not necessarily so, however it shall always be BASE64 encoded in the parameter value.

Encoding considerations:

This format is a binary one, however as it is restricted to usage over RTP, no special considerations are needed.

Restrictions on usage:

This type is only defined for transfer over RFC3550.

Security considerations:

This format carries source data and instructions used in FEC decoding. Thus alteration or insertion of packets can cause wide spread corruption of recovered data. Thus authentication and integrity protection of the format is appropriate. See also clause 7 of RFC 3452.

Interoperability considerations:

The greatest interoperability issue with this format is that it supports any FEC encoding that follows the rules of RFC 3452. Thus the issue is to ensure that both sender and receiver are capable of using the same FEC codes.

Published specification:

3GPP Technical specification TS 26.346

Applications which use this media type:

MBMS terminals capable of receiving streaming media.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh File Type Code(s): N/A

Person & email address to contact for further information:

Magnus Westerlund (magnus.westerlund@ericsson.com)

Intended usage:

COMMON

Author:

3GPP SA4

Change controller:

3GPP TSG SA

8.2.1.14 Mapping the Media types to SDP

The information carried in the MIME media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [6], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the FEC payload format, the mapping is as follows:

- The Media type (application, audio, video, or text) goes in SDP "m=" as the media name.
- The Media subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" SHALL be 10000 for application/rtp-mbms-fec-symbols, and according to the rate parameter for audio/rtp-mbms-fec-tag, video/rtp-mbms-fec-tag, or text/rtp-mbms-fec-tag.
- Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the MIME media type string as a semicolon separated list of parameter=value pairs.

These payload formats is only intended to be used in declarative SDP use cases and no offer/answer negotiation procedures is defined.

8.2.1.15 Example of SDP for FEC

An example of how an SDP could look for a session containing two media streams that are FEC protected. In this example we have assumed an audiovisual stream, using 56 kbps for video and 12 kbps for audio. We further assume that we send redundant packets for the video part at 6 kbps and redundant packets for the audio part at 3 kbps. Hence, the total media session bandwidth is 56+6+12+3 = 77 kbps. In addition another 1200 bits/second of RTCP packets from the source is used for the both sessions.

The FEC encoding symbols payloads does also declare that the minimal required buffering time for either of the streams are 2.6 seconds. As the value is defined as a limitation on the sender side, further buffering to handle jitter in the transmission is also likely to be required.

```
v=0
o=ghost 2890844526 2890842807 IN IP4 192.168.10.10
s=3GPP MBMS Streaming SDP Example
i=Example of MBMS streaming SDP file
u=http://www.infoserver.example.com/ae600
e=ghost@mailserver.example.com
c=IN IP4 224.1.2.3
t=3034423619 3042462419
b=AS:77
m=video 4002 RTP/AVP 97 96 100
b=AS:62
b=RR:0
b=RS:600
a=rtpmap:96 H263-2000/90000
a=fmtp:96 profile=3;level=10
a=framesize:96 176-144
a=rtpmap: 97 rtp-mbms-fec-tag/90000
```

a=fmtp:97 opt=96; FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="
a=rtpmap: 100 rtp-mbms-fec-symbols/10000
a=fmtp:100 FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="; min-buffer-time=2600
m=audio 4004 RTP/AVP **99** 98 **101**
b=AS:15
b=RR:0
b=RS:600
a=rtpmap:98 AMR/8000
a=fmtp:98 octet-align=1
a=rtpmap: 99 rtp-mbms-fec-tag/8000
a=fmtp: 99 opt=98;FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="
a=rtpmap: 101 rtp-mbms-fec-symbols/10000
a=fmtp:101 FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="; min-buffer-time=2600

8.3 Session description

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

8.3.1 SDP Parameters for MBMS streaming session

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

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

NOTE: The use of the above parameters is according to [FLUTE/SDP].

- Extended FEC-OTI (defined below)

NOTE: The use of rtpmap is according to SDP [14].

A FEC-OTI-extension attribute is defined which results in, e.g.:

a=FEC-OTI-extension:0 A0B1C2D3E4F5

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

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

sdp-fec-oti-extension-line = "a=FEC-OTI-extension:" fec-ref SP oti-extension CRLF

fec-ref = 1*DIGIT (the SDP-internal identifier for the associated FEC-declaration).

oti-extension = base64

base64 = *base64-unit [base64-pad]

base64-unit = 4base64-char

base64-pad = 2base64-char "==" / 3base64-char "="

base64-char = ALPHA / DIGIT / "+" / "/"

9 Associated delivery procedures

9.1 Introduction

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

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

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

This specification describes these associated delivery procedures:

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

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

An instance of an ì associated procedure descriptionî 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.

9.2 Associated Procedure Description

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

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

The most recently delivered configuration file shall take priority, such that configuration parameters received prior to, and out-of-band of, the download session they apply to are regarded as ì global defaultsî, 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 îapplication/mbms-associated-procedure-parameterî identifies associated delivery procedure description instances (configuration files).

In XML, each associated delivery procedure entry shall be configured using a ìprocedureî element. All configuration parameters of one associated delivery procedure are contained as attributes of a ìprocedureî element. The ìlabelî attribute of a ìprocedureî element identifies the associated procedure to configure. The associated delivery procedure description is specified formally as an XML schema below.

9.3 File Repair Procedure

Editor's note : it was agreed in SA4 PSM SWG#6 that ptm repair will be specified. Tdoc S4-AHP156 gives an agreed basis for the specification text."

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 clientsí 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.

9.3.1 Identification of Missing Data from an MBMS Download

[Editorís 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.]

9.3.2 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.

9.3.2.1 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 ìoffset-timeî attribute.

9.3.2.2 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 ìrandom-time-periodî attribute.

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

9.3.2.3 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.

9.3.3 File Repair Server Selection

9.3.3.1 List of Server URIs

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

~~6.2.2.1.14 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.

6.2.2.1.15 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 ìnumber1.aacî 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)~~

~~6.2.2.1.16 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.~~

~~6.2.2.1.17 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.~~

~~6.2.2.1.18 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 ìapplication/simpleSymbolContainerî, 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.

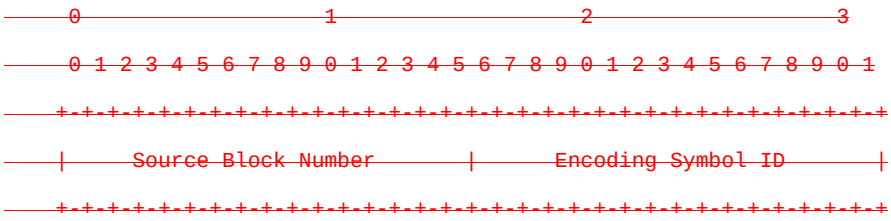


Figure 3 Example FEC Payload ID Format

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

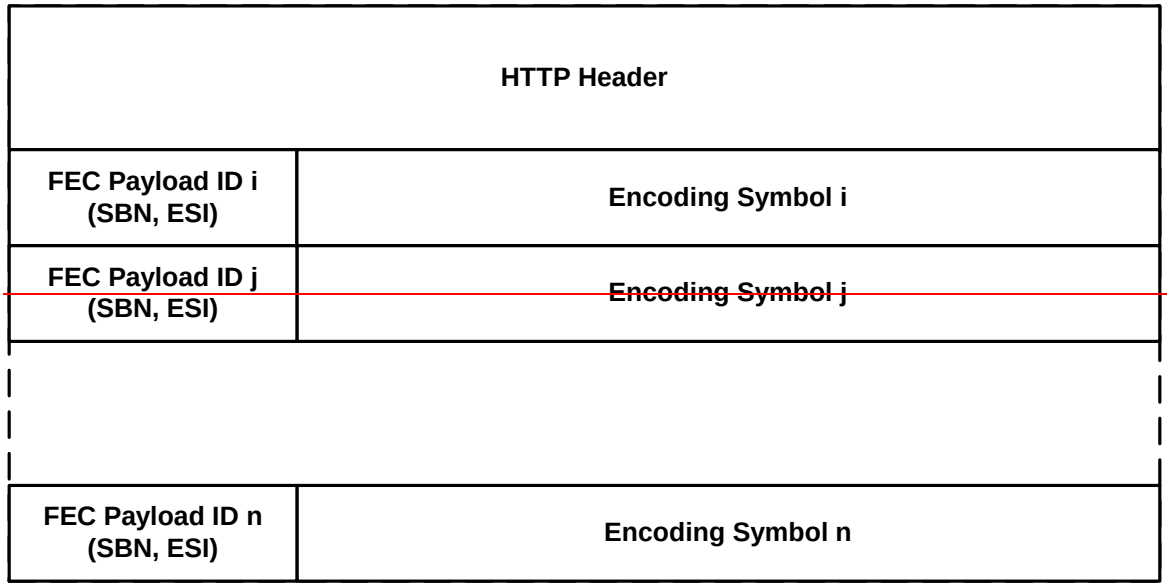


Figure 4 File Repair Response Message Format

6.2.2.1.19 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 sessionís 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 mayprovide 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 ìServerî attribute (type ìxs:anyURIî) shall only take on base URI values and not give the URI resource part.

ìLabelî value = {ìPostFileRepairî}

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 ì Sender Current Time (SCT)î field (when T=1) in the ALC/LCT header.
- R flag: The R flag is used to indicate the use of the optional ì 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.

~~AnnexC (informative):~~
~~Change history~~

~~Change history~~							
~~Date~~	~~TSG #~~	~~TSG Doc.~~	~~CR~~	~~Rev~~	~~Subject/Comment~~	~~Old~~	~~New~~
~~09-2004~~	~~25~~	~~SP-040632~~			~~Version 1.0.0~~		~~1.0.0~~

associated delivery procedure description shall be of the same type, e.g. all IP addresses of the same version, or all domain names. The number of URIs is determined by the number of ì serverURIî elements, each of which shall be a child-element of the ì procedureî element. The ì serverURIî element provides the references to the file repair server via the ì xs:anyURIî value. At least one ì serverURIî element shall be present.

9.3.3.2 Selection from the Server URI List

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

9.3.4 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 [18]. The repair request is routed to the repair server IP address resolved from the selected ì serverURIî.

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 clause 9.3.2. If there is more than one repair request to be made these are sent immediately after the first.

9.3.4.1 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) and is found from the download delivery method (the FLUTE FDT Instances describe file URIs). 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 [18] 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 [18]:

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

Where, for MBMS File Repair Request:

```
query = application "&" [ sbn_info ]

application = "mbms-rel6-flute-repair"

sbn_info = "SBN=" sbn_range *( "+" sbn_range )

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

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

esi_range = esiA [ "-" esiZ ]

sbnA = 1*DIGIT     ; the SBN, or the first of a range of SBNs

sbnZ = 1*DIGIT     ; the last SBN of a range of SBNs

esiA = 1*DIGIT     ; the ESI, or the first of a range of SBNs

esiZ = 1*DIGIT     ; 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 ìnumber1.aacî 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)

[Editorís note: future applications may define a new syntax for the query]

9.3.5 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.

9.3.5.1 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.

9.3.5.2 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 ìapplication/simpleSymbolContainerî, which denotes that the message body is a simple container of encoding symbols as described below.

[Editorís note: this Content-Type shall be IANA registered and subsequently also referenced ñ can be a separate clause dedicated to all IANA registrations]

This header is as follows:

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

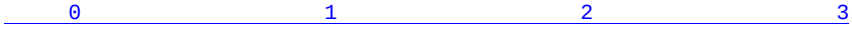
Note, other HTTP headers [18] may also be used but are not mandated by this mechanism.

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



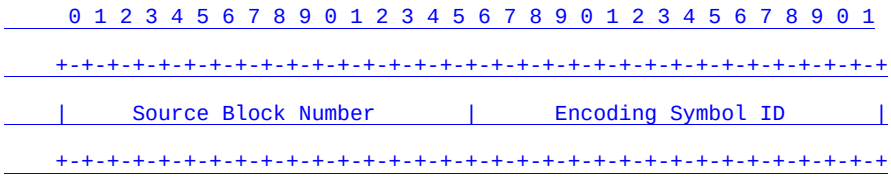


Figure 17: Example FEC Payload ID Format

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

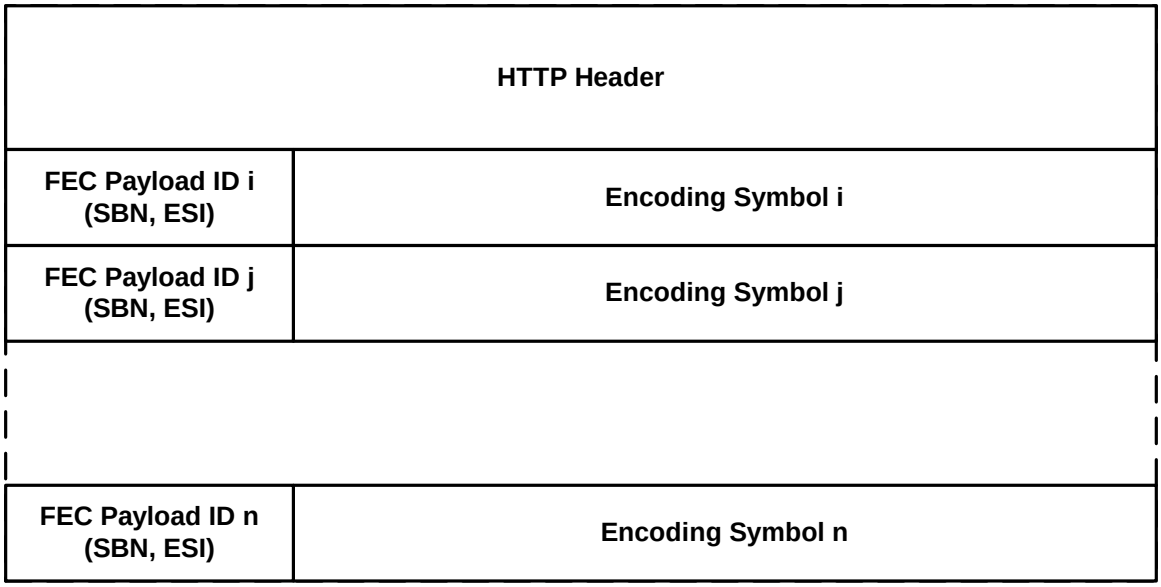


Figure 18: File Repair Response Message Format

9.3.6 Server Not Responding Error Case

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

If all of the repair servers from the serverURI list are determined to be not responding, the UE may attempt an HTTP GET to retrieve a, potentially new, instance of the sessionís 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)

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

9.4 The Reception Reporting Procedure

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

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

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

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

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

The MBMS Receiver:

4. Identifies the complete reception of a content item (e.g. a file). See clauses 9.4.1and 9.4.2.
5. Determines the need to report reception. See clause 9.4.3
6. Selects a time (Request time) at which a reception report request will be sent and selects a server from a list ñ both randomly and uniformly distributed. See clause 9.4.4 and 9.4.5.
7. Sends a *reception report request* message to the selected server at the selected time. See clause 9.4.6.

Then the server

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

9.4.1 Identifying Complete File Reception from MBMS Download

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

9.4.2 Identifying Complete MBMS Delivery Session Reception

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

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

For MBMS download sessions, FLUTE provides a "Close session flag" (see clause 7.2.5) which, when used, indicates to the UE that the session is complete.

9.4.3 Determining Whether a Reception Report Is Required

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

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

The Reception Reporting Procedure is initiated if:

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

One of the following will determine the UE behaviour:

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

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

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

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

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

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

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

9.4.4 Request Time Selection

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

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

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

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

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

In some circumstances, the system bottleneck may be in the server handling of reception reporting. In this case the *forceTimeIndependence* attribute may be used and set to true. (false is the default case and would be a redundant use of this optional attribute). When *forceTimeIndependence* is true the UE shall not use file repair point-to-point connections

to send reception reporting messages. Instead it will allow the times to expire and initiate point-to-point connections dedicated to reception report messaging.

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

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

9.4.5 Reception Report Server Selection

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

9.4.6 Reception Report Message

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

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

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

[Editorís Note: The reception report described here provides a mechanism to identify sessions in StaR and StaR-all, and files also in RAck ñ but not sessions in RAcks)]

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

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

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

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

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

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

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

- Each *fileURI* element has an optional *receptionSuccess* status code attribute which defaults to ì trueî (ì 1î) when not used. This attribute shall be used for StaR-all reports. This attribute shall not be used for StaR reports.
- [Provision of QoE Metrics using the qoeMetrics element and qoeMetricsType is pending further contribution]
- The *sessionID* attribute identified the delivery session. This is of the format source_IP_address + ì :î + FLUTE_TSI/RTP_source_port
- The *sessionType* attribute defines the basic delivery method session type used = ì downloadî || ì streamingî || ì mixedî
- The *serviceId* attribute is value and format is taken from the respective userServiceDescription serviceID definition
- The *clientId* attribute is unique identifier for the receiver. [format is FFS]

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

9.4.7 Reception Report Response Message

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

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

[Editorís note: A Reception Report response message might be used to deliver keys or as a trigger to key delivery (e.g. MSK or MTK). The use of the Reception Reporting response for this purpose is FFS.]

9.5 XML-Schema for Associated Delivery Procedures

9.5.1 Generic Associated Delivery Procedure Description

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

```
<?xml version="1.0" encoding="UTF-8"?> <xs:schema
xmlns:xs=http://www.w3.org/2001/XMLSchema elementFormDefault="qualified">

  <xs:element name="associatedProcedureDescription" type="associatedProcedureType"/>

  <xs:complexType name="associatedProcedureType">
    <xs:sequence>
      <xs:element name="postFileRepair" type="basicProcedureType" minOccurs="0" maxOccurs="1">
      <xs:element name="postReceptionReport" type="reportProcedureType" minOccurs="0" maxOccurs="1">
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="basicProcedureType">
    <xs:sequence>
      <xs:element name="serverURI" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="waitTime" type="xs:unsignedLong" use="required"/>
    <xs:attribute name="maxBackOff" type="xs:unsignedLong" use="required"/>
  </xs:complexType>

  <xs:complexType name="reportProcedureType">
    <xs:simpleContent>
      <xs:extension base="basicProcedureType">
            <xs:attribute name="samplePercentage" type="xs:string" use="optional"/>
            <xs:attribute name="forceTimingIndependence" type="xs:boolean" use="optional"/>
            <xs:attribute name="reportType" type="xs:string" use="optional"/>
```

```
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```

9.5.1.1 Use of specific value

The ìServerî attribute (type ìxs:anyURIî) shall be used to provide a fully qualified domain name (FDQN) which may be resolved to an IP address - e.g. using Domain Name Service (DNS).

9.5.1.2 Example Associated Delivery Procedure Description Instance

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

```
<?xml version="1.0" encoding="UTF-8"?>
<associatedProcedureDescription
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/mbms-associated-descrition.xsd">
  <postFileRepair
    offsetTime="5"
    randomTimePeriod="10">
    <baseURI server="http://mbmsrepair.operator.umts/"/>
    <baseURI server="http://mbmsrepair1.operator.umts/"/>
    <baseURI server="http://mbmsrepair2.operator.umts/"/>
  </postFileRepair>
  <postReceptionReport
    offsetTime="5"
    randomtimePeriod="10"
    reportType="StR-all"
    samplePercentage="100"
    forceTimingIndependence="0">
    <baseURI server="http://mbmsrepair.operator.umts/"/>
  </postReceptionReport>
</associatedProcedureDescription>
```

9.5.2 XML Syntax for a Reception Report Request

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="receptionReport">
```

```
        <xs:choice>
        <xs:element name="receptionAcknowledgement" type="rackType"/>
        <xs:element name="statisticalReport" type="starType"/>
        </xs:choice>
    </xs:element>
    <xs:complexType name="rackType">
        <xs:sequence>
            <xs:element name="fileURI" type="xs:anyURI"
                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="starType">
        <xs:simpleContent>
            <xs:element name="fileURI" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded">
                <xs:attribute name="receptionSuccess" type="xs:boolean" use="optional"/>
            </xs:element>
            <xs:element name="qoeMetrics" type="qoeMetricsType" minOccurs="0"/>
            <xs:attribute name="sessionId" type="xs:string" use="optional"/>
            <xs:attribute name="sessionType" type="xs:string" use="optional"/>
            <xs:attribute name="serviceId" type="xs:string" use="optional"/>
            <xs:attribute name="clientId" type="xs:string" use="optional"/>
            <xs:attribute name="serverURI" type="xs:anyURI" use="optional"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

[Editorís note: xs:complexType name="qoeMetricsType" is pending further contribution]

9.5.2.1 Use of Specific Values

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

9.5.2.2 Example XML for the Reception Report Request

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

<fileURI>"http://www.example.com/mbms-files/file2.3gp"</fileURI>

<fileURI>"http://www.example.com/mbms-files/file4.3gp"</fileURI>

</receptionReport>

10 Media codecs and formats

[Editorís note: This clause writes individual codec specifications.]

10.1 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.

10.2 Speech

10.3 Audio

If audio is supported, then the following two audio decoders [should/shall] be supported:

- Enhanced aacPlus [28] [29] [30]
- Extended AMR-WB [24] [25] [26]

Editor's note: SA4#33 has agreed that both audio decoders will be included here either as recommended ("should") or as mandatory ("shall").

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

Content type Bit rate	Music	Speech over Music	Speech between Music	Speech
14 kbps mono				
18 kbps stereo				
24 kbps stereo				
24 kbps mono				
32 kbps stereo				
48 kbps stereo				

10.4 Video

10.5 Still images

10.6 Text

10.7 3GPP file format

Editorís note: A container file format, e.g. as specified in S4-040699, is currently considered for MBMS Release 6. At SA4#33 in Helsinki, container formats for multimedia presentations were discussed and all participants of SA4 were encouraged to bring proposals on container file formats to SA4#34 in Lisbon for consideration in MBMS Release 6.

Annex <A> (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 ì Sender Current Time (SCT)î field (when T=1) in the ALC/LCT header.
- R flag: The R flag is used to indicate the use of the optional ì 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 <B> (normative): FEC encoder and decoder specification

Annex <C> (informative): IANA registration

This clause provides the required IANA registration

Annex <X> (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2003	SA4#28				Initial draft for SA4 #28		0.0.0
Mar 2004	SA4 PSM SWG#05				Second draft for SA4 PSM SWG#05		0.0.1
Apr 2004	SA4 PSM SWG#05				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	SA4#31				Draft for SA4 #31	0.0.2	0.0.3
May 2004	SA4#31				Changes agreed during the PSM SWG of SA4#31	0.0.3	0.0.4
August 2004	SA4 #32				Changes agreed during the PSM SWG of SA4#32: Tdoc S4-040395 with modifications Tdoc S4-040411 clause 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
August 2004	SA4 #32	S4-040521			Agreed in SA4#32 and raised as v1.0.0. Presented for information at SA#25.	0.0.5	1.0.0
October 2004	SA4 PSM SWG#6	S4-AHP137			Correction: document S4-040413 was not agreed and therefore clause 6.3.2.2 was emptied. However, the proposal to mandate ptp repair was agreed (one sentence in clause 6.3.1)	1.0.0	1.0.1
October 2004	SA4 PSM SWG#6	S4-AHP176			1-S4-AHP160 on MBMS User service architecture Agreed with several amendments: - don't specify who passes the TMGI within this specification - Figure 2 of Tdoc: add explicitly that internal BM-SC interfaces are not within the scope. Existing clause on UE client architecture was merged with the new MBMS UE clause 2- added a note in clause 6.3.2.1 on ptm repair to be specified (S4-AHP156). 3- added a note in clause 6.3.1 on content reception verification reporting (S4-AHP174) 4- added a clause and a note reflecting the agreement on RTP mechanisms for FEC (S4-AHP181 and associated documents).	1.0.1	1.0.2
November 2004	SA4 #33	S4-040676			Minor editorial	1.0.2	1.0.3
November 2004	SA4 #33	S4-040773			Addition of text proposals agreed during the PSM SWG session. 1. S4-040698: "Specification text for the MBMS Streaming Delivery method FEC framework". Agreed to put in the TS with some modifications: - temporary tables in yellow. - add a note "This scheme is intended to be generic. If the chosen FEC scheme(s) doesn't fit, it can be modified." 2. Editorial changes/corrections, checking of numbering of titles, figures and references. 3. Added definitions 4. S4-AHP172: "Delivery Method Definition". Agreed. 5. S4-040793 " Metadata for MBMS User Services ñ Specification Text " agreed. 6. S4-040794 agreed (note that xml schema in 6.3.3 was not taken into account since overruled by 795) 7. S4-040738 Agreed with updates (?). Updated text given offline by Rod Walsh.	1.0.3	1.0.4

					8. added a note on file format based on S4-040699. 8. S4-040818 "Introduction of security functionality in BMSC" (=662v2). agreed 9. S4-040795 "Delivery Confirmation Procedure" agreed.		
November 2004	SA4 #33	S4-040835			Reorganization of chapters (without revision marks): Addition of annex B and C. Addition of text agreed during SA4#33 plenary about audio codecs. Several editorials corrections.	1.0.4	1.1.0
November 2004	SA4 #33	S4-040859			- Correction of version number to reflect the substantial changes. - Removal of one sentence in annex B. - revision marks indicated changes since v1.0.0 (S4-040521) To be presented for information to 3GPP TSG SA#26. - removal of automatic figure numbering	1.1.0	1.5.0

3GPP TS 26.346 ~~V1.0.0 (2004-09)~~V1.5.0 (2004-

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)

- 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> MBMS, FLUTE, RTP, FEC, streaming, download, broadcast, multicast

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:

ï Broadcast Mode

ï 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: ì Vocabulary for 3GPP Specificationsî .

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

[3] 3GPP TS 22.246: ì MBMS User Servicesî .

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

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

[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 ñ 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: ì Asynchronous Layered Coding (ALC) Protocol Instantiationî , M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, J. Crowcroft, December 2002.

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

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

[13] IETF RFC 3695: ì Compact Forward Error Correction (FEC) Schemesî , 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)"

[20] 3GPP TS 33.246 [TBA].

[21] UML [TBA].

[22] XML Schema Part 2: Datatypes Second Edition, W3C Recommendation 28 October 2004

[23] ABNF reference [TBA]

[24] 3GPP TS 26.290: ì Extended AMR Wideband codec; Transcoding functionsî

[25] 3GPP TS 26.304: ì ANSI-C code for the Floating-point; Extended AMR Wideband codecî

[26] 3GPP TS 26.273: ì ANSI-C code for the Fixed-point; Extended AMR Wideband codecî

[27] Real-Time Transport Protocol (RTP) Payload Format for Extended AMR Wideband (AMR-WB+) Audio Codec, draft-ietf-avt-rtp-amrwbplus-01.txt

[28] 3GPP TS 26.401: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; General description".

[29] 3GPP TS 26.410: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; Floating-point ANSI-C code".

[30] 3GPP TS 26.411: "General audio codec audio processing functions; Enhanced aacPlus general audio codec; Fixed-point ANSI-C code".

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: ì Multimedia Broadcast/Multicast Serviceî [2].

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

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

Multicast joining: See TS 22.146: ì Multimedia Broadcast/Multicast Serviceî [2].

Multicast session: See TS 22.146: ì Multimedia Broadcast/Multicast Serviceî [2].

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

MBMS user services: See TS 22.246: ì Multimedia Broadcast/Multicast Service: User Serviceî [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. An MBMS delivery method uses MBMS bearers in delivering content and may make use of associated procedures.

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 ñ Bearers, Delivery method and User service. Figure 1 depicts these layers with examples of bearer types, delivery methods and applications.

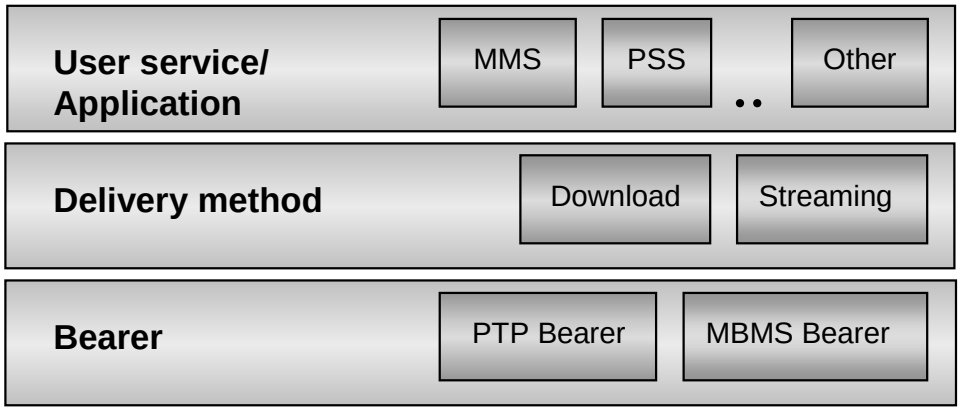


Figure ~~1:~~1: Functional Layers for MBMS User Service

– **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.~~associated delivery procedures such as file-repair, delivery verification. Two delivery methods are defined, namely download and ~~streaming. MBMS delivery may utilize both~~streaming. Delivery methods may be added beyond release 6. Delivery methods may use MBMS bearers and ~~PTP bearers.~~may make use of point-to-point bearers through a set of MBMS associated procedures.

– **User service**. The MBMS User service enables applications. Different application impose different requirements when delivering content to MBMS subscribers and may use different MBMS delivery methods. As an example a messaging application such as MMS would use the download delivery method while a streaming application such as PSS would use the streaming delivery method.

4.2 MBMS User Service Entities

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

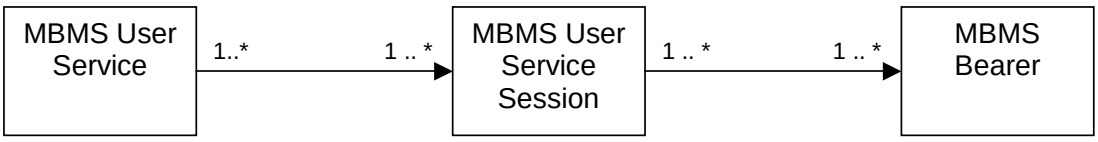


Figure ~~2:~~2: Entities and Relations

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~~clause 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

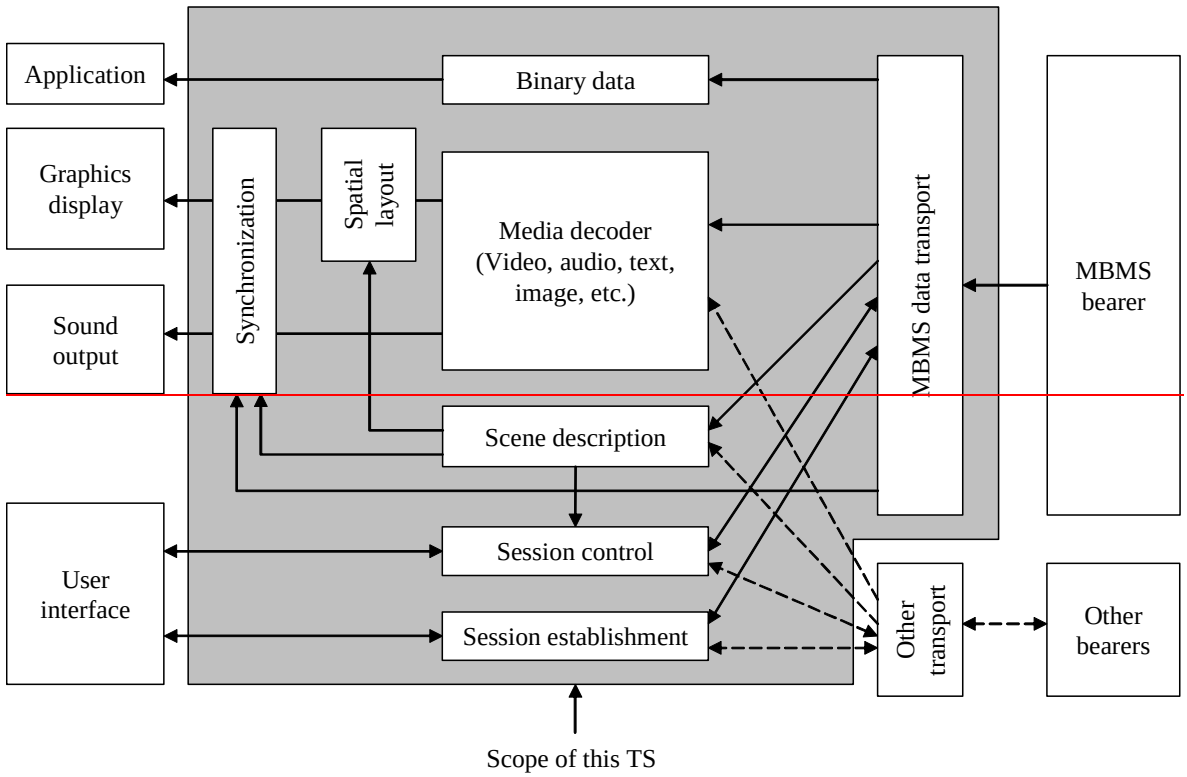
- The BM-SC;
- The GGSN;
- The UE.

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.~~Functional Entities to support MBMS User Services

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

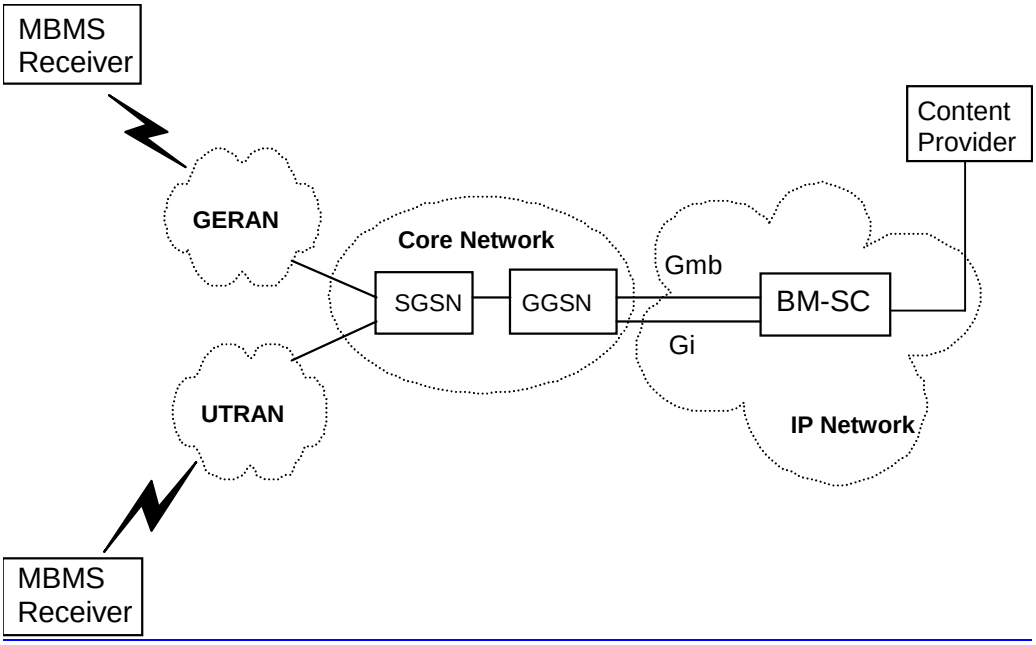


Figure 3: MBMS network architecture model

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

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

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

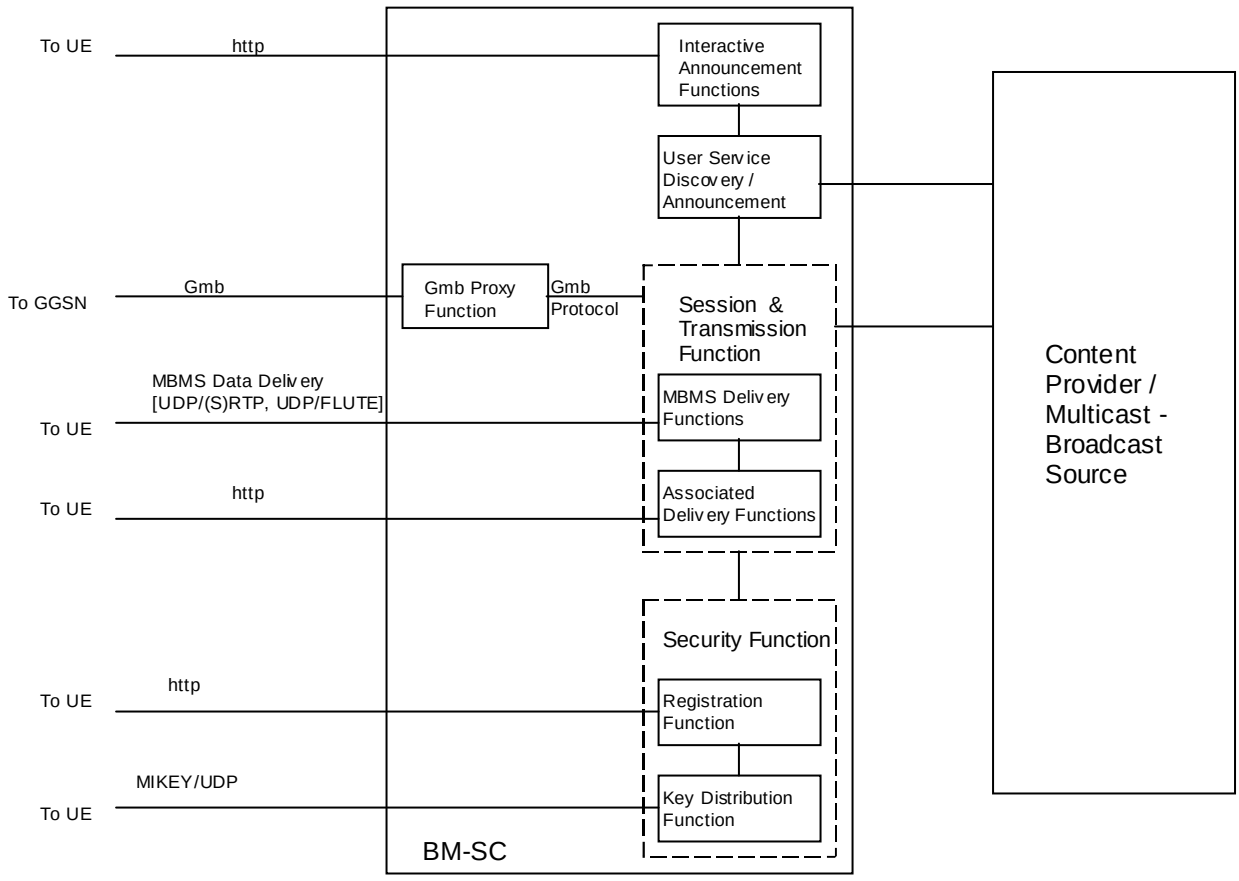


Figure 4: BM-SC sub-functional structure

[Editorís note: The http interaction between the MBMS UE and the User Service Discovery / Announcement function in is outside the scope of this specification]

[Editorís note: note that this picture highlights those BM-SC functional elements which are relevant for the SA4 work. For instance, the Security Function depicted here is part of the more general Membership function found in Fig. 5a of TS 23.246. Similar, the User Service Discovery/Announcement and Interactive Announcement functions are part of the more general Service Announcement function found in Fig. 5a of TS 23.246. Better consistency with Fig. 5a of TS 23.246 will be achieved during the final cleanup of this specification]

The Gmb-Proxy function relays the Gmb protocol between the Session function and the GGSN.

[Editorís note: SA2 is currently discussing the Gmb Proxy in the architecture. It is currently unclear whether the Gmb Proxy should be a BM-SC sub-function or belongs to an entity between a GGSN and a number of BM-SC functions. It is also unclear whether the Gmb Proxy would be an optional functionality]

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

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

- User Service Discovery / Announcement providing service description material to be presented to the end-user as well as application parameters used in providing service content to the end-user
- MBMS-based delivery of data/content (optionally confidentiality and/or integrity protected) from the BM-SC to the UE over IP multicast.
- Key Request and Registration procedure for receiving keys and key updates.

- Key distribution procedures whereby the BM-SC distributes key material required to access service data and delivered content.

- Associated Delivery functions are invoked by the UE in relation to the MBMS data transmission. The following associated delivery functions are available:

    - Point-to-point repair for download delivery method used to complement missing data using point-to-point sessions.

    - Delivery verification and reception statistics collection procedures

The interfaces between internal BM-SC functions are outside the scope of this specification.

4.4.1 Content Provider / Multicast Broadcast Source

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

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

4.4.2 MBMS Security Function

MBMS user services may use security functions for integrity and/or confidentiality protection of MBMS data. The MBMS security function is used for distributing MBMS keys (Key Distribution Function) to authorized UEs. UEs request the initial keys from the BM-SC and register (using the Registration Function) to receive key updates by key management procedures. Detailed description of the security functions is provided in [20] (TS 33.246).

[Editorís Note: The MBMS Security function is to be confirmed by SA3]

4.4.3 MBMS Session and Transmission Function

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

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

The BM-SC Session and Transmission function is further described in later clauses of this specification as well as in [4] (TS 23.246).

If security functions are activated for the MBMS User Service, the confidentiality and/or integrity protection is applied by the BM-SC to outgoing MBMS data transmissions. The traffic protection is applied between the BM-SC and the UEs. The security is based on symmetric keys, which are shared between the BM-SC and the UEs accessing the service. For further details on traffic protection see [20] (TS 33.246).

[Editorís Note: The MBMS Security function is to be confirmed by SA3].

4.4.4 Gmb Proxy function

The Gmb Proxy function relays the Gmb protocol and may fill in the MBMS bearer service oriented attributes. The BM-SC Proxy function is described in [4] (TS 23.246).

[Editorís note: SA2 is currently discussing the Gmb Proxy in the architecture. It is currently unclear whether the Gmb Proxy should be a BM-SC sub-function or belongs to an entity between a GGSN and a number of BM-SC functions. It is also unclear whether the Gmb-Proxy would be a mandatory functionality]

4.4.5 User Service Discovery / Announcement function

The User Service Discovery / Announcement provides service description information, which may be delivered via an MBMS bearer or via the interactive announcement function.

[Editorís note: Description of the User Service Description / Announcement function in this clause is ffs]

4.4.6 Interactive Announcement Function.

~~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~~An Interactive Announcement Function may offer an alternative means to provide service descriptions to the UE, e.g. using HTTP. The specification of this function is 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~~4.4.7 MBMS UE

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

In case the MBMS user service is secured, the UE needs one or more cryptographic MBMS service keys, therefore the UE requests the relevant cryptographic MBMS service keys using the MBMS security functions. The received keys are then used for securing the current session.

[Editorís Note: The MBMS Security function is to be confirmed by SA3].

5 Procedures and Protocol

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

5.1 Introduction

~~5.2 User service discovery/announcement~~This clause specifies the MBMS User service procedures and protocols.

5.2 User Service Discovery/Announcement

5.2.1 Introduction

User service discovery refers to methods for the UE to obtain a list of available MBMS user services 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.~~clause 0.

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.~~clause 5.2.3. Other user service announcement and discovery mechanisms by other means than the download delivery method are out of scope of the present specification.

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

~~User service announcement over a MBMS bearer~~

5.2.2 ~~5.2.2.1 Introduction~~MBMS User Service Description metadata fragments

MBMS ~~Service Announcements are~~User Service Discovery/ Announcement is 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.~~clause 7 or using interactive announcement functions.

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

The metadata consists of:

- a metadata fragment object describing details of MBMS user services.
- ~~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.~~service sessions
- a metadata fragment object(s) describing details of Associated delivery methods
- a metadata fragment object(s) describing details of service protection

Metadata management information consists of:

~~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

5.4Protocols

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

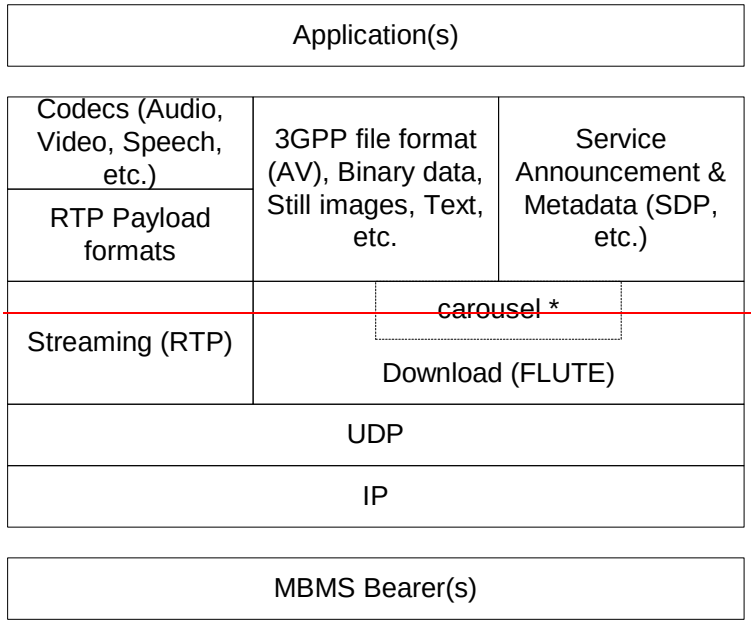


Figure 2: Protocol stack view of the MBMS User Services over MBMS bearer(s)

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.

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

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

Figure 5: Simple Description Data Model

Figure 5 illustrates the simple data model relation between these description instances using UML [21] for a single User Service Description (note, ì Nî means any number in each instance). One MBMS User Service Description instance shall include at least one delivery method description instance. The delivery method description shall refer to one session description instance.

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

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

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

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

5.2.2.1 Session Description

One or more session descriptions are contained in one session description object. The session description instance shall be formatted according to the session description protocol (SDP). Each session description instance shall contain only descriptions for only one RTP Streaming session or FLUTE Download session. One session description may describe both one download and one streaming session. The *sessionDescriptionURI* references the session description object. The session description is specified in clause 7.3for the MBMS download delivery method and in clause 8.3.16.2.3.1 for the MBMS streaming delivery method.

5.2.2.2 Associated Delivery Procedure Description

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

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

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

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

5.2.2.3 Service Protection Description

[Editorís note: this clause will be completed when the overall security functions are clear]

5.2.2.4 XML-Schema for MBMS User Service Description

The root element of the MBMS user service description is the *userServiceDescription* element. The element is of type *userServiceDescriptionType*.

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

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

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

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

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

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

```
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema elementFormDefault="qualified"

   targetNamespace="urn:3gpp:metadata:2004:userservicedescription"

   xmlns="urn:3gpp:metadata:2004:userservicedescription"

   xmlns:xs="http://www.w3.org/2001/XMLSchema">


   <xs:element name="userServiceDescription" type="userServiceDescriptionType"/>


   <xs:complexType name="userServiceDescriptionType">

      <xs:sequence>

         <xs:element name="name" type="nameType" minOccurs="0"

                     maxOccurs="unbounded"/>

         <xs:element name="serviceLanguage" type="xs:language" minOccurs="0"

                     maxOccurs="unbounded"/>

         <xs:element name="deliveryMethod" type="deliveryMethodType"
```

```
                    maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="serviceId" type="xs:anyURI" use="required"/>
    </xs:complexType>

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

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

</xs:schema>
```

5.2.2.5 Example MBMS User Service Description Instances

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

```
<?xml version="1.0" encoding="UTF-8"?>
<userServiceDescription
    xmlns="www.example.com/3gppUserServiceDescription"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    userServiceId="urn:3gpp:0010120123hotdog">
      <deliveryMethod
          sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>

</userServiceDescription>
```

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

```
<?xml version="1.0" encoding="UTF-8"?>
<userServiceDescription
  xmlns="www.example.com/3gppUserServiceDescription"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  serviceId="urn:3gpp:1234567890coolcat">
    <name lang="EN">something in english</name>
    <name lang="DE">something in german</name>
    <name lang="FR">something in french</name>
    <name lang="FI">something in finnish</Name>
    <serviceLanguage>EN</serviceLanguage>
    <serviceLanguage>DE</serviceLanguage>
    <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session1.sdp"/>
    <deliveryMethod
      sessionDescriptionURI=http://www.example.com/3gpp/mbms/session2.sdp
        associatedProcedureDescriptionURI="http://www.example.com/3gpp/mbms/procedureX.xml"/>
    <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session3.sdp"
        associatedProcedureDescriptionURI="http://www.example.com/3gpp/mbms/procedureY.xml"/>
    <deliveryMethod
      sessionDescriptionURI="http://www.example.com/3gpp/mbms/session4.sdp"
</userServiceDescription>
```

5.2.3 User service announcement over a MBMS bearer

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

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

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

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

5.2.3.1 Supported Metadata Syntaxes

The MBMS metadata syntax supports the following set of features:

- Support of carriage of SDP descriptions, and SDP is expected to sufficiently describe at least: MBMS Streaming sessions and, MBMS download sessions;

- Support for multiple metadata syntaxes, such that the delivery and use of more than one metadata syntax is possible;
- Consistency control of metadata versions, between senders and receivers, independent of the transport and bearer use for delivery;
- Metadata fragments are identified, versioned and time-limited (expiry described) in a metadata fragment syntax-independent manner (which is a consequence of the previous two features).

5.2.3.2 Consistency control and Syntax Independence

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

5.2.3.3 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.

5.2.3.4 Delivery of the Metadata Envelope

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

5.2.3.5 Metadata Envelope Transport

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

5.2.3.6 Metadata Envelope and Metadata Fragment Association with FLUTE

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

5.3 User Service Initiation/Termination

5.3.1 Initiation

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

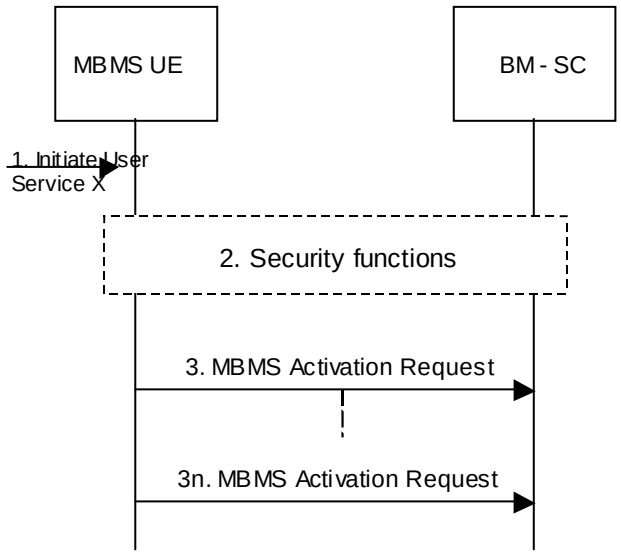


Figure 6: Initiation of an MBMS User Service

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

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

[Editorís Note: The MBMS Security function is to be confirmed by SA3].

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

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

The MBMS user service activation consists of (not necessarily in this order):

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

[Editor's Note: : Details on the security parameters are FFS]

[Editor's Note: Some more information is required on the MBMS User Service Description to finalize the MBMS user service initiation clause]

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

- Unique Service Identifier: MBMS operator specific
- Service Name: language tagged concise name - which may be human readable. (Optional - zero or more).
- Service Language: list of those provided for the service. (Optional - zero or more).
- One or more delivery method elements, each with:
    - Session Description Pointer: URI identifier of the session description (SDP file)
    - Associated Delivery Procedure Description Pointer: URI identifier of the relevant ìassociated delivery procedure descriptionî instance for this session. (More than one procedure may be listed from within the identified description, including file repair and reception reporting procedures). (Optional) If this is not used, associated delivery procedures are ìoffî for the related session
    - Security Description: URI identifier of the security description (SDP file). (Optional) If this is not used, the security procedures are ìoffî for the related session
- Session start/stop time: start time, end time (SDP)
- Session Identifier: (Source IP address + Transport Session Identifier for FLUTE) (SDP)
- Data rates: bandwidth and modifiers (SDP)
- IP addressing and ports: Destination IP address, UDP destination port (SDP)
- Protocol ID: RTP/UDP for streaming, FLUTE/UDP for download (SDP)
- Media types and codecs: media type and fmt-list (SDP)
- Bearer mode: Multicast/Broadcast bearer mode per media (SDP)

- Component language: language per media (SDP)

- FEC on/off, related parameters: FEC instance numbers as capability requirement (SDP)

NOTE: The list of languages that the service is named in (Service Name) may differ for the list of service languages the service is provided in (Service Language).

[Editor's note: RTP session Id is To Be Decided]

5.3.2 MBMS User Service termination procedure

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

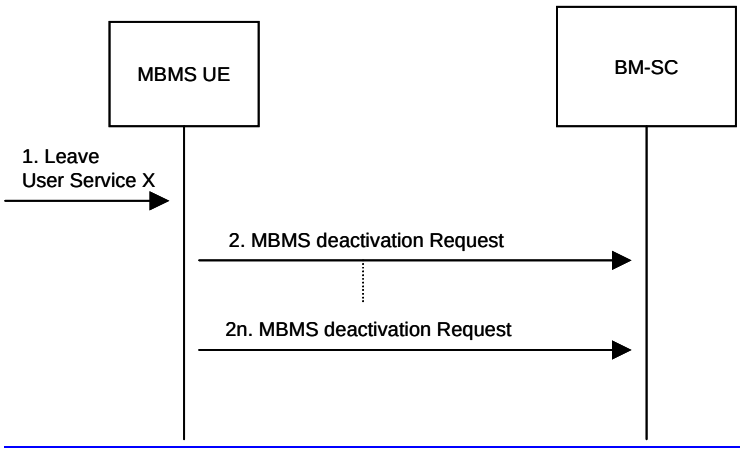


Figure 7: Termination of an MBMS user service

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

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

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

5.4 MBMS Data Transfer Procedure

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

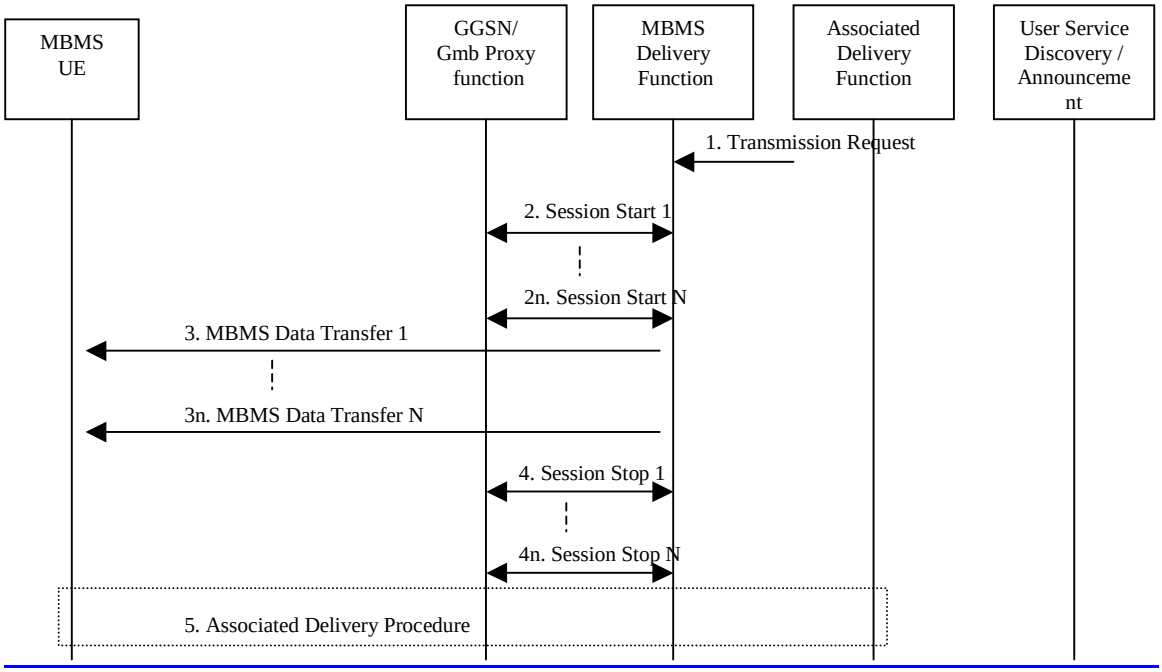


Figure 8: Procedure of MBMS Data Transfer

[Editorís Note: security related interactions are not yet depicted in the sequence]

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

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

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

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

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

[Editorís Note: The MBMS Security function is to be confirmed by SA3]

5.5 MBMS Protocols

Figure 9 illustrates the protocol stack used by MBMS User services.

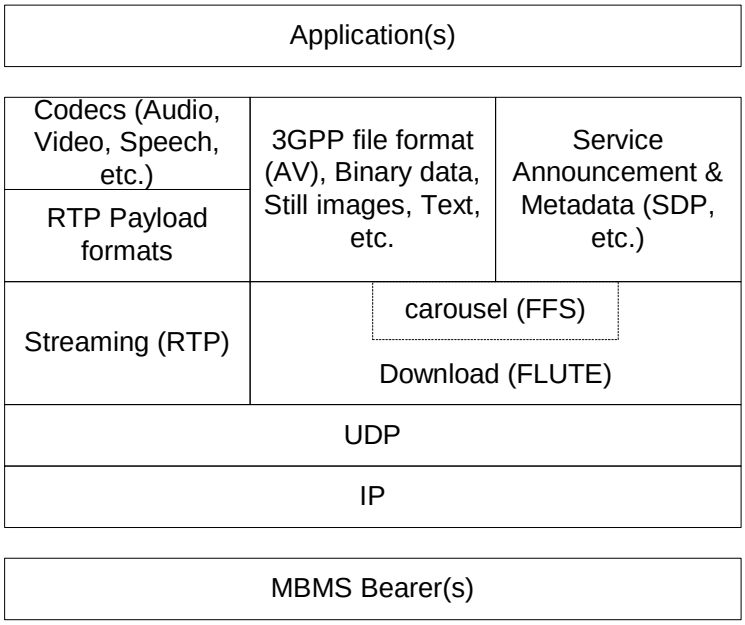


Figure ~~3: Building block structure of FLUTE~~

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

6 Introduction on Delivery Methods

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

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

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

7 Download Delivery Method

7.1 Introduction

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

FLUTE is built on top of the Asynchronous Layered Coding (ALC) protocol instantiation [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 ~~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 ñ the File Description Table (FDT) Instances ñ 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 ì Sender Current Time (SCT)î field (when T=1).
- The R flag shall indicate the use of the optional ì Expected Residual Time (ERT)î 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.

6.2.2.1.1 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 (ìa=source-filter:î) [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.

6.2.2.1.2 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.

6.2.2.1.3 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 ìconnection dataî field (ìc=î) of SDP [14, 15]. The destination port number shall be defined according to the <port> sub-field of the media announcement field (ìm=î) of SDP [14, 15].

The presence of a FLUTE session on a certain channel shall be indicated by using the ëm lineí 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.

6.2.2.1.4 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 = 1*aî 1 =î 1 flute-tsiî 1 :î 1 integer CRLF*

integer = as defined in [14]

6.2.2.1.5 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.

6.2.2.1.6 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 (ît=î) [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 Sessiondescription

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 deliveredan 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 ìassociated procedure descriptionî 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 ìglobal defaultsî, 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 îapplication/mbms-associated-procedure-parameterî identifies associated delivery procedure description instances (configuration files).

In XML, each associated delivery procedure entry shall be configured using a ìprocedureî element. All configuration parameters of one associated delivery procedure are contained as attributes of a ìprocedureî element. The ìlabelî attribute of a ìprocedureî 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 clientsí 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.

6.2.2.1.7 6.3.2.1.1 Identification of Missing Data from an MBMS Download

6.2.2.1.8 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.

6.2.2.1.9 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 ìoffset-timeî attribute.

6.2.2.1.10 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 ìrandom-time-periodî attribute.

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

6.2.2.1.11 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.

6.2.2.1.12 6.3.2.1.3 File Repair Server Selection

6.2.2.1.13 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 theassociated delivery procedure

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 10 for an illustration of FLUTE building block structure. FLUTE is carried over UDP/IP, and is independent of the IP version and the underlying link layers used.

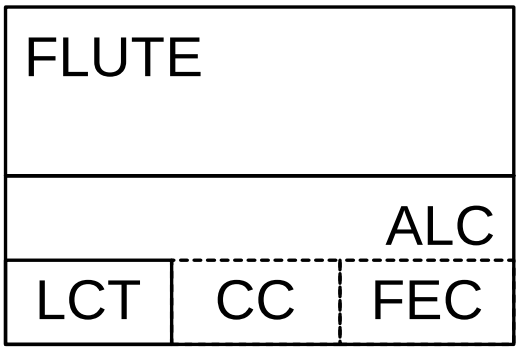


Figure 10: Building block structure of FLUTE

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

7.2 FLUTE usage for MBMS download

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

In 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.

7.2.1 Fragmentation of Files

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

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

[Editorís note: The limitation of FLUTE packet payload length is still under discussion]

7.2.2 Symbol Encoding Algorithm

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

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

7.2.3 Blocking Algorithm

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

7.2.4 Congestion Control

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

7.2.5 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 ì Sender Current Time (SCT)î field (when T=1).
- The R flag shall indicate the use of the optional ì Expected Residual Time (ERT)î 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.

[Editorís note: This clause is still under discussion]

7.2.6 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.

[Editorís note: This clause is still under discussion]

7.2.7 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.

[Editorís note: This clause is still under discussion.]

[Editorís note: There may be other FDT extension parameters to include]

7.3 SDP for Download Delivery Method

7.3.1 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 [23] (ABNF reference).

7.3.2 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 per media;
- The Transport Session Identifier (TSI) of the session;
- The start time and end time of the session;
- The protocol ID (i.e. FLUTE/UDP);
- Media type(s) and fmt-list;
- Data rate using existing SDP bandwidth modifiers;
- Mode of MBMS bearer per media;
- FEC capabilities and related parameters;
- Service-language(s) per media.

This list includes the parameters required by FLUTE [9]

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

7.3.2.1 Sender IP address

There shall be exactly one IP sender address per MBMS download session, and thus there shall be exactly one IP source address per complete MBMS download session SDP description. The IP source address shall be defined according to the source-filter attribute (ì a=source-filter:î) [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.

7.3.2.2 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.

7.3.2.3 Destination IP address and port number for channels

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

- IP destination address
- Destination port number.

The IP destination address shall be defined according to the ì connection dataî field (ì c=î) of SDP [14, 15]. The destination port number shall be defined according to the <port> sub-field of the media announcement field (ì m=î) of SDP [14, 15].

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

m=application 12345 FLUTE/UDP 0

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

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

7.3.2.4 Transport Session Identifier (TSI) of the session

The combination of the TSI and the IP source address identifies the FLUTE session. Each TSI shall uniquely identify a FLUTE session for a given IP source address during the time that the session is active, and also for a large time before and after the active session time (this is also an LCT requirement [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 = ì aî ì =î ì flute-tsi*î î :î integer CRLF*

integer = as defined in [14]

7.3.2.5 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.

7.3.2.6 Session Timing Parameters

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

7.3.2.7 Mode of MBMS bearer per media

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

a=mbms-mode:broadcast 1234

Where any multicast mode media are described, the MBMS bearer mode declaration attribute shall not be used at session level and shall not be used at media level for multicast mode media.

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

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

tmgi = 1*DIGIT

7.3.2.8 FEC capabilities and related parameters

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

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

This can be session-level (and so the first instance (fec-ref=0) becomes the default for all media) and media-level to specify differences between media. This is optional as the information will be available elsewhere (e.g. FLUTE FDT Instances). If this attribute is not used, and no other FEC-OTI information is signaled to the UE by other means, the UE may assume that support for FEC id 0 is sufficient capability to enter the session.

A new FEC-declaration attribute shall be defined which results in, e.g.;

a=FEC:0

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

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

sdp-fec-declaration-line = "a=FEC-declaration:" fec-ref SP fec-enc-id ";" [SP fec-inst-id] CRLF

fec-ref = 1*DIGIT (value is the SDP-internal identifier for FEC-declaration).

fec-enc-id = "encoding-id=" enc-id

end-id = 1*DIGIT (value is the FEC Encoding ID used).

fec-inst-id = "instance-id=" inst-id

inst-id = 1*DIGIT (value is the FEC Instance ID used).

sdp-fec-line = "a=FEC:" fec-ref CRLF

fec-ref = 1*DIGIT (value is the FEC-declaration identifier).

7.3.2.9 Service-language(s) per media

The existing SDP attribute ìa=langî is used to label the language of any language-specific media. The values are taken from RFC 3066 which in turn takes language and (optionally) country tags from ISO639 and 3661 (e.g. "a=lang:EN-US"). These are the same tags used in the User Service Description XML.

7.3.3 SDP Examples for FLUTE Session

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

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

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

8 Streaming delivery method

8.1 Introduction

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

8.2 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.

[Editorís note: it was agreed that RTCP feedback could be useful in future releases]

8.2.1 FEC mechanism for RTP

[Editor's note: this scheme is intended to be generic. If the chosen FEC scheme(s) doesn't fit, it can be modified]

The generic mechanism for systematic FEC of RTP streams consists of two RTP payload formats, one for FEC source packets, one for FEC protection data, and their related signaling. In addition a FEC algorithm definition is needed including source block construction, definition of one or more FEC payload ID (FPID) and Object Transmission Information (OTI). A receiver supporting this delivery method shall support the FEC payload format for source RTP packets and shall support the payload format for FEC symbols. The two RTP payload formats interact to create the FEC mechanism. The concept of the protection mechanism is briefly discussed below. More details can be found in the following sub-clauses.

Figure 11 depicts the interaction between the different protocol implementations of a sender and receiver. On the sender side, an original RTP packet has been created. The original packet consists of an RTP payload and the RTP header including the RTP header fields specified by the payload (timestamp, payload type, marker bit). An original packet has been processed to generate the RTP header fields, i.e. assigned SSRC and sequence number, CRSRC list, etc. If the packet were sent without FEC protection it would be forwarded for RTP processing. RTP processing is defined here as the activities that the RTP stack performs to gather RTCP statistics (bytes and packet sent) and optionally perform padding, i.e. operations that are dependent on the packet size. For original RTP packets that are going to be FEC protected, the RTP packet is encapsulated into the FEC payload format for source RTP packets. This encapsulation consists of the addition of a FEC payload ID field prior to the original payload. The RTP header is maintained with the exception of the substitution of the original RTP payload type (indicating the type of media) to the FEC source payload type, so to identify the use of the FEC source payload format. The FEC payload ID data is provided by the FEC encoder. The completed FEC source RTP packet is forwarded for RTP processing thus ensuring correct RTCP statistics.

In addition of being scheduled for transmission, a copy of the complete RTP packet is also processed by the FEC encoder. The FEC encoder starts with mirroring the de-capsulation operation that will be performed by the receiver after the reception of a source packet. The FEC de-capsulation implements the steps necessary to reconstruct the original RTP packet that would have been sent had it not been FEC protected. More precisely, the de-capsulation substitutes the payload type value of the FEC payload format with the payload type value of the source payload format, and removes the FEC payload ID field. The FEC decoder takes this reconstructed original RTP packet and places it into its encoding source block in accordance with the FEC payload ID data, see clause 8.2.1.7.

After processing all original RTP packets to be protected, the FEC encoder generates the desired amount of FEC protection data, i.e. encoding symbols. These encoding symbols are packetized in the FEC protection data payload format and sent to the receiver as RTP packets. The protection data RTP packets shall be sent using an SSRC different from the source RTP packetís SSRC, but within the same RTP session. Doing so avoids non-continuous sequence numbering spaces for both the FEC protection and the source media RTP packet streams.

The receiver forwards the received source packets directly to the media receiver buffer, after reception and de-capsulation of the FEC source payload format as discussed above. A copy of (or reference to) the reconstructed original RTP packet is also processed by the FEC decoder. The FEC decoder places the original RTP packet into the source block in accordance with the source packet's FEC payload ID. When receiving protection data packets, the receiver depacketizes the encoding symbols and uses them if necessary to decode missing portions of the source block. If any source RTP packets have been lost, and sufficient source and protection data packets have been received, FEC decoding can be performed. If complete original RTP packets are recovered, they are forwarded to the media receiver buffer, and for further media processing.

Note that the RTP receiver buffer, must have enough depth to buffer all the original packets and allow time for the protection packets to arrive and decode missing packets before any of these packets are consumed by the media decoder.

The FEC payload ID (Source and Protection) are used to associate the source RTP packets and protection RTP packets, respectively, to a source block. The FEC payload ID are part of the definition of the FEC scheme and is identified by the FEC encoding ID and instance ID for underspecified FEC encoding IDs. One FEC encoding ID for the streaming delivery method is specified in clause 8.2.1.6. Any FEC encoding IDs using the defined RTP payload formats shall be a systematic FEC code and may use two different FEC payload IDs. If two FEC payload ID is used, the FEC Encoding ID shall define which FEC payload ID format shall be used in the source and protection data RTP payload format respectively.

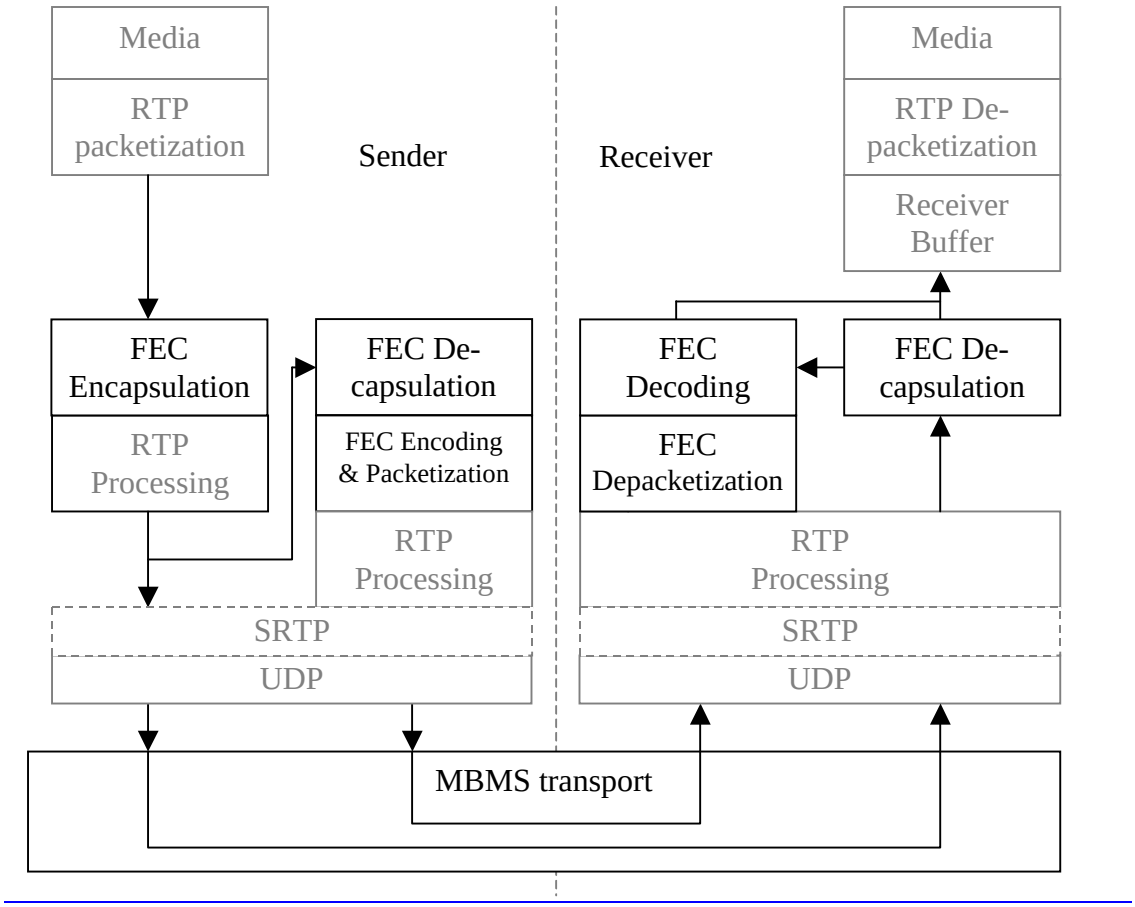


Figure 11: FEC mechanism for RTP interaction diagram

8.2.1.1 Sending Terminal Operation

A sender may send RTP packets that are not FEC protected (original RTP packets), and shall then use the original PT assigned to payload format and bypass the below defined operation. Original RTP packets that are going to be FEC protected shall use the below procedure.

A sender performs the following operations:

1. Start with an original RTP packet complete with RTP header and payload, including profile specific extensions, and variable sized fields as CSRC list and extension headers.
2. The sender reserves the position and the amount of space needed for the original RTP packet information in the FEC code's source block. In doing so, the FPID information can be determined. See Clause 8.2.1for details.
3. The source RTP packet is created by inserting the FPID after the RTP header and before the payload header.
4. To allow the receiver to identify the original RTP payload type and indicate that the FPID is present, the RTP PT is changed from its original value to the value that indicate the combination of original payload format and the inserted FPID.
5. The source RTP packet generated is forwarded for RTP processing where RTCP statistics are gathered.
6. After RTP processing, a copy of the packet being sent to the receiver(s) is processed to reconstruct the original RTP packet. In particular, the FPID is removed, and the RTP payload type (that indicated the insertion of the FPID in the original RTP packet) is replaced with the original PT.
7. The resulting RTP packet is placed in the correct source block and in the position determined in step 2 above.

8. When a source block is complete, the FEC encoder generates encoding symbols and places these symbols into protection RTP packets, to be conveyed using normal RTP mechanisms. The SSRC for the protection packets are used in these transmissions.

8.2.1.2 Receiving Terminal Operation

A receiver operates as follows, when receiving a packet:

1. If an RTP packet is not FEC encapsulated as indicated by the payload type, it is forwarded to the receiver buffer and the operation ends.
2. If a source RTP packet with a payload type that indicates that it is FEC encapsulated is received:
    a. The FPID is removed from the RTP packet and stored.
    b. The RTP payload type is replaced with the original PT based on the received PT.
    c. The original packet is forwarded to RTP receiving entity, in Figure 11 the receiver buffer. A copy is forwarded together with the FPID to the FEC decoder. As an alternative, a reference to the receiver buffer together with the FPID may be used.
    d. FEC decoder places the RTP packet into the source block according to the FPID.
3. If a protection RTP packet is received as indicated by the payload type, the encoding symbols it contains are depacketized and passed on to the FEC decoder.
4. If all source packets have been received no FEC decoding is necessary.
5. If some source RTP packets are missing as detected by the gaps in the sequence number space, then the FEC decoder determines if it has received enough source and protection packets to decode and performs the decoding operation.
6. Any missing source RTP packets that were reconstructed during the decoding operation can be forwarded directly to the RTP receiving entity, i.e. receiver buffer.

It should be understood that the RTP receiving entity will require some form of buffering and packet re-ordering so to insert any reconstructed packets into their appropriate sequencing. The amount of buffering depends primarily on the media decoder implementation, and its tolerance to do decoding on reordered packets, and the (maximum) size of the source block. For media requiring decoding in order, the minimal delay can be directly derived as the maximum time duration represented by the RTP packets allowed in each source block. To allow receivers to determine the minimal buffering requirement, the repair RTP packets payload formats has a parameter "min-buffer-time" as defined in 8.2.1.11.

8.2.1.3 RTP Payload format for source RTP packets

The FEC payload format for source RTP packets shall be used to add a FEC payload ID field to an original RTP packet that is going to be FEC protected. This enables the FEC sender and receiver to correctly place the source (original) information into its FEC encoding and decoding process respectively. The FEC algorithm specific addressing of any source information is provided in the FEC payload ID. This payload format is identified by the media type defined in clause 8.2.1.13.

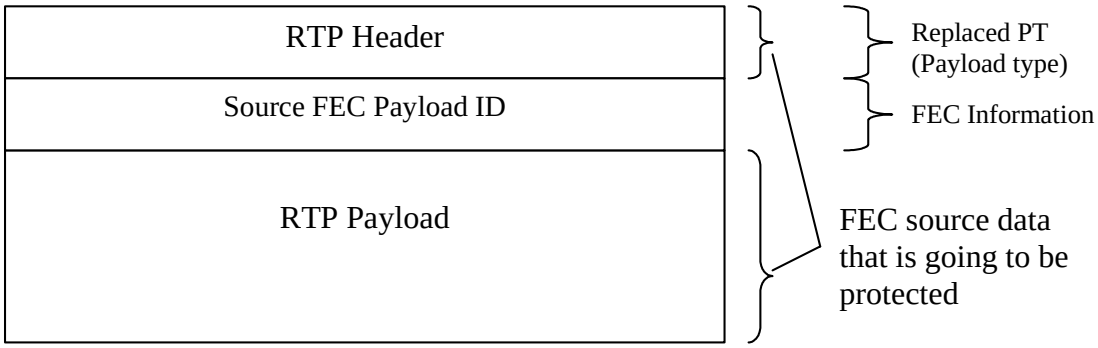


Figure 12: Principle of FEC payload format for source RTP packets

Figure 12 depicts the layout of a FEC payload packet. Into the original RTP packet and between the RTP header and the RTP payload (including any payload specific headers), the FEC payload ID is inserted. In addition, the Payload Type field in the RTP header shall be changed to indicate the presence of the Source FEC Payload ID. No other changes to the RTP header shall be done. This forms the RTP packet to be sent over the wire. For the FEC calculation, however, the reconstructed original RTP packet, is used in the source block as source information.

The reconstruction of the original RTP packet is done in two simple steps:

1. Replace the RTP header payload type (PT) field with the original payload type value as indicated by the "opt" parameters value for the actually present PT.

2. Remove the source FEC payload ID by placing the RTP payload part directly after the RTP header.

The precise structure and length of the Source FEC payload ID depends on the FEC scheme employed. The FEC scheme and its necessary information is defined in Annex <B> and indicated by out-of-band methods as defined in clause 8.2.1.14.

8.2.1.4 RTP Payload Format for Protection Data

The RTP payload format for protection data carries encoding symbols generated by the FEC encoding process as payload. The basic format for a protection RTP packet is depicted in Figure 13. The RTP header is followed by the Repair FEC Payload ID, which are the RTP payload header for one or more encoding symbols. Those encoding symbols fill the rest of the packet. The RTP payload format is identified by the media type (application/rtp-mbms-fec-symbols) defined in Clause 8.1.2.13.

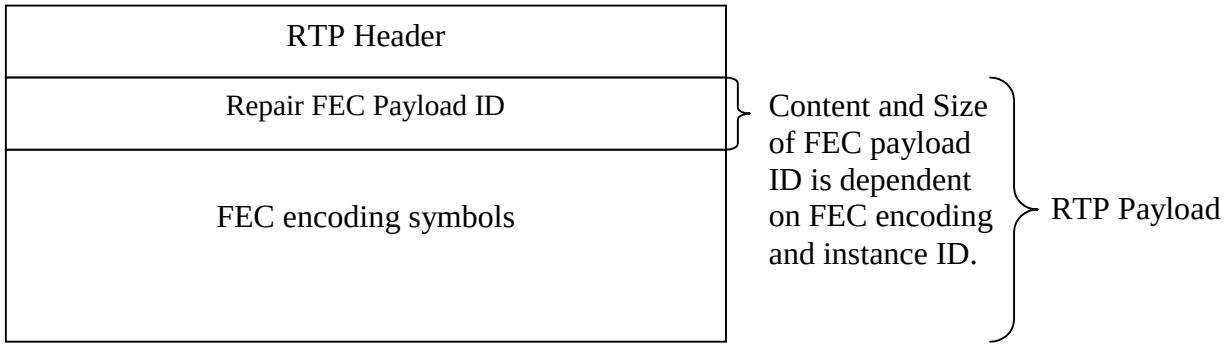


Figure 13: RTP payload structure for repair RTP packets

The RTP header information for the protection RTP packet is set as follows:

Marker bit: The marker bit shall be set 1 for the last protection RTP packet sent for each source block, and otherwise set to 0.

Timestamp: The timestamp rate shall be 10 kHz and shall be set to a time corresponding to the packet's transmission time. The timestamp value has no use in the actual FEC protection process and is only set to a value to produce reasonable resolution for arrival measuring and jitter calculation.

Sequence number: Is set in accordance with RFC 3550 [6]. The sequence number is primarily used to detect losses of the protection RTP packets. All protection RTP packets for a source block shall be sent in consecutive order, and be complete before starting the transmission of the next source block.

Payload type (PT): Is dynamically allocated using an out-of-band signalling mechanism. The PT is used to determine the FEC parameters for the payload. The FEC encoding ID, FEC instance ID, and any additional FEC object transmission information is all determined through the PT.

SSRC: One SSRC is used per source SSRC. The SSRC used by the protection payload format shall be different the one used by the source RTP packets. The binding of the source SSRC to the repair SSRC shall be performed using the RTCP SDES CNAME, which shall be identical for the two SSRCs.

The FEC Payload ID is of a fixed in size for a given PT, as the PT defines the FEC scheme employed and its parameters (if any). A FEC encoding ID and its FEC payload ID is defined in 8.2.1.

8.2.1.5 A Structure of the FEC source block

Editor's Note: Other methods may be defined by the selected FEC code.

This clause defines the method for forming the FEC source block that is utilized by the FEC encoding ID specified in the Clause 8.2.1.6.

How to reconstruct the original RTP packets that are to be included in a source block from source RTP packets are described in Clause 8.1.2.3. Let L be the length in bytes of an original RTP packet that is to be added to the source block. The source symbol size is T bytes and shall be whole bytes. When the original RTP packet is placed into the source block the value of L is first written as a two-byte value, followed by the packet. If the 2 byte RTP packet length indicator and source RTP packets with size L do not align with a encoding symbol boundary, then padding up to the next boundary shall be done using the value 0 in each byte. As long as any source RTP packets remain to be placed, the procedure is repeated starting writing the RTP packet size in the next encoding symbol.

An example of forming a source block follows, where the symbol size T is 16 bytes. Initially the source block contains K = 0 source symbols. If the first source RTP packet is 26 bytes in length then 26 is written into the first two bytes of source symbol K = 0 of the source block followed by the 26 bytes of the RTP packet. The packet then spans U = ceil((2+26)/16) = 2 source symbols. The last 4 bytes of source symbol K+Uñ1 = 1 are filled with zeroes to the next source symbol boundary, and the number of source symbols K in the source block is updated to K+U = 2. If the second original RTP packet is 52 bytes in length then 52 is written into the first two bytes of source symbol K = 2 of the source block followed by the 52 bytes of the RTP packet. The packet then spans U = ceil((2+52)/16) = 4 source symbols. The last 10 bytes of source symbol K+Uñ1 = 5 are filled with zeroes to the next source symbol boundary, and K is updated to K+U = 6. If the third original RTP packet is 103 bytes in length then 103 is written into the first two bytes of source symbol K = 6 of the source block followed by the 103 bytes of the RTP packet. The packet then spans U = ceil((2+103)/16) = 7 source symbols. The last 7 bytes source symbol K+Uñ1 = 12 are filled with zeroes to the next source symbol boundary, and K is updated to K+U = 13. If the third original RTP packet is the last of the source block then the number of source symbols in the source block is K = 13. The source block for this example is illustrated in Figure 14, where each entry is a byte and the rows correspond to the source symbols and are numbered from 0 to 12.

26		$B_{0,0}$	$B_{0,1}$	$B_{0,2}$	$B_{0,3}$	$B_{0,4}$	$B_{0,5}$	$B_{0,6}$	$B_{0,7}$	$B_{0,8}$	$B_{0,9}$	$B_{0,10}$	$B_{0,11}$	$B_{0,12}$	$B_{0,13}$
$B_{0,14}$	$B_{0,15}$	$B_{0,16}$	$B_{0,17}$	$B_{0,18}$	$B_{0,19}$	$B_{0,20}$	$B_{0,21}$	$B_{0,22}$	$B_{0,23}$	$B_{0,24}$	$B_{0,25}$	0	0	0	0
52		$B_{1,0}$	$B_{1,1}$	$B_{1,2}$	$B_{1,3}$	$B_{1,4}$	$B_{1,5}$	$B_{1,6}$	$B_{1,7}$	$B_{1,8}$	$B_{1,9}$	$B_{1,10}$	$B_{1,11}$	$B_{1,12}$	$B_{1,13}$
$B_{1,14}$	$B_{1,15}$	$B_{1,16}$	$B_{1,17}$	$B_{1,18}$	$B_{1,19}$	$B_{1,20}$	$B_{1,21}$	$B_{1,22}$	$B_{1,23}$	$B_{1,24}$	$B_{1,25}$	$B_{1,26}$	$B_{1,27}$	$B_{1,28}$	$B_{1,29}$

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

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

8.2.1.6 FEC Encoding ID definition

This clause defines a FEC encoding scheme and is identified by the FEC encoding ID [TBA]. It utilized the method for forming FEC source block as defined in Clause 8.2.1.3. It defines two different FEC payload IDs, one for source RTP packets and another for FEC encoding symbols that are used with the corresponding RTP payload formats.

Editorís Note: This clause requires some rewording after the final decision on the FEC scheme(s) to be supported.

8.2.1.7 FEC Payload ID for Source symbols

The Source FEC payload ID is composed as follows:

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

Figure 1: Source FEC Payload ID

Source Block Number (SBN), (8 or 16 bits): The I-D of the source block the media packet belongs to.

Encoding Symbol ID (ESI), (8 or 16 bits): The starting symbol index of the source packet in the source block.

Editorís Note: The Source Block number could be 1 or 2 bytes and is independent of the FEC scheme ñ its length is a tradeoff decision between wrap-round resilience and overhead. It would also be possible to allow both lengths and distinguish by SDP. This needs to be decided.

Editorís Note: For your information, as far as we understand the various FEC proposals, the length of SBN and ESI is as follows:

	DF	Siemens	NEC	Bamboo	Nokia
SBN	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2
ESI	2	2	2	2	1

8.2.1.8 FEC payload ID for Repair symbols

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

Source Block Number SBN

Encoding Symbol ID ESI
Source block length SBL
Encoding block length EBL
Symbol Length T

Figure 2: Repair FEC Payload ID

Source Block Number (SBN), (8 or 16 bits): The I-D of the source block the media packet belongs to.

Encoding Symbol ID (ESI), (8 or 16 bits): The starting symbol index of the source packet in the source block.

Source Block Length (SBL), (8, 16, or 32 bits): The number of source symbols in the source block.

Encoding Block Length (EBL), (not present, 8, 16, or 32 bits): The total number of symbols (source symbols of the source block plus encoding symbols generated from the source block) that are sent for the source block.

Symbol Length (T), (not present, 8 or 16 bits): The length of a symbol in bytes.

Editorís note: The rest of this clause requires rewording after the FEC schemes supported have been selected.

Editorís note: For your information, as far as we understand the various FEC proposals, the length of the various FEC payload ID fields is as follows:

	DF	Siemens	NEC	Bamboo	Nokia
SBN	1 or 2	1 or 2	1 or 2	1 or 2	1 or 2
ESI	2	2	2	2	1
SBL	2	4	2	2	1
EBL	0	4 (*)	2 (*)	2 (*)	1
T	2 (*)	2 (*)	2 (*)	2 (*)	1

Editorís note: the (*) in the above table denotes fields that could probably be of zero length here, and conveyed in SDP.

8.2.1.9 FEC encoding procedures

Editor's note: Here the exact procedures of the FEC scheme is to be defined. Information that needs IANA registration also needs to be indicated.

8.2.1.10 Signalling

The two different RTP payload formats requires each a media type to identify them and define their respective set of parameters.

8.2.1.11 Registration of media type application/rtp-mbms-fec-symbols

This media type represents the RTP payload format defined in Clause 8.2.1.4.

Type name: application

Subtype name: rtp-mbms-fec-symbols

Required parameters:

FEID: The FEC Encoding ID used in this instantiation. Expressed either as an integer or a string without white space.

min-buffer-time: The minimum RTP receiver buffer time need to ensure that FEC repair has time to happen. The value is in milliseconds. The min-buffer-time shall be the maximum time allowed for the sender between sending the first and the last packet, source or repair, belonging to the same source block.

Optional parameters:

FIID: The FEC Instance ID used in this instantiation. Expressed either as an integer or a string without white space. Parameter shall be present for all FEC encoding IDs that are not fully specified.

FOTI: The additional FEC object transmission information if any that the FEC instantiation uses as defined by the FEID and FIID. The content is expected to be binary data but is not necessarily so, however it shall always be BASE64 encoded in the parameter value.

Encoding considerations:

The binary parameter FOTI shall be encoded using BASE 64. The RTP payload format is a binary one, however as it is restricted to usage over RTP, no special considerations are needed.

Restrictions on usage:

This format is only defined for transfer over RTP RFC3550.

Security considerations:

This format carries protection data and instructions used in FEC decoding. Thus alteration or insertion of packets can cause wide spread corruption of recovered data. Thus authentication and integrity protection of the format is appropriate. See also clause 7 of RFC 3452.

Interoperability considerations:

The greatest interoperability issue with this format is that it virtually supports any systematic FEC encoding scheme. Thus the issue is to ensure that both sender and receiver are capable of using the same FEC codes.

Published specification:

3GPP Technical specification TS 26.346

Applications which use this media type:

MBMS terminals capable of receiving the MBMS streaming delivery method.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh File Type Code(s): N/A

Person & email address to contact for further information:

Magnus Westerlund (magnus.westerlund@ericsson.com)

Intended usage:

COMMON

Author:

3GPP SA4

Change controller:

3GPP TSG SA

8.2.1.13 Registration of media type audio, video, or text/rtp-mbms-fec-tag

This media type represents the RTP payload format defined in Clause 8.1.2.4.

Type name: audio, video, or text

Subtype name: rtp-mbms-fec-tag

Required parameters:

opt: The original payload type (OPT) of the media that is tagged by this instantiation of the format. This is an unsigned integer which range depends on the RTP profile in use, but commonly 0-127.

rate: An integer value, equal to the RTP timestamp rate value for the payload type specified by "opt".

FEID: The FEC Encoding ID used in this instantiation. Expressed either as an integer or a string without white space.

Optional parameters:

FIID: The FEC Instance ID used in this instantiation. Expressed either as an integer or a string without white space. Parameter shall be present for all FEC encoding IDs that are not fully specified.

FOTI: The additional FEC object transmission information if any that the FEC instantiation uses as defined by the FEID and FIID. The content is expected to be binary data but is not necessarily so, however it shall always be BASE64 encoded in the parameter value.

Encoding considerations:

This format is a binary one, however as it is restricted to usage over RTP, no special considerations are needed.

Restrictions on usage:

This type is only defined for transfer over RFC3550.

Security considerations:

This format carries source data and instructions used in FEC decoding. Thus alteration or insertion of packets can cause wide spread corruption of recovered data. Thus authentication and integrity protection of the format is appropriate. See also clause 7 of RFC 3452.

Interoperability considerations:

The greatest interoperability issue with this format is that it supports any FEC encoding that follows the rules of RFC 3452. Thus the issue is to ensure that both sender and receiver are capable of using the same FEC codes.

Published specification:

3GPP Technical specification TS 26.346

Applications which use this media type:

MBMS terminals capable of receiving streaming media.

Additional information:

Magic number(s): N/A

File extension(s): N/A

Macintosh File Type Code(s): N/A

Person & email address to contact for further information:

Magnus Westerlund (magnus.westerlund@ericsson.com)

Intended usage:

COMMON

Author:

3GPP SA4

Change controller:

3GPP TSG SA

8.2.1.14 Mapping the Media types to SDP

The information carried in the MIME media type specification has a specific mapping to fields in the Session Description Protocol (SDP) [6], which is commonly used to describe RTP sessions. When SDP is used to specify sessions employing the FEC payload format, the mapping is as follows:

- The Media type (application, audio, video, or text) goes in SDP "m=" as the media name.
- The Media subtype (payload format name) goes in SDP "a=rtpmap" as the encoding name. The RTP clock rate in "a=rtpmap" SHALL be 10000 for application/rtp-mbms-fec-symbols, and according to the rate parameter for audio/rtp-mbms-fec-tag, video/rtp-mbms-fec-tag, or text/rtp-mbms-fec-tag.
- Any remaining parameters go in the SDP "a=fmtp" attribute by copying them directly from the MIME media type string as a semicolon separated list of parameter=value pairs.

These payload formats is only intended to be used in declarative SDP use cases and no offer/answer negotiation procedures is defined.

8.2.1.15 Example of SDP for FEC

An example of how an SDP could look for a session containing two media streams that are FEC protected. In this example we have assumed an audiovisual stream, using 56 kbps for video and 12 kbps for audio. We further assume that we send redundant packets for the video part at 6 kbps and redundant packets for the audio part at 3 kbps. Hence, the total media session bandwidth is 56+6+12+3 = 77 kbps. In addition another 1200 bits/second of RTCP packets from the source is used for the both sessions.

The FEC encoding symbols payloads does also declare that the minimal required buffering time for either of the streams are 2.6 seconds. As the value is defined as a limitation on the sender side, further buffering to handle jitter in the transmission is also likely to be required.

```
v=0
o=ghost 2890844526 2890842807 IN IP4 192.168.10.10
s=3GPP MBMS Streaming SDP Example
i=Example of MBMS streaming SDP file
u=http://www.infoserver.example.com/ae600
e=ghost@mailserver.example.com
c=IN IP4 224.1.2.3
t=3034423619 3042462419
b=AS:77
m=video 4002 RTP/AVP 97 96 100
b=AS:62
b=RR:0
b=RS:600
a=rtpmap:96 H263-2000/90000
a=fmtp:96 profile=3;level=10
a=framesize:96 176-144
a=rtpmap: 97 rtp-mbms-fec-tag/90000
```

a=fmtp:97 opt=96; FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="
a=rtpmap: 100 rtp-mbms-fec-symbols/10000
a=fmtp:100 FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="; min-buffer-time=2600
m=audio 4004 RTP/AVP **99** 98 **101**
b=AS:15
b=RR:0
b=RS:600
a=rtpmap:98 AMR/8000
a=fmtp:98 octet-align=1
a=rtpmap: 99 rtp-mbms-fec-tag/8000
a=fmtp: 99 opt=98;FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="
a=rtpmap: 101 rtp-mbms-fec-symbols/10000
a=fmtp:101 FEID=129;FIID=12435;FOTI="1SCxWEMNe397m24SwgyRhg=="; min-buffer-time=2600

8.3 Session description

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

8.3.1 SDP Parameters for MBMS streaming session

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

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

NOTE: The use of the above parameters is according to [FLUTE/SDP].

- Extended FEC-OTI (defined below)

NOTE: The use of rtpmap is according to SDP [14].

A FEC-OTI-extension attribute is defined which results in, e.g.:

a=FEC-OTI-extension:0 A0B1C2D3E4F5

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

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

sdp-fec-oti-extension-line = "a=FEC-OTI-extension:" fec-ref SP oti-extension CRLF

fec-ref = 1*DIGIT (the SDP-internal identifier for the associated FEC-declaration).

oti-extension = base64

base64 = *base64-unit [base64-pad]

base64-unit = 4base64-char

base64-pad = 2base64-char "==" / 3base64-char "="

base64-char = ALPHA / DIGIT / "+" / "/"

9 Associated delivery procedures

9.1 Introduction

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

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

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

This specification describes these associated delivery procedures:

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

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

An instance of an ì associated procedure descriptionî 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.

9.2 Associated Procedure Description

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

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

The most recently delivered configuration file shall take priority, such that configuration parameters received prior to, and out-of-band of, the download session they apply to are regarded as ì global defaultsî, 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 îapplication/mbms-associated-procedure-parameterî identifies associated delivery procedure description instances (configuration files).

In XML, each associated delivery procedure entry shall be configured using a ìprocedureî element. All configuration parameters of one associated delivery procedure are contained as attributes of a ìprocedureî element. The ìlabelî attribute of a ìprocedureî element identifies the associated procedure to configure. The associated delivery procedure description is specified formally as an XML schema below.

9.3 File Repair Procedure

Editor's note : it was agreed in SA4 PSM SWG#6 that ptm repair will be specified. Tdoc S4-AHP156 gives an agreed basis for the specification text."

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 clientsí 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.

9.3.1 Identification of Missing Data from an MBMS Download

[Editorís 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.]

9.3.2 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.

9.3.2.1 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 ìoffset-timeî attribute.

9.3.2.2 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 ìrandom-time-periodî attribute.

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

9.3.2.3 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.

9.3.3 File Repair Server Selection

9.3.3.1 List of Server URIs

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

~~6.2.2.1.14 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.

6.2.2.1.15 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 ìnumber1.aacî 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)~~

~~6.2.2.1.16 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.~~

~~6.2.2.1.17 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.~~

~~6.2.2.1.18 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 ìapplication/simpleSymbolContainerî, 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.

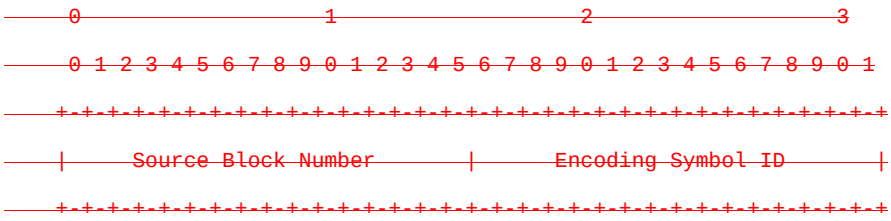


Figure 3 Example FEC Payload ID Format

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

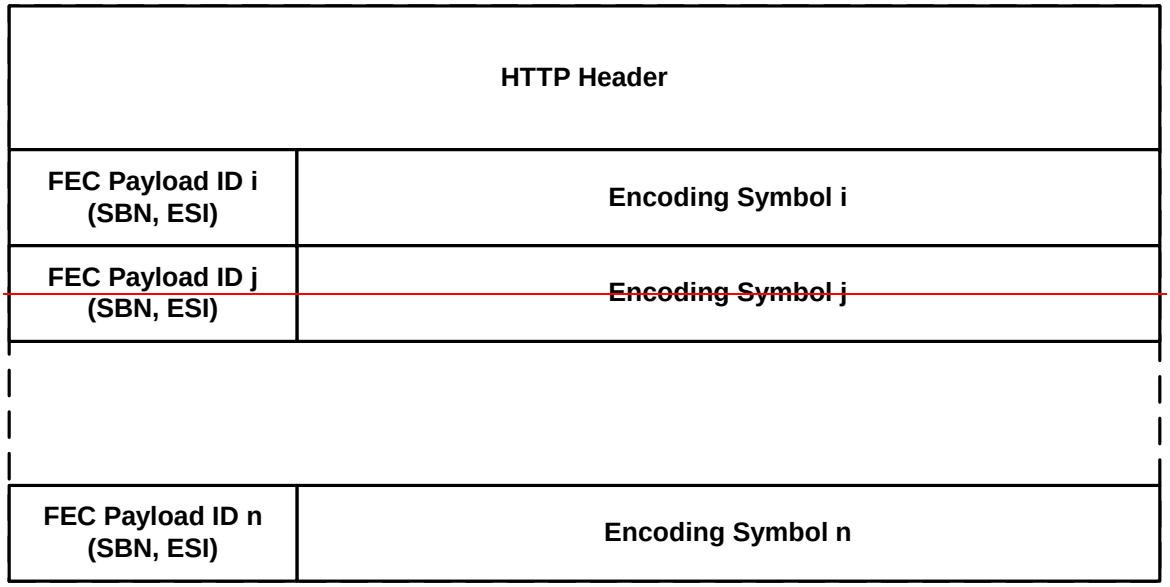


Figure 4 File Repair Response Message Format

6.2.2.1.19 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 sessionís 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 mayprovide 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 ìServerî attribute (type ìxs:anyURIî) shall only take on base URI values and not give the URI resource part.

ìLabelî value = {ìPostFileRepairî}

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 ì Sender Current Time (SCT)î field (when T=1) in the ALC/LCT header.
- R flag: The R flag is used to indicate the use of the optional ì 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.

~~AnnexC (informative):~~
~~Change history~~

~~Change history~~							
~~Date~~	~~TSG #~~	~~TSG Doc.~~	~~CR~~	~~Rev~~	~~Subject/Comment~~	~~Old~~	~~New~~
~~09-2004~~	~~25~~	~~SP-040632~~			~~Version 1.0.0~~		~~1.0.0~~

associated delivery procedure description shall be of the same type, e.g. all IP addresses of the same version, or all domain names. The number of URIs is determined by the number of ì serverURIî elements, each of which shall be a child-element of the ì procedureî element. The ì serverURIî element provides the references to the file repair server via the ì xs:anyURIî value. At least one ì serverURIî element shall be present.

9.3.3.2 Selection from the Server URI List

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

9.3.4 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 [18]. The repair request is routed to the repair server IP address resolved from the selected ì serverURIî.

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 clause 9.3.2. If there is more than one repair request to be made these are sent immediately after the first.

9.3.4.1 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) and is found from the download delivery method (the FLUTE FDT Instances describe file URIs). 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 [18] 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 [18]:

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

Where, for MBMS File Repair Request:

```
query = application "&" [ sbn_info ]

application = "mbms-rel6-flute-repair"

sbn_info = "SBN=" sbn_range *( "+" sbn_range )

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

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

esi_range = esiA [ "-" esiZ ]

sbnA = 1*DIGIT     ; the SBN, or the first of a range of SBNs

sbnZ = 1*DIGIT     ; the last SBN of a range of SBNs

esiA = 1*DIGIT     ; the ESI, or the first of a range of SBNs

esiZ = 1*DIGIT     ; 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 ìnumber1.aacî 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)

[Editorís note: future applications may define a new syntax for the query]

9.3.5 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.

9.3.5.1 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.

9.3.5.2 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 ìapplication/simpleSymbolContainerî, which denotes that the message body is a simple container of encoding symbols as described below.

[Editorís note: this Content-Type shall be IANA registered and subsequently also referenced ñ can be a separate clause dedicated to all IANA registrations]

This header is as follows:

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

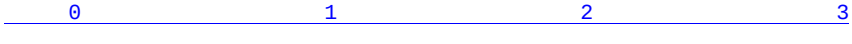
Note, other HTTP headers [18] may also be used but are not mandated by this mechanism.

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



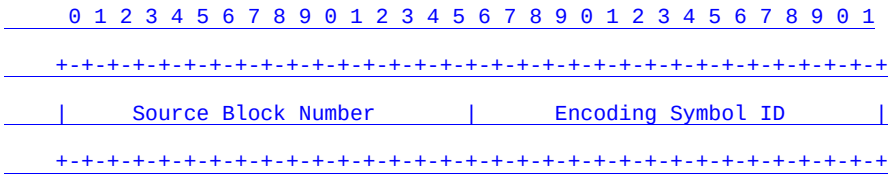


Figure 17: Example FEC Payload ID Format

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

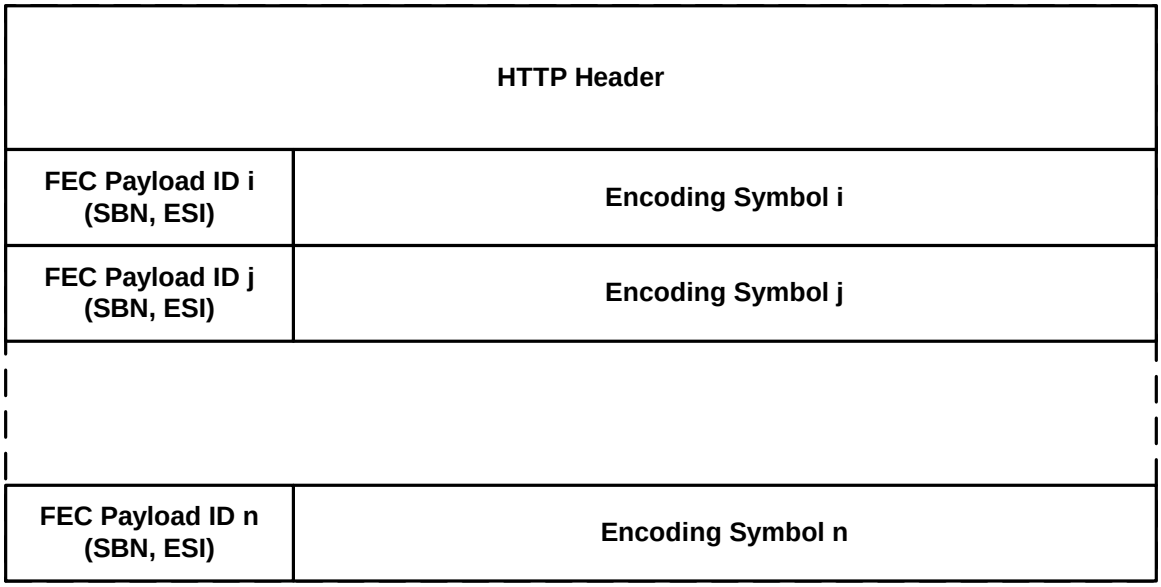


Figure 18: File Repair Response Message Format

9.3.6 Server Not Responding Error Case

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

If all of the repair servers from the serverURI list are determined to be not responding, the UE may attempt an HTTP GET to retrieve a, potentially new, instance of the sessionís 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)

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

9.4 The Reception Reporting Procedure

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

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

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

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

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

The MBMS Receiver:

4. Identifies the complete reception of a content item (e.g. a file). See clauses 9.4.1and 9.4.2.
5. Determines the need to report reception. See clause 9.4.3
6. Selects a time (Request time) at which a reception report request will be sent and selects a server from a list ñ both randomly and uniformly distributed. See clause 9.4.4 and 9.4.5.
7. Sends a *reception report request* message to the selected server at the selected time. See clause 9.4.6.

Then the server

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

9.4.1 Identifying Complete File Reception from MBMS Download

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

9.4.2 Identifying Complete MBMS Delivery Session Reception

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

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

For MBMS download sessions, FLUTE provides a "Close session flag" (see clause 7.2.5) which, when used, indicates to the UE that the session is complete.

9.4.3 Determining Whether a Reception Report Is Required

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

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

The Reception Reporting Procedure is initiated if:

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

One of the following will determine the UE behaviour:

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

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

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

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

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

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

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

9.4.4 Request Time Selection

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

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

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

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

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

In some circumstances, the system bottleneck may be in the server handling of reception reporting. In this case the *forceTimeIndependence* attribute may be used and set to true. (false is the default case and would be a redundant use of this optional attribute). When *forceTimeIndependence* is true the UE shall not use file repair point-to-point connections

to send reception reporting messages. Instead it will allow the times to expire and initiate point-to-point connections dedicated to reception report messaging.

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

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

9.4.5 Reception Report Server Selection

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

9.4.6 Reception Report Message

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

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

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

[Editorís Note: The reception report described here provides a mechanism to identify sessions in StaR and StaR-all, and files also in RAck ñ but not sessions in RAcks)]

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

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

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

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

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

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

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

- Each *fileURI* element has an optional *receptionSuccess* status code attribute which defaults to ì trueî (ì 1î) when not used. This attribute shall be used for StaR-all reports. This attribute shall not be used for StaR reports.
- [Provision of QoE Metrics using the qoeMetrics element and qoeMetricsType is pending further contribution]
- The *sessionID* attribute identified the delivery session. This is of the format source_IP_address + ì :î + FLUTE_TSI/RTP_source_port
- The *sessionType* attribute defines the basic delivery method session type used = ì downloadî || ì streamingî || ì mixedî
- The *serviceId* attribute is value and format is taken from the respective userServiceDescription serviceID definition
- The *clientId* attribute is unique identifier for the receiver. [format is FFS]

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

9.4.7 Reception Report Response Message

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

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

[Editorís note: A Reception Report response message might be used to deliver keys or as a trigger to key delivery (e.g. MSK or MTK). The use of the Reception Reporting response for this purpose is FFS.]

9.5 XML-Schema for Associated Delivery Procedures

9.5.1 Generic Associated Delivery Procedure Description

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

```
<?xml version="1.0" encoding="UTF-8"?> <xs:schema
xmlns:xs=http://www.w3.org/2001/XMLSchema elementFormDefault="qualified">

  <xs:element name="associatedProcedureDescription" type="associatedProcedureType"/>

  <xs:complexType name="associatedProcedureType">
    <xs:sequence>
      <xs:element name="postFileRepair" type="basicProcedureType" minOccurs="0" maxOccurs="1">
      <xs:element name="postReceptionReport" type="reportProcedureType" minOccurs="0" maxOccurs="1">
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="basicProcedureType">
    <xs:sequence>
      <xs:element name="serverURI" type="xs:anyURI" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="waitTime" type="xs:unsignedLong" use="required"/>
    <xs:attribute name="maxBackOff" type="xs:unsignedLong" use="required"/>
  </xs:complexType>

  <xs:complexType name="reportProcedureType">
    <xs:simpleContent>
      <xs:extension base="basicProcedureType">
            <xs:attribute name="samplePercentage" type="xs:string" use="optional"/>
            <xs:attribute name="forceTimingIndependence" type="xs:boolean" use="optional"/>
            <xs:attribute name="reportType" type="xs:string" use="optional"/>
```

```
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

</xs:schema>
```

9.5.1.1 Use of specific value

The ìServerî attribute (type ìxs:anyURIî) shall be used to provide a fully qualified domain name (FDQN) which may be resolved to an IP address - e.g. using Domain Name Service (DNS).

9.5.1.2 Example Associated Delivery Procedure Description Instance

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

```
<?xml version="1.0" encoding="UTF-8"?>
<associatedProcedureDescription
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.com/mbms-associated-descrition.xsd">
  <postFileRepair
    offsetTime="5"
    randomTimePeriod="10">
    <baseURI server="http://mbmsrepair.operator.umts/"/>
    <baseURI server="http://mbmsrepair1.operator.umts/"/>
    <baseURI server="http://mbmsrepair2.operator.umts/"/>
  </postFileRepair>
  <postReceptionReport
    offsetTime="5"
    randomtimePeriod="10"
    reportType="StR-all"
    samplePercentage="100"
    forceTimingIndependence="0">
    <baseURI server="http://mbmsrepair.operator.umts/"/>
  </postReceptionReport>
</associatedProcedureDescription>
```

9.5.2 XML Syntax for a Reception Report Request

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

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="receptionReport">
```

```
        <xs:choice>
        <xs:element name="receptionAcknowledgement" type="rackType"/>
        <xs:element name="statisticalReport" type="starType"/>
        </xs:choice>
    </xs:element>
    <xs:complexType name="rackType">
        <xs:sequence>
            <xs:element name="fileURI" type="xs:anyURI"
                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="starType">
        <xs:simpleContent>
            <xs:element name="fileURI" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded">
                <xs:attribute name="receptionSuccess" type="xs:boolean" use="optional"/>
            </xs:element>
            <xs:element name="qoeMetrics" type="qoeMetricsType" minOccurs="0"/>
            <xs:attribute name="sessionId" type="xs:string" use="optional"/>
            <xs:attribute name="sessionType" type="xs:string" use="optional"/>
            <xs:attribute name="serviceId" type="xs:string" use="optional"/>
            <xs:attribute name="clientId" type="xs:string" use="optional"/>
            <xs:attribute name="serverURI" type="xs:anyURI" use="optional"/>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

[Editorís note: xs:complexType name="qoeMetricsType" is pending further contribution]

9.5.2.1 Use of Specific Values

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

9.5.2.2 Example XML for the Reception Report Request

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

<fileURI>"http://www.example.com/mbms-files/file2.3gp"</fileURI>

<fileURI>"http://www.example.com/mbms-files/file4.3gp"</fileURI>

</receptionReport>

10 Media codecs and formats

[Editorís note: This clause writes individual codec specifications.]

10.1 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.

10.2 Speech

10.3 Audio

If audio is supported, then the following two audio decoders [should/shall] be supported:

- Enhanced aacPlus [28] [29] [30]
- Extended AMR-WB [24] [25] [26]

Editor's note: SA4#33 has agreed that both audio decoders will be included here either as recommended ("should") or as mandatory ("shall").

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

Content type Bit rate	Music	Speech over Music	Speech between Music	Speech
14 kbps mono				
18 kbps stereo				
24 kbps stereo				
24 kbps mono				
32 kbps stereo				
48 kbps stereo				

10.4 Video

10.5 Still images

10.6 Text

10.7 3GPP file format

Editorís note: A container file format, e.g. as specified in S4-040699, is currently considered for MBMS Release 6. At SA4#33 in Helsinki, container formats for multimedia presentations were discussed and all participants of SA4 were encouraged to bring proposals on container file formats to SA4#34 in Lisbon for consideration in MBMS Release 6.

Annex <A> (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 ì Sender Current Time (SCT)î field (when T=1) in the ALC/LCT header.
- R flag: The R flag is used to indicate the use of the optional ì 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 <B> (normative): FEC encoder and decoder specification

Annex <C> (informative): IANA registration

This clause provides the required IANA registration

Annex <X> (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
Sep 2003	SA4#28				Initial draft for SA4 #28		0.0.0
Mar 2004	SA4 PSM SWG#05				Second draft for SA4 PSM SWG#05		0.0.1
Apr 2004	SA4 PSM SWG#05				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	SA4#31				Draft for SA4 #31	0.0.2	0.0.3
May 2004	SA4#31				Changes agreed during the PSM SWG of SA4#31	0.0.3	0.0.4
August 2004	SA4 #32				Changes agreed during the PSM SWG of SA4#32: Tdoc S4-040395 with modifications Tdoc S4-040411 clause 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
August 2004	SA4 #32	S4-040521			Agreed in SA4#32 and raised as v1.0.0. Presented for information at SA#25.	0.0.5	1.0.0
October 2004	SA4 PSM SWG#6	S4-AHP137			Correction: document S4-040413 was not agreed and therefore clause 6.3.2.2 was emptied. However, the proposal to mandate ptp repair was agreed (one sentence in clause 6.3.1)	1.0.0	1.0.1
October 2004	SA4 PSM SWG#6	S4-AHP176			1-S4-AHP160 on MBMS User service architecture Agreed with several amendments: - don't specify who passes the TMGI within this specification - Figure 2 of Tdoc: add explicitly that internal BM-SC interfaces are not within the scope. Existing clause on UE client architecture was merged with the new MBMS UE clause 2- added a note in clause 6.3.2.1 on ptm repair to be specified (S4-AHP156). 3- added a note in clause 6.3.1 on content reception verification reporting (S4-AHP174) 4- added a clause and a note reflecting the agreement on RTP mechanisms for FEC (S4-AHP181 and associated documents).	1.0.1	1.0.2
November 2004	SA4 #33	S4-040676			Minor editorial	1.0.2	1.0.3
November 2004	SA4 #33	S4-040773			Addition of text proposals agreed during the PSM SWG session. 1. S4-040698: "Specification text for the MBMS Streaming Delivery method FEC framework". Agreed to put in the TS with some modifications: - temporary tables in yellow. - add a note "This scheme is intended to be generic. If the chosen FEC scheme(s) doesn't fit, it can be modified." 2. Editorial changes/corrections, checking of numbering of titles, figures and references. 3. Added definitions 4. S4-AHP172: "Delivery Method Definition". Agreed. 5. S4-040793 " Metadata for MBMS User Services ñ Specification Text " agreed. 6. S4-040794 agreed (note that xml schema in 6.3.3 was not taken into account since overruled by 795) 7. S4-040738 Agreed with updates (?). Updated text given offline by Rod Walsh.	1.0.3	1.0.4

					8. added a note on file format based on S4-040699. 8. S4-040818 "Introduction of security functionality in BMSC" (=662v2). agreed 9. S4-040795 "Delivery Confirmation Procedure" agreed.		
November 2004	SA4 #33	S4-040835			Reorganization of chapters (without revision marks): Addition of annex B and C. Addition of text agreed during SA4#33 plenary about audio codecs. Several editorials corrections.	1.0.4	1.1.0
November 2004	SA4 #33	S4-040859			- Correction of version number to reflect the substantial changes. - Removal of one sentence in annex B. - revision marks indicated changes since v1.0.0 (S4-040521) To be presented for information to 3GPP TSG SA#26. - removal of automatic figure numbering	1.1.0	1.5.0