ETSI TR 126 981 V16.0.0 (2020-09)

LTE;
MBMS Extensions for Provisioning and Content Ingestion
(3GPP TR 26.981 version 16.0.0 Release 16)

Reference

RTR/TSGS-0426981vg00

Keywords

LTE

Intellectual Property Rights

Essential patents

Trademarks

Legal Notice

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

Foreword

This Technical Report 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

The present document has been created as part of the study item on "MBMS Extensions on Provisioning and Content Ingestion". The intention is to identify key functionality of an interface from external application service/content providers to the BM-SC for provisioning and content ingestion in order to leverage all delivery methods and procedures through the interface. Mechanisms that would be suitable for delivery of multimedia content from external application service/content providers to the BM-SC should be identified. The provisioning and content ingestion interface definition should be aligned with the Client side APIs for MBMS (TRAPI) and the study on Enhanced TV Services.

1 Scope

The present document is a study of an interface from external content providers to the BM-SC for provisioning and content ingestion.

2 References

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

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

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

[2] 3GPP TS 22.246: "Multimedia Broadcast/Multicast Service (MBMS) user services; Stage 1".

[3] 3GPP TS 26.346: "Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs ".

[4] 3GPP TR 26.917: "MBMS/PSS Enhancements to Support Television Services".

[5] 3GPP TR 23.746: "Study on System Architecture Enhancements to eMBMS for Television Video Service".

[6] 3GPP TS 23.468: "Group Communication System Enablers for LTE (GCSE_LTE); Stage 2".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the terms and definitions given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

3.2 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

ADF	Associated Delivery Function
API	Application Programming Interface
BM-SC	Broadcast Multicast Service Center
CP	Content Provider
DASH	Dynamic Adaptive Streaming over HTTP
DRM	Digital Rights Management
GBR	Guaranteed Bitrate
GUI	Graphical User Interface
MBMS	Multimedia Broadcast/Multicast Service
MPD	Media Presentation Description
MSA	MBMS Service Area

MDF	MBMS (Download & Streaming) Delivery Function
SAF	User Service Discovery / Announcement Function
SACH	Service Announcement Channel
TMGI	Temporary Mobile Group Id
TV	Television
URL	Uniform Resource Locator
VoD	Video on Demand

4 Use-Cases

4.1 Live Video from multiple cameras angles into a stadium

4.1.1 Use-Case Description

A Content Provider wants to offer multiple camera angles of a sports match into stadiums. The role of the Content Provider and the role of the network operator may be in the same organization. When the roles are separated across companies, the procedure remains the same and additional authorization and security functions may be added.

The Content Provider wants to offer for example four live feeds and a statistics or replay channel around the ongoing match into the stadium.

Only devices inside of the stadium are supposed to receive the content streams.

The Content Provider negotiates service related aspects like the broadcast areas (i.e. locations of the stadiums), the broadcast times (i.e. times of the matches) and the media quality (i.e. desire bitrates, etc.) with the operator.

The network operator (i.e. BM-SC operations) offers the Content Provider APIs or a self-service GUI for session and content provisioning. The Content Provider authenticates itself in order to use the self-service features. After authorization, the Content Provider can plan and provision sessions and content within the negotiated range.

The Content Provider plans and provisions MBMS Broadcast session by providing start and stop time, the target MBMS Broadcast area and the expected content bitrate, including tolerable losses (note, amount of AL-FEC redundancy should be hidden to the Content Provider). For each MBMS Broadcast session, the Content Provider may needs to determine whether one or more delivery sessions are provided with the same MBMS Bearer.

The Content Provider uses MBMS Broadcast for the Live Data. The Content Provider provides service information before and after the broadcast session using unicast.

Devices receive the transport control metadata before the Content Provider starts the content streams. The operator may use a Service Announcement Channel (SACH) for the purpose of distributing service announcement information. Devices may receive additional service information using unicast.

4.1.2 Recommended requirements and working assumption

Working assumptions:

- The Content Provider and the operator have a business agreement.
- The Content Provider owns the content rights to distribute the content.
- The Content Provider and the network operator role may belong to the same company.
- The Content Provider secures a dedicated App, which is used by end-users for consuming the service. The App uses the MBMS Client APIs, aligned with offering principle of the Content Provider.
- The Content Provider controls the content preparation chain.
- The App/Content Provider can utilize unicast for additional communications e.g. when the MBMS broadcast bearer is not active.

- The Service Announcement Channel (SACH) is used to distribute MBMS Service Announcement information.

Recommended Requirements:

- The interface (between BM-SC and the Content Provider) should support authentication procedures so that only authorized Content Providers can change GUI view and/ or modify the session information. The interface may support different levels of authorization, e.g. for system administrators to configure the negotiate usage ranges and system users to select a certain parameter set of a broadcast session.
- The operational complexity of sending content via broadcast should be hidden as much as possible to Content Providers.
- The Content Provider should be able to select a content ingestion method into the system, e.g. for large files and for small files. The interface should allow selecting different methods.
- The interface allows separation of different provisioning steps.
- The Content Provider provides an MPD and the needed Initialization Segments to the operator to create the service announcement information. The Content Provider may need to update the MPD during the broadcast. The interface supports providing a first MPD, providing updated MPDs and providing Initialization segments.

4.2 Nation-Wide TV channels

4.2.1 Use-Case Description

A Content Provider wants to offer nation-wide TV channels. The Content Provider negotiates with the operator the media quality and the service area (i.e. nation-wide).

The use-case is identical with use-case in Clause 4.1 with the following differences:

- The system also unicast access of the content in areas, where no MBMS broadcast is offered. The Content Provider may provide Multiple Representation (Multi Rep) content or the operator creates the MBR content from a single, high quality input.
- The Content Provider does not control the precise delivery option, i.e. unicast bearers or MBMS bearers. The operator selects the mode of transmission, e.g. based on current popularity or based on predictions from past events. The operational complexity of switching between unicast and MBMS is hidden from the Content Provider.
- - The Content Provider receives consumption statistics and / or quality of experience statistics from the operator.

The operator may have an agreement with multiple Content Providers so that a certain set of TV channels can be provided.

4.2.2 Recommended requirements and working assumption

Working assumptions are identical to the working assumptions in clause 4.1.2 with the following differences:

- The Service Announcement Information should also be available via unicast.

Recommended Requirements are identical to the working assumptions in clause 4.1.2 with the following differences:

- The operational complexity of sending content via broadcast or via unicast should be hidden as much as possible to Content Providers.
- The Content Provider provides a Multiple Representation (MultiRep) offering for unicast and a Single Bitrate for MBMS Broadcast.
- The interface allows activating and retrieval of session usage and quality of experience information.

4.3 VOD prepositioning

4.3.1 Use-Case Description

A Content Provider wants to preposition VoD clips on the device. The Content Provider negotiates target reception areas (geographical area) and the distribution duration with the operator. The Content Provider estimates the VoD clip size so that the needed transmission resources for the target duration can be determined. The Content Provider targets all devices in the MBMS Service Area. Each VoD clip could be protected using a DRM scheme.

The system enables the Content Provider to create delivery sessions and MBMS Service Areas (in range of the SLA). The system provides a delivery session id for each delivery session. A delivery session is for instance an MBMS Download Delivery Session.

The Content Provider plans transmission sessions in the desired geographical area and for the desired time. The Content Provider may use a self-service portal or the operator may configure the transmission sessions on behave of the Content Provider. The usage of a file repair function is hidden to the Content Provider. The Content Provider could require reception acknowledgements.

The Content Provider could choose just before the actual transmission, which VoD clips should be distributed. The Content Provider will initially provision broadcast sessions and later on the actual content. At some short time before the start of transmission of a certain VoD clip, the Content Provider will instruct the BM-SC to prepare the clip for MBMS delivery, for instance by loading the VoD clip into the memory in order to execute file partitioning and FEC calculation. Note other realizations may support on-the-fly preparation of larger files.

When the Content Provider caches content in the BM-SC, then the Content Provider has also additional management procedures to update or delete cached files.

The Content Provider provides a dedicated App, which handles the reception of the VoD clips and offers the prepositioned clips to the user. The App may also offer additional VOD clips, which are not pre-positioned in order to increase the choice. However, prepositioned clips are marked, since consumption is independent from network connectivity or network availability.

4.3.2 Recommended requirements and working assumption

Working assumptions are identical to the working assumptions in clause 4.1.2 and 4.2.2 with the following difference:

- The Content Provider has prepared one or more VoD files, optionally DRM protected.

Recommended requirements are identical to the working assumptions in clause 4.1.2 and 4.2.2 with the following differences:

- Operational complexity is hidden as much as possible to Content Providers. In particular, the Content Provider may influence the file repair procedure (e.g. duration of the backoff window), but does not have to be aware of the details
- The Content Provider should be able to select a content ingestion method into the system, e.g. for large files versus small files. It is recommended that the interface allows the separation of the preparation and transmission and a mode where preparation and transmission are combined. The same ingestion method is used for the entire delivery session.
- The interface allows separation of Broadcast Session Provisioning (time, location and bitrates) from Broadcast Content Provisioning (which VoD files to send) functions.
- For BM-SC cached content, the interface may provide management procedures to delete or update cached objects.

4.4 Software Update

4.4.1 Use-Case Description

The use-case is similar to the VoD prepositioning use-case with the difference, that a binary software packet is distributed in this case. The Content Provider is the provider of either the App software or the operating system software, and thus is authorized to provide the software updates. The Content Provider also provides an software update management App.

The software update management App utilizes the TRAPI APIs and interacts with the MBMS client. The software update management App is also able to identify the correct software version for the device and can also verify correct reception.

The Content Provider may repeat the broadcast during the day.

The system enables the Content Provider to create delivery sessions and MBMS Service Areas (in range of the SLA). The system provides a delivery session id for each delivery session to the Content Provider so that the Content Provider can pre-cache objects (optional) or request statistics for that session A delivery session is for instance an MBMS Download Delivery Session.

For larger software binaries, the Content Provider can instruct the BM-SC to cache the object and start file partitioning and FEC redundancy calculation. The Content Provider can send later on a "start transmission" command or can provision on the BM-SC a transmission start time so that the BM-SC can start on its own.

4.4.2 Recommended requirements and working assumption

Recommended Requirements are identical to the working assumptions in the previous clauses.

4.5 TV Program Guide update delivery

4.5.1 Use-Case Description

A program guide for a TV service describes the schedule of individual programs and program repetitions. Essentially, a program guide contains for each program a title, a start time and a duration or a start and stop time, optionally a short description of the program and optionally additional information like languages (aka audio tracks), genre, parental rating, etc.

The programs of a certain number of days (like the next 5 days) are described by the program guide. The program guide may be updated / extended on a per-hour or per-day basis. Conceptually, the window of described programs is progressively moving forward in time, new programs are appended to the end and past programs are removed from the front.

Sometimes, the program schedule changes on-short notice, when a special event happens. Consequently, some programs are removed from the existing list and replaced by other program descriptions. Each program is uniquely identified by the start time (in UTC) and the TV service. There is only a single program entry allowed for each timeslot. Other identifiers may be used as well.

Details of program guide realization for TV services are studied and described in [4]. In the following one realization consideration is presented.

A number of program guide formats are publicly available. It is assumed that an existing program guide format can be re-used. With today's device platform / SDK model, the Content Provider can select the preferred program guide format on its own and implement the application accordingly.

The program guide of a linear TV service is often rather small in size. Modern 3GPP devices are equipped with (semi) persistent memory. 3GPP devices can keep a copy in local memory so that the application can work with previously received information at start-up of the application. When the application is started, the application first renders the stored program guide information and then checks for program guide updates.

The device can use unicast for initial program guide update checks. When the program guide information is wisely partitioned into several, not too small files, the check can be executed quickly and at low traffic volume exchange. The device can detect changes of the file (thus, the described time range) using an HTTP HEAD request or other HTTP methods.

When the program guide is partitioning into not-too-small-files, the program guide can still be progressively extended. When each file contains all programs of a 12h time range, then the program guide can be extended in 12h steps (always all programs of the range).

Program guide updates may be provided as a separate MBMS User Service. The device starts listening for program guide updates when the reception of the audio visual content of the associated TV service has started. The Content Provider uses a continuous (Push) interface into the BM-SC in order to send or update certain EPG files.

4.5.2 Recommended requirements and working assumption

Working Assumptions:

- The Content Provider uses a continuous push interface to ingest file data into the BM-SC. The BM-SC is just sending the files without additional processing (except 3GPP define procedures such as file partitioning and FEC redundancy generation).
- The Content Provider may send files in various different orders.

5 Architecture Considerations

5.1 Introduction

The Content Provider uses a reference point towards the BM-SC for all control plane signalling (e.g., initiation and termination of MBMS user services, status notification etc.) and data plane transfer for MBMS User Services. Based on the signalling from the Content Provider, the BM-SC converts the signalling requests to corresponding procedures onto the SGmb/Sgi-mb interface to the MBMS GW. The MBMS User Service includes all needed service announcement procedures between BM-SC and UE as well as the content delivery procedures.

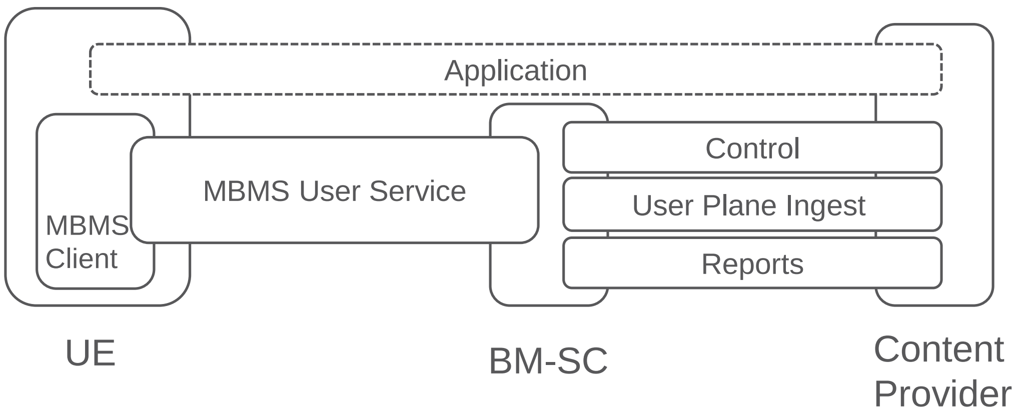


Figure 5.1-1: Scope of the MBMS User Service versus the Control and Ingest Interface

Conceptually, an MBMS User Service spans from the BM-SC to the MBMS Client in the UE. There may be additional procedures between an application in the UE and the Content Provider. The Control and Ingest Interface spans from the Content Provider to the BM-SC.

For transport-only services (cf. MBMS Service Type 1 as defined in TR 23.746 [5]) a new delivery method should be realized.

5.2 Reference Model

As shown in Figure 5.2-1, the reference point between Content Provider and BM-SC is called the "XMB" interface. Using the XMB reference point, user service provider can invoke procedures supported by BM-SC and vice versa to setup and manage user service sessions from BM-SC to the end users. BM-SC defines an endpoint with all supported procedures on the XMB interface, which can then be converted to SGmb procedures for the interface between BM-SC and MBMS GW (not depicted).

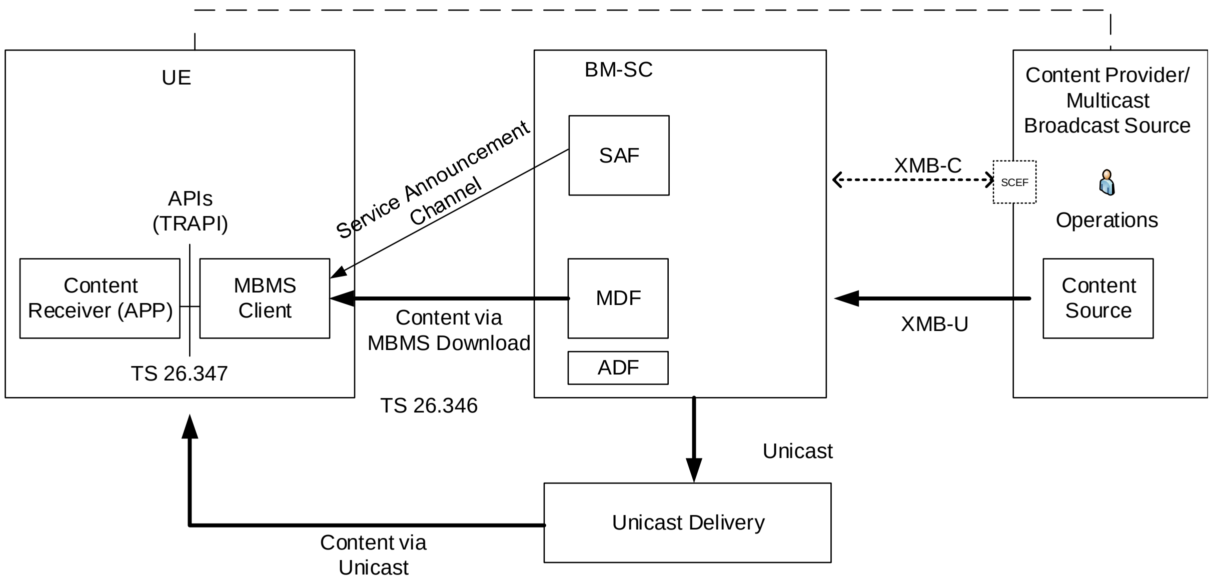


Figure 5.2-1: The XMB reference model

The BM-SC substructure relevant for the XMB interface is:

- SAF: User Service Discovery / Announcement Function
- MDF: MBMS (Download & Streaming) Delivery Function
- ADF: Associated Delivery Function

The BM-SC may forward the received content for unicast delivery for appropriate functions.

The control plane (XMB-C) may be optionally terminated by an SCEF, which exposes the same or a different interface to Content Providers. The exposed API by SCEF is not in scope of SA4.

The Content Provider may optionally exchange application level information like service metadata (e.g. serviceIds or Urls of USDs or other service identifies) directly with the application in the UE.

5.3 Provisioning and Ingestion Procedures

5.3.1 Introduction

The provisioning and ingestion procedures in this section describe realizations for the identified Use-Cases in clause 4.

Provisioning procedures are used to create and control MBMS User Services from external sources. An MBMS User Service spans from the BM-SC to the UE and can contain one or more MBMS delivery methods. The provisioning procedure offer functions to create one or more delivery sessions (such as a MBMS Download Delivery session) and allows association of the delivery sessions to MBMS Bearer Services. As part of the provisioning procedures for MBMS User Services, content ingestion for the user-plane data are negotiated. As a result of the provisioning procedure, the BM-SC starts service announcement and activates MBMS bearer services.

The Content Provider can query its entitlements, for instance the list of broadcast areas it is authorized to use.

The Content Provider my query the status of delivery sessions.

The Content Provider can request reception statistics.

5.3.2 Provisioning and Ingestion Procedure for Live DASH services (MBMS Broadcast only)

A provisioning and ingestion procedure in this section describes a realization of the "Live Video from multiple cameras angles into a stadium" use-case as introduced in clause 4.1. When the target broadcast area is not restricted to the small coverage of a stadium (e.g. broadcast coverage covers a region), the provisioning procedure describes the configuration of a MBMS Broadcast only TV services.

This procedure describes the provisioning for a Live DASH service into a single broadcast area. A Push Interface like WebDAV is used here as ingestion method for the user-plane data. The push interface is identified by a unique URI. The source of the user plane data (CP Source) are the DASH Media Segments as produced by a Live Encoder / Segmenter and the source pushes each new segment when it becomes available. The Media Presentation Description (MPD) URL for the Live DASH session is provided to BM-SC during Session creation or on a subsequent update request.

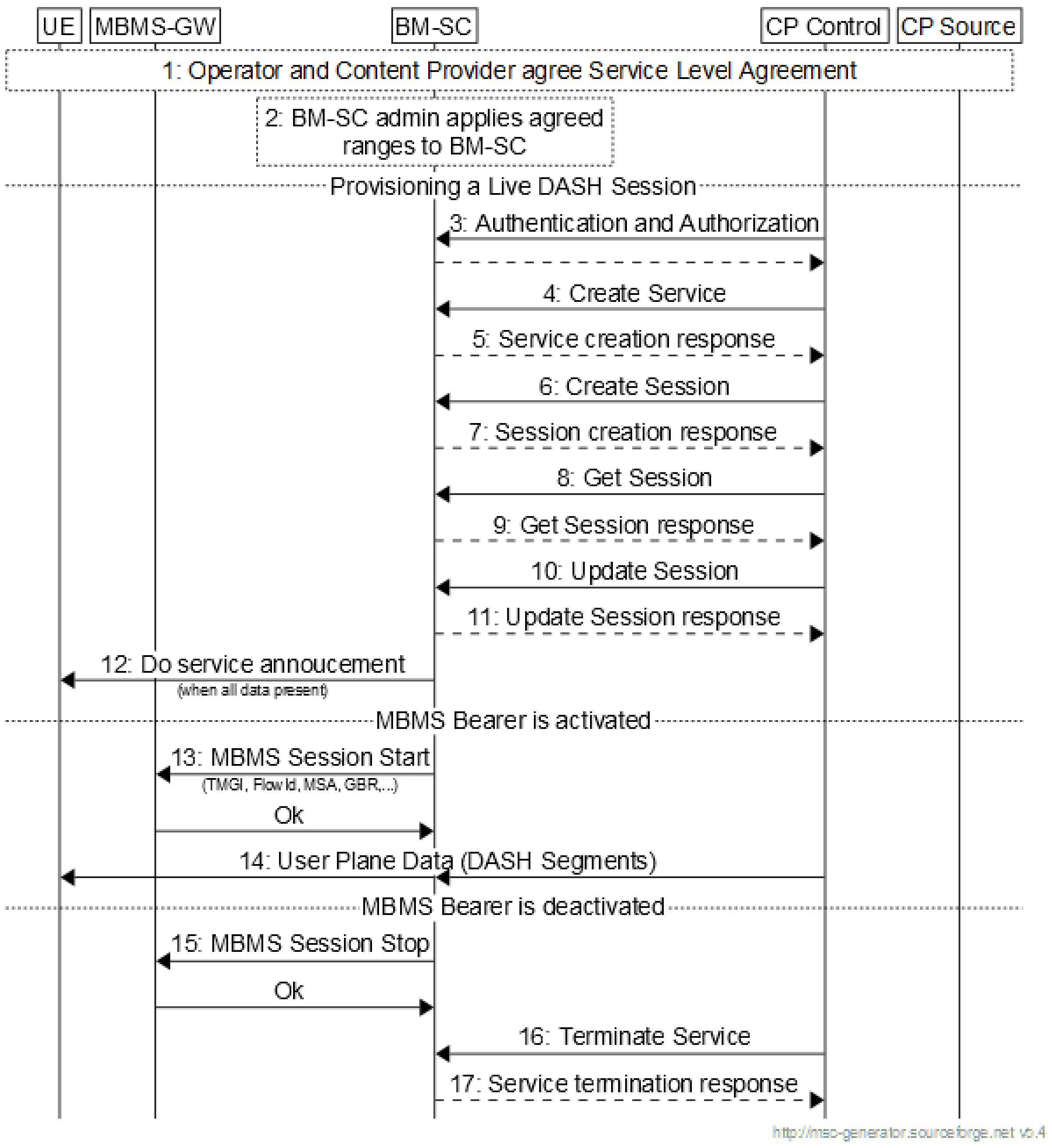

UE
MBMS-GW
BM-SC
CP Control
CP Source
1: Operator and Content Provider agree Service Level Agreement
2: BM-SC admin applies agreed ranges to BM-SC
Provisioning a Live DASH Session
3: Authentication and Authorization
4: Create Service
5: Service creation response
6: Create Session
7: Session creation response
8: Get Session
9: Get Session response
10: Update Session
11: Update Session response
12: Do service annoucement
(when all data present)
MBMS Bearer is activated
13: MBMS Session Start
(TMGI, Flow Id, MSA, GBR,...)
Ok
14: User Plane Data (DASH Segments)
MBMS Bearer is deactivated
15: MBMS Session Stop
Ok
16: Terminate Service
17: Service termination response
http://msc-generator.sourceforge.net v6.4

Figure 5.3.2-1: Live DASH Provisioning and Ingestion Procedure

1: The operator and the Content Provider agree a Service Level Agreement, which entitles the Content Provider to use the MBMS system (in accordance to some rules) for content delivery. For instance, the SLA can include day time ranges, during which the Content Provider can distributed content. The SLA can also include geographical areas, in which the Content Provider is allowed to distribute content. The SLA also includes target bitrates and likely definitions of tolerable losses per service.

2: The BM-SC administrators (operator) apply the agreed ranges. This can imply to add additional Service Areas, and other system related configurations.

The Content Provider provisioning a single Live DASH session in a single broadcast area.

3: The Content Provider authenticates itself as authorized user. The Content Provider can only see those configurations, sessions and services, which belong to the Content Provider. On successful authentication, an access token is provided to CP and will be present in every subsequent message sent by CP to BM-SC.

4: The Content Provider creates a new Service by providing service class, service languages, service names, notification configuration as well as consumption reporting configuration. The Content Provider can select whether the CP or the operator distributes service announcement by providing a list of Service Announcement Channel (SACH, as defined in Annex L.2 / L.3 of [3]) services used for operator-driven service announcement.

NOTE 1: BM-SC derives the required UE capabilities from the provided service and session properties.

5: Upon successful Service creation by the BM-SC, the BM-SC will provide a unique Service Id, that the Content Provider will use for subsequent requests.

6: The Content Provider creates a Session for previously created Service. Common session properties provided as input by Content Provider: unique Service Id this session belongs to, max ingest bitrate (excluding any FEC redundancy and transport overhead), scheduling information (start time, stop time), QoE Reporting configuration and session type (set to Application). DASH specific session properties provided as input by Content Provider: Mime-type of ASD fragment (set to application/dash+xml), Application Entry Point URL (i.e. MPD URL), ingest mode (push/pull), Unicast Delivery Indicator, Components.

NOTE 2: BM-SC allocates following parameters for SDP: TMGI, FLUTE IP Multicast Address, UDP Port and TSI.

NOTE 3: BM-SC derives the SAI list from Geographical Area provided in Content Provider request and from PLMN id negotiated in step 1. FEC information (codec and ratio) and MBMS Bearer QoS (ARP, QCI) are also negotiated in step 1.

NOTE 4: Service Announcement start time can be provided in request. If not, BM-SC starts announcing service as soon as all required service and session properties are provided by Content Provider.

NOTE 5: In the case of regional services, i.e. that deliver region specific content, a Session can be cloned so that all Sessions of user service use same FLUTE parameters.

7: A unique Session Id, which identifies the created Session, is responded. Additionally, the push URL (if required ingest mode is push) and QoE Report URL are added to the response.

8: The Content provider queries the Session configuration, providing Session Id.

9: The BM-SC provides the information in response. All readable session properties are provided in response.

10: The Content Provider updates the session by providing Application Entry Point URL

11: The BM-SC sends response with update status.

12: Once all information for service announcement is available, and if service announcement start time is elapsed, the BM-SC starts announcing the service automatically. Service announcement is automatically updated following subsequent Session updates.

The BM-SC activates automatically the MBMS Bearer at Session start time.

13: The BM-SC activates the MBMS bearer by providing the TMGI, the Flow ID, the MBMS Service Area (MSA), the GBR and other parameters to the MBMS-GW. The BM-SC can notify the Content Provider about the activation of the MBMS Bearer.

14: The BM-SC activates the MBMS Broadcast bearer according to the time schedule. When the MBMS Broadcast bearer is activated, then the BM-SC MBMS Delivery Function (MDF) starts forwarding the user plane data (push interface). Any user plane data received before activation of the MBMS bearer can be discarded.

15: At Session stop time, the MBMS bearer is terminated. The BM-SC can notify the Content Provider about the termination of the MBMS Bearer.

16: The Content Provider terminates the service. All sessions, which are still created or active will be deleted automatically by BM-SC with the termination of the service.

17: BM-SC sends service termination response.

5.3.3 Provisioning and Ingestion Procedure for Live DASH services (with Service Continuity)

A provisioning and ingestion procedure in this section describes a realization of "Nation Wide TV channels" Use-Case as introduced in clause 4.2. MBMS bearers are only used in areas of high interest to the TV service.

This procedure describes the provisioning for a Live DASH service with Service Continuity. Service continuity allows UEs to enter or leave the MBMS service areas. UEs can switch to unicast as defined in clause 7.6 of TS 26.346 when leaving the MBMS service area.

In case of service continuity support, the system offers representations via unicast and via MBMS Broadcast. A Unified MPD contains the according retrieval information. When Service Continuity is supported, the Content Provider provides MPD and Initialization segments for both unicast and MBMS broadcast access and also the according media segments. The Content Handler Functions forwards the content to a DASH (unicast) Server. The DASH (unicast) server can use a CDN for unicast delivery.

A push interface is used here as ingestion method for the user-plane data. The source of the user plane data (CP Source) are the DASH Media Segments as produced by a Live Encoder / Segmenter, which produces the content for unicast and MBMS bearer delivery. The Media Presentation Description (MPD) and Initialization Segment (IS) for the Live DASH session is provided separately to the BM-SC.

The Service Announcement Function (SAF) of the BM-SC creates the needed metadata fragments for the MBMS User Service. To support Service Continuity, the SAF adds base pattern elements to the USBD. The MBMS Client in the UE matches the base pattern against a portion of the entire request URL. The SAF creates unified MPD by adding information specific elements to it. The SAF makes the service announcement information available via unicast and via MBMS.

The Content Handler Function (CHF) handles the separation of unicast and MBMS bearer content. The CHF makes the content available in operators CDN for unicast access.

5.3.4 Provisioning and Ingestion Procedure for File Delivery Services (without File Schedule)

A provisioning and ingestion procedure in this section describes a realization of "VOD prepositioning" use-case (as introduced in clause 4.3) and "Software update" Use-Case (as introduced in clause 4.4).

This use-case described the provisioning procedure for a File Delivery service without any file schedule element into a single broadcast area. The file schedule element is carried in the schedule description fragment during service announcement and contains transmission timings information for each URL. Consequently, the file URLs has to be present when creating service announcement information.

This use-case assumes that the BM-SC automatically fetches the file using a pull method and prepares the transmission. File URLs can be provided in Session creation request or any subsequent Session update request. When file preparation ends after Session start time, the file is automatically added to user plane flow. It is up to Content Provider to secure that Session scheduling is large enough to allow files preparation and transmission according to bitrate between BM-SC and file location, and bitrate of user plane.

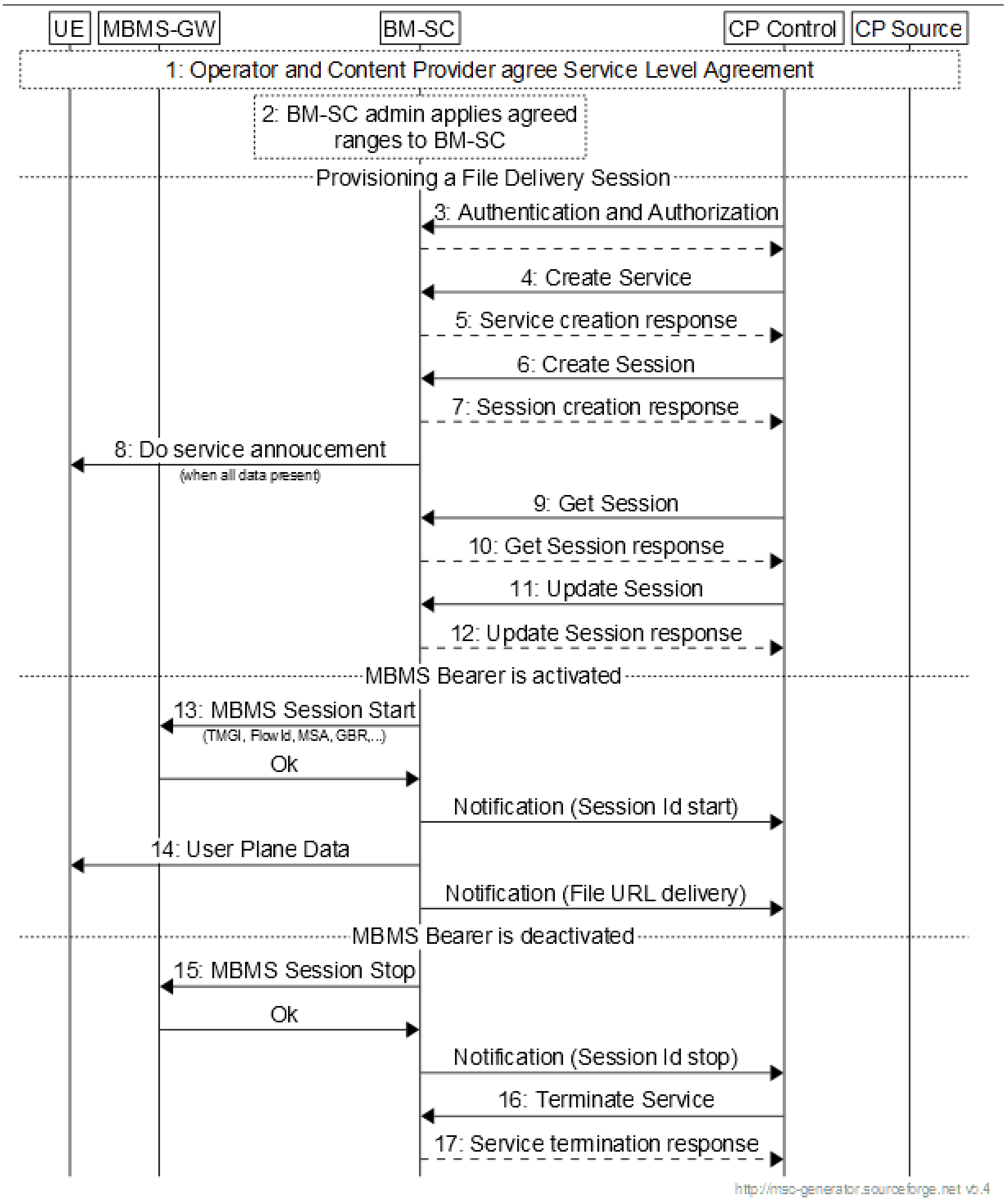

UE
MBMS-GW
BM-SC
CP Control
CP Source
1: Operator and Content Provider agree Service Level Agreement
2: BM-SC admin applies agreed ranges to BM-SC
Provisioning a File Delivery Session
3: Authentication and Authorization
4: Create Service
5: Service creation response
6: Create Session
7: Session creation response
8: Do service annoucement
(when all data present)
9: Get Session
10: Get Session response
11: Update Session
12: Update Session response
MBMS Bearer is activated
13: MBMS Session Start
(TMGI, Flow Id, MSA, GBR,...)
Ok
Notification (Session Id start)
14: User Plane Data
Notification (File URL delivery)
MBMS Bearer is deactivated
15: MBMS Session Stop
Ok
Notification (Session Id stop)
16: Terminate Service
17: Service termination response
http://msc-generator.sourceforge.net v3.4

Figure 5.3.4-1: File Delivery Provisioning and Ingestion Procedure

1: The operator and the Content Provider agree a Service Level Agreement, which entitles the Content Provider to use the MBMS system (in accordance to some rules) for content delivery. For instance, the SLA can include day time ranges, during which the Content Provider can distributed content. The SLA can also include geographical areas, in which the Content Provider is allowed to distribute content. The SLA also includes target bitrates and likely definitions of tolerable losses per service.

2: The BM-SC administrators (operator) apply the agreed ranges. This can imply to add additional Service Areas, and other system related configurations.

The Content Provider provisioning a file delivery session in a single broadcast area.

3: The Content Provider authenticates itself as authorized user. The Content Provider can only see those configurations, sessions and services, which belong to the Content Provider. On successful authentication, an access token is provided to CP and will be present in every subsequent message sent by CP to BM-SC.

4: The Content Provider creates a new Service by providing service class, service languages, service names, notification configuration as well as consumption reporting configuration. The Content Provider can select whether the CP or the operator distributes service announcement by providing a list of Service Announcement Channel (SACH, as defined in Annex L.2 / L.3 of [3]) services used for operator-driven service announcement.

Note: BM-SC derives the required UE capabilities from the provided service and session properties.

5: Upon successful Service creation by the BM-SC, the BM-SC will provide a unique Service Id, that the Content Provider will use for subsequent requests.

6: The Content Provider creates a Session for previously created Service.

Common session properties provided as input by Content Provider: unique Service Id this session belongs to, max ingest bitrate (excluding any FEC redundancy and transport overhead), scheduling information (start time, stop time), Geographical Area and QoE Reporting configuration and session type (set to Files).

Files specific session properties provided as input by Content Provider: ingest mode (pull/push), file list if pull mode.

NOTE 1: BM-SC allocates following parameters for SDP: TMGI, FLUTE IP Multicast Address, UDP Port and TSI.

NOTE 2: BM-SC derives the SAI list from Geographical Area provided in Content Provider request and from PLMN id negotiated in step 1. FEC information (codec and ratio) and MBMS Bearer QoS (ARP, QCI) are also negotiated in step 1.

NOTE 3: In pull ingest mode, file URLs can be provided now or at a later stage through the Session Update procedure.

NOTE 4: Service Announcement start time can be provided in request. If not, BM-SC starts announcing service as soon as all required service and session properties are provided by Content Provider

NOTE 5: In the case of regional services, i.e. that deliver region specific content, a Session can be cloned so that all Sessions of user service use same FLUTE parameters.

7: A unique Session Id, which identifies the created Session, is responded. If push ingest mode is used, BM-SC provides also the push URL the Content Provider will use.

NOTE: For file URLs provided in Session creation request, the BM-SC starts automatically to fetch the file resource(s) from the content location when file earliest fetch time elapses and generates the FLUTE and FEC symbols (if any). The BM-SC can notify the Content Provider when the process is finalized.

8: Once all information for service announcement is available, and if service announcement start time is elapsed, the BM-SC starts announcing the service automatically. Service announcement is automatically updated following subsequent Session updates. File schedule element can be present in Schedule fragment for files URLs provided in Session creation request.

9: The Content Provider queries the Session configuration, providing Session Id.

10: The BM-SC provides the information in response.

11: The Content Provider updates the session by providing additional File URLs.

12: The BM-SC sends response with update status.

NOTE 6: The BM-SC starts automatically to fetch the new file resource(s) from the content location when file earliest fetch time elapses and generates the FLUTE and FEC symbols (if any). The BM-SC can notify the Content Provider when the process is finalized.

NOTE 7: Steps 9-12 can be executed at any time after Session is created and prior to the Session stop time. Any file URL added after Session start time will be automatically fetched, processed and sent on user plane.

The BM-SC activates automatically the MBMS Bearer at Session start time.13: The BM-SC activates the MBMS bearer by providing the TMGI, the Flow ID, the MBMS Service Area (MSA), the GBR and other parameters to the MBMS-GW. The BM-SC can notify the Content Provider about the activation of the MBMS Bearer.

14: The BM-SC activates the MBMS Broadcast bearer according to the time schedule. When the MBMS Broadcast bearer is activated, then the BM-SC MBMS Delivery Function (MDF) starts sending the user plane data, according to Target reception completion time.

NOTE 8: The BM-SC can notify Content Provider of file delivery start/end.

15: At Session stop time, the MBMS bearer is terminated. The BM-SC can notify the Content Provider about the termination of the MBMS Bearer.

16: The Content Provider terminates the service. All sessions, which are still created or active will be deleted automatically by BM-SC with the termination of the service.

17: BM-SC sends service termination response.

5.3.5 Provisioning and Ingestion Procedure for File Carousel Services

This use-case describes the provisioning procedure for a File Carousel service. The provisioning procedure in clause 5.3.4 can also be used to realize File Carousel Services.

This use-case assumes that the BM-SC automatically fetches the file using a pull method and prepares the transmission. BM-SC sends files in a carousel back-to-back mode during Session transmission. File URLs can be provided in Session creation request or any subsequent Session update request. When file preparation ends after Session start time, the file is automatically added to user plane flow.

When Session starts, File Carousel Service checks on a regular basis the content location of each file URL. If file changed, updated version is automatically fetched, prepared and transmitted on user plane in place of old version.

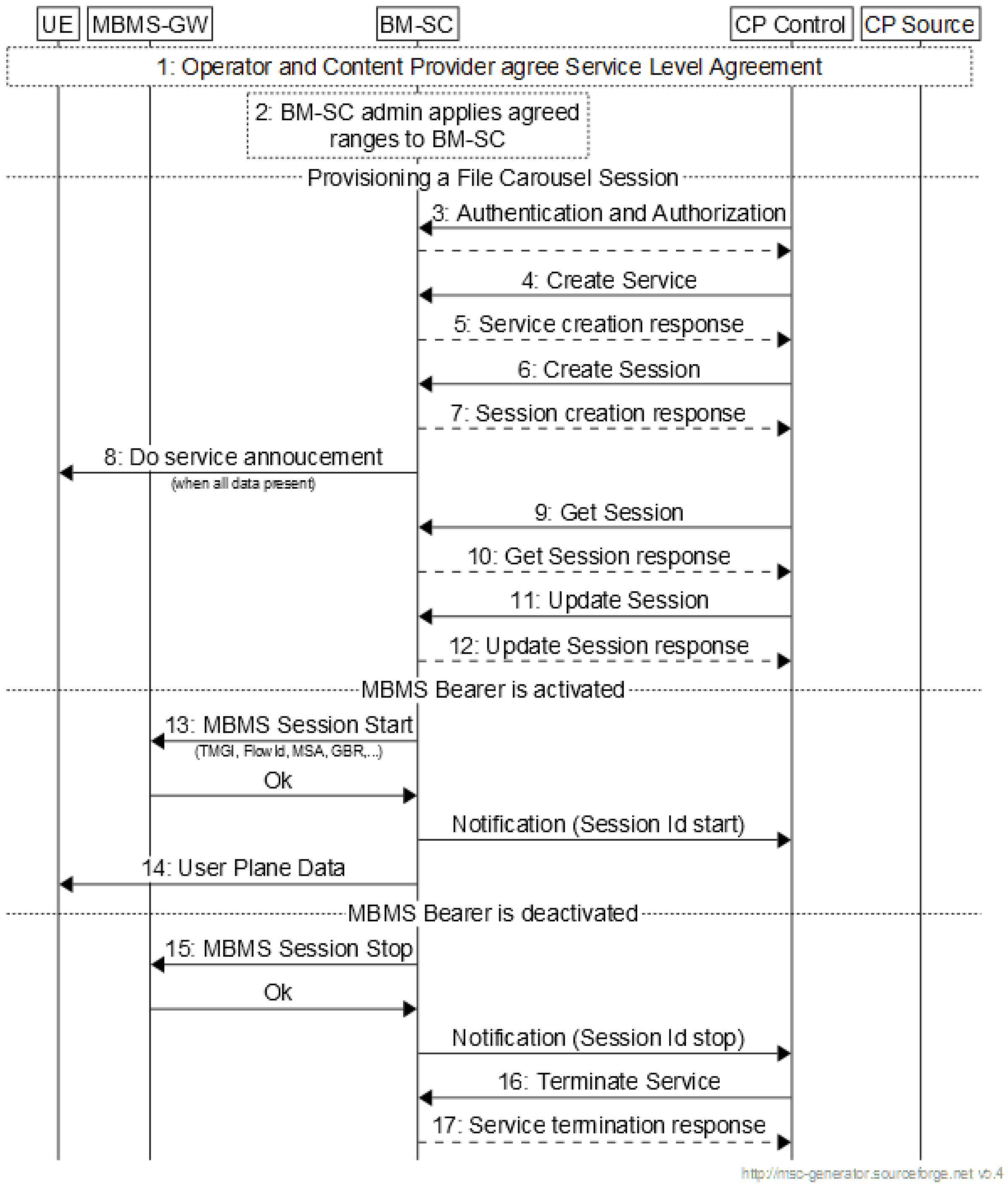

UE
MBMS-GW
BM-SC
CP Control
CP Source
1: Operator and Content Provider agree Service Level Agreement
2: BM-SC admin applies agreed ranges to BM-SC
Provisioning a File Carousel Session
3: Authentication and Authorization
4: Create Service
5: Service creation response
6: Create Session
7: Session creation response
8: Do service annoucement
(when all data present)
9: Get Session
10: Get Session response
11: Update Session
12: Update Session response
MBMS Bearer is activated
13: MBMS Session Start
(TMGI, Flow Id, MSA, GBR,...)
Ok
Notification (Session Id start)
14: User Plane Data
MBMS Bearer is deactivated
15: MBMS Session Stop
Ok
Notification (Session Id stop)
16: Terminate Service
17: Service termination response
http://msc-generator.sourceforge.net v3.4

Figure 5.3.5-1: File Carousel Provisioning and Ingestion Procedure

1: The operator and the Content Provider agree a Service Level Agreement, which entitles the Content Provider to use the MBMS system (in accordance to some rules) for content delivery. For instance, the SLA can include day time ranges, during which the Content Provider can distributed content. The SLA may also include geographical areas, in which the Content Provider is allowed to distribute content. The SLA also includes target bitrates and likely definitions of tolerable losses per service.

2: The BM-SC administrators (operator) apply the agreed ranges. This can imply to add additional Service Areas, and other system related configurations.

The Content Provider provisioning a file delivery session in a single broadcast area.

3: The Content Provider authenticates itself as authorized user. The Content Provider can only see those configurations, sessions and services, which belong to the Content Provider. On successful authentication, an access token is provided to CP and will be present in every subsequent message sent by CP to BM-SC.

4: The Content Provider creates a new Service by providing service class, service languages, service names, notification configuration as well as consumption reporting configuration. The Content Provider can select whether the CP or the operator distributes service announcement by providing a list of Service Announcement Channel (SACH, as defined in Annex L.2 / L.3 of [3]) services used for operator-driven service announcement.

NOTE 1: BM-SC derives the required UE capabilities from the provided service and session properties.

5: Upon successful Service creation by the BM-SC, the BM-SC will provide a unique Service Id, that the Content Provider will use for subsequent requests.

6: The Content Provider creates a Session for previously created Service.

Common session properties provided as input by Content Provider: unique Service Id this session belongs to, max ingest bitrate (excluding any FEC redundancy and transport overhead), scheduling information (start time, stop time), Geographical Area, QoE Reporting configuration and session type (set to Files).

Files specific session properties provided as input by Content Provider: ingest mode (pull/push), file list if pull mode. In this use-case Content Provider provides a Keep Update Interval for files.

NOTE 2: BM-SC allocates following parameters for SDP: TMGI, FLUTE IP Multicast Address, UDP Port and TSI.

NOTE 3: BM-SC derives the SAI list from Geographical Area provided in Content Provider request and from PLMN id negotiated in step 1. FEC information (codec and ratio) and MBMS Bearer QoS (ARP, QCI) are also negotiated in step 1.

NOTE 4: In pull ingest mode, file URLs can be provided now or at a later stage through the Session Update procedure.

NOTE 5: Service Announcement start time can be provided in request. If not, BM-SC starts announcing service as soon as all required service and session properties are provided by Content Provider

NOTE 6: In the case of regional services, i.e. that deliver region specific content, a Session can be cloned so that all Sessions of user service use same FLUTE parameters.

NOTE 7: A unique Session Id, which identifies the created Session, is responded. If push ingest mode is used, BM-SC provides also the push URL the Content Provider will use.

NOTE 8: For file URLs provided in Session creation request, the BM-SC starts automatically to fetch the file resource(s) from the content location when file earliest fetch time elapses and generates the FLUTE and FEC symbols (if any). The BM-SC can notify the Content Provider when the process is finalized.

8: Once all information for service announcement is available, and if service announcement start time is elapsed, the BM-SC starts announcing the service automatically. Service announcement is automatically updated following subsequent Session updates.

9: The Content Provider queries the Session configuration, providing Session Id.

10: The BM-SC provides the information in response.

11: The Content Provider updates the session by providing additional File URLs.

12: The BM-SC sends response with update status.

NOTE 9: The BM-SC starts automatically to fetch the new file resource(s) from the content location when file earliest fetch time elapses and generates the FLUTE and FEC symbols (if any). The BM-SC can notify the Content Provider when the process is finalized.

NOTE 10: Steps 9-12 can be executed at any time after Session is created and prior to the Session stop time. Any file URL added after Session start time will be automatically fetched, processed and sent on user plane.

The BM-SC activates automatically the MBMS Bearer at Session start time.

13: The BM-SC activates the MBMS bearer by providing the TMGI, the Flow ID, the MBMS Service Area (MSA), the GBR and other parameters to the MBMS-GW. The BM-SC can notify the content provider about the activation of the MBMS Bearer.

14: The BM-SC activates the MBMS Broadcast bearer according to the time schedule. When the MBMS Broadcast bearer is activated, then the BM-SC MBMS Delivery Function (MDF) starts sending the user plane data.

NOTE 11: The BM-SC starts to check files update on content location using Keep Update Interval value. Any updated file is automatically fetched, processed and replaced in user plane flow, taking into account File repeat parameter. The BM-SC can realize file repetitions in various ways, e.g. simply repeating the source data of the file or send only FEC redundancy information during file repetitions.

15: At Session stop time, the MBMS bearer is terminated. The BM-SC can notify the content provider about the termination of the MBMS Bearer.

16: The Content Provider terminates the service. All sessions, which are still created or active will be deleted automatically by BM-SC with the termination of the service.

17: BM-SC sends service termination response.

5.3.6 Provisioning and Ingestion Procedure for File Delivery Services using continuous Push ingest (without File Schedule)

A provisioning and ingestion procedure in this section describes a realization of an "TV Program Guide update delivery" (Use-Case in clause 4.5) use-case.

6 Security aspects

The provisioning use-cases in clause 5 use HTTP for provisioning (Control Plane) and content ingest (user plane). Usage of TLS is a common industry practice to confidentiality protect the communication of HTTP end-points. The usage of HTTP over TLS protocols for user plane ingest is recommended.

7 Survey of Existing Protocols for a Provisioning Interface

The interface specification can be described in two different ways:

- programmatic interface, in which case the interface specification can be described as a set of procedures and functions of the server application that the client application can invoke in its program (e.g., as a library). For this kind of interface specification, the IDL representation presented in clauses 7 and 8 can be mapped directly to the target language as that of the client application.

- remote interface, in which case the interface specification can be described as a set of messages that carry message content with method names, and resources to access or modify. For this kind of interface specification, an interface specification protocol is needed that can used by the client application to exchange messages with the server application and invoke the procedures included in the remote interface

In this section, a list of protocols that can be used to specify the interface is presented. All protocols are presented in brief and then a recommendation is made.

1) Hypertext Transfer Protocol (HTTP):

   HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [3]. The HTTP/1.1 specification describes methodologies for message syntax and routing, semantics and content, conditional request, range requests, caching, and authentication.

HTTP protocol can be used a generic application level protocol for communication between any two Internet systems (commonly referred to as client and server in client server computing model). This protocol is used as request response protocol where the requests are issued by the client and the server serves those requests and responds back to the client as response messages. In this way, the client and server exchange different HTTP messages for varied tasks such as session control, payload exchange, capability queries, data population and binding, security, routing etc.

As a generic application level protocol, HTTP can be used an interface specification protocol for the interface. With its wide set of capabilities, the interface can be defined using HTTP protocol to specify the control and data plane functions of the interface.

2) Diameter:

The Diameter base protocol provides Authentication, Authorization, and Accounting framework for network applications (commonly referred to as Diameter applications). The base protocol specifies the message format, transport, error reporting, and security services to be used by Diameter applications. Further, base protocol provides support for failover, transmission-level security, reliable transport, agent support, server-initiated messages, transition support, capability negotiation, peer discovery and configuration.

Diameter applications exchange commands (command requests and answer pairs) to realize a service. Each command represents a specific action the Diameter client wants the Diameter server to perform. All data delivered by the protocol is in the form of AVPs. The base protocol defines a set of common Diameter commands and AVPs as part of the specification.

Diameter applications can extend the base protocol by defining additional commands and AVPs. With this flexibility, Diameter can be seen as a generic request response protocol where the Diameter client and the Diameter server communicate with each other to carry out necessary tasks to perform a service. Due to this generic behaviour, Diameter can be used as an interface specification protocol to define the set of methods, message formats, and agent behaviour for the interface.

3) EXtensible Markup Language (XML):

XML [4][5] is a text based encoding format for representing structured information such as documents, data, configuration etc. It is a simplified format for sharing information between networked systems, computer programs, and even humans. With detailed markup, representation, parsing, editing, and searching of data becomes considerably easier. Some of the advantages of XML include redundancy, self-describing, and simplicity.

With the capabilities to represent structured data, XML can be used to describe message and payload formats for communication between two networked systems. However, since XML is a data markup language, it is often used with other control protocols (e.g., HTTP) when two entities are communicating with each other. While the control protocols are used for session management purposes (e.g., setup and termination of sessions), XML could be used for data formats to represent data exchanged during those interactions.

Due to above reasons XML cannot be used as an interface specification protocol to specify the interface, but it can be used with an interface specification protocol to describe the data that is exchanged.

4) Representational State Transfer (REST):

REST is an architecture style for designing networked applications. This architectural style is based on six constraints – that the architecture should have a client server interface, the communication should be stateless, responses should be cacheable, it should be a layered system (with ability to have intermediate systems between client and server), code on demand (where the server can temporarily extend the functionality of a client by transfer of executable code), and a uniform interface (which facilitates familiarity, interoperability, and scalability). Almost all the time, REST client and server use HTTP protocol for their communication needs.

With REST, the application components are modelled as "resources", and the interaction between a REST client and a REST server often involves reading, creating, or updating the state of the said resources. By modelling application components as resources, the application can expose "RESTful APIs" that specify the available methods to interact with the server application. All these APIs can be invoked using standard HTTP methods such as GET, PUT, POST, DELETE, HEAD, and OPTIONS. As a result, the interactions between REST client and server become much simpler and lightweight.

The interface between MBMS service provider and BM-SC can be specified as RESTful API by modelling the application components at BM-SC as RESTful resources. With RESTful APIs for interacting with BM-SC, the transport protocol between the service provider and BM-SC would be HTTP, and the client at the service provider would use HTTP methods to invoke operations on the RESTful resources using the above RESTful APIs.

5) Simple Object Access Protocol (SOAP):

SOAP is a protocol for exchanging structured information between peers in a distributed environment. SOAP uses XML for message format and application layer protocols such as HTTP or SMTP etc. as a base transport protocol. SOAP protocol mainly defines three parts – a message envelop that describes message format and rules for processing it, encoding rules for representing application data, and a convention for representing remote procedure calls and responses.

A SOAP sender and a SOAP receiver can exchange SOAP messages (encoded using XML) and thereby implement a pattern of requests and responses. The request and response protocol is based on a defined programing interface published by the SOAP receiver. However, unlike in REST architecture where the APIs exposed are for manipulation (create, delete, and update) of application resources (data), the programing interface in SOAP is primarily based on the procedures supported at the receiver that drive application logic. As a result, the sender has to have detailed knowledge of the receiver's interface, which could become an issue if the receiver's interface is complex and the sender is forced to perform multiple intermediate steps to realize a service.

The interface can be specified using SOAP where the procedures supported could be exposed as an interface for the SOAP client at service provider to use. The client at the service provider would then use SOAP to interact with the SOAP receiver to execute the services provided by the receiver's interface procedures.

Recommendation: After surveying the probable set of protocols that could be used as interface specification protocols, RESTful APIs is recommended to provide the interface specification for the interface. The benefits of RESTful APIs outweigh the complexities of other protocols. RESTful APIs not only simplify the interface specification, but also simplifies implementation tasks and has lesser overhead compared to other protocols.

8 Conclusions and Recommendations

It is recommended to consider the realizations of use-cases 4.1, 4.2 and 4.5 in the stage 2 procedure definition of the new external interface to 3rd Party Content Providers (AE_enTV-MI_MTV work item). Additionally, it is recommended to consider the Non-TV-centric use cases 4.3 (VOD Prepositioning) and 4.4 (Software update) in the stage 2 procedure definition.

It is recommended RESTful APIs to provide the interface specification for the interface. The benefits of RESTful APIs outweigh the complexities of other protocols. RESTful APIs not only simplify the interface specification, but also simplifies implementation tasks and has lesser overhead compared to other protocols.

It is recommended to use OAI (formerly known as Swagger) as modelling language for RESTful APIs.

After studying content ingestion into the BM-SC for Live and File Delivery services, the usage of HTTP over TLS protocols for user plane ingest is recommended. Usage of HTTP over TLS for control and user plane transactions is a common practice in the industry. Existing 3GP DASH Segments can push segments using HTTP push procedures, such as WebDAV.

Annex A: Provisioning Procedure Overview

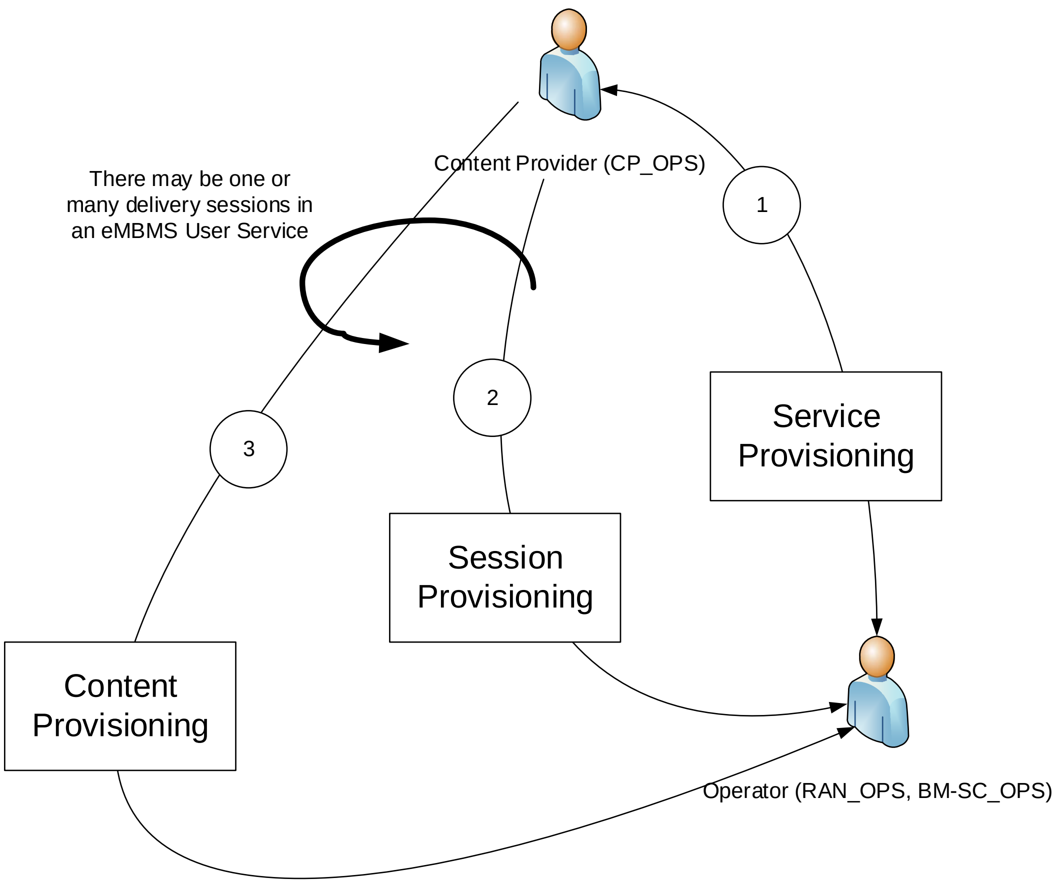
A.1 Service, Session and Content Provisioning Overview

NOTE: This annex contains one example for a provisioning procedure. This example does not consider provisioning for MooD enabled services.

A.1.0 Workflow

The entire workflow is sub-divided into three provisioning steps, being Service, Session and content provisioning.

The following figure illustrates how Service Provisioning is undertaken once (defining the Service and the planned APP(s)) and Session and Content provisioning are repeated, depending on the number of Delivery Sessions appropriate to the Service and thus defined in Service Provisioning.



Content Provider (CP_OPS)
There may be one or many delivery sessions in an eMBMS User Service
1
2
3
Service Provisioning
Session Provisioning
Content Provisioning
Operator (RAN_OPS, BM-SC_OPS)

Figure A.1 Illustrating the Main Provisioning Workflow

For example, if the planned user service defines a UE application requiring 3 DASH Streaming feeds and 2 on-demand file download content sources, then the Session and Content provisioning steps are run 5 times.

A simple but generic actor grouping is defined that would suit the business requirements both today and into the foreseeable future. The actors chosen here are the Content Provider operations people and the Operator operations people.

Depending on the business relationships these people may either be fulfilling a role within the operator's business (meaning both actor groups are operator trusted employees but from different organizations), or may be people employed by separate, different and several businesses in their own right, but with a business affiliation with the operator to deliver their content over the operator's transport network.

The separation of the provisioning steps and the actors is independent whether the Content Provider is an "internal Content Provider" or an "external / 3rd Party Content Provider". The security and privacy procedures surrounding the operations may differ, simply put – the operator trusts his employees, the operator may have varying trust with different Content Providers and Content Providers are unlikely to trust each other (i.e. 3rd Party Content Providers should be shielded against each other).

A.1.1 Service Provisioning

The Service Provisioning phase is largely concerned with negotiation concerning the planned service. It includes a lot of Service design and iterative discussion which takes place in meetings, phone calls, emails etc. This stage would include APP design and allocation of ServiceClass, ServiceID information.

There comes a point when the transport aspects of the service design needs to be checked and "committed" towards the Content Provider. This will involve Network Analysis - running RAN capacity analysis to determine if the service could be committed (resources available and no collisions exist in the Broadcast Area (i.e. MBMS Service Area, represented as a SAI list) arising from (collisions) other committed broadcasts services using resource allocation in those areas.

The Network Analysis includes assessment of the RAN site in order to establish the MBMS Single Frequency Network (MBSFN). In particular suitable Modulation and Coding Schemes (MCS) should be identified in combination with an amount of Application Layer FEC to reach the negotiated service quality. The feasible Modulation and Coding Schemes with AL-FEC for the MBSFN defines feasible service layer bitrates (GBR).

When the bitrates in the potential broadcast areas can be committed, then an agreement can be made between the operator and Content Provider followed by the Service Layer and RAN, Packet Core provisioning of SAIs.

There is no traffic transmission associated with this process – it is purely provisioning.

A.1.2 Session Provisioning

This is the main provision procedure which has to be repeated for each Delivery Session specified in the Service provisioning process. It requires a number of steps in order to setup all the required eMBMS and Delivery Session Instances. It involves checking the negotiated content bit rate budget negotiated in Service Provisioning is not exceeded and it also involves providing content ingestion points towards Content Provider.

As for Service Provisioning, there is no traffic transmission associated with this process. Individual delivery session instances are planned and potentially already announced on the Service Announcement Channel.

The Content Provider may indicate during session provisioning, whether QoE or consumption reporting feedback is expected. The operator may activate QoE and / or consumption reporting independently from the Content Provider selection in order to understand the service quality.

The Content Provider selects the content ingestion procedure during Session Provisioning. Different types of ingestion procedures are possible, depending on the Use-Case:

- WebDAV (HTTP) for continuous data like media segments or other small files. Here, the BM-SC forwards each file immediately via the MBMS download delivery session, which is associated to the WebDAV folder. During Session Provisioning, the BM-SC provides a unique WebDAV Folder URL as content ingestion point description to the Content Provider.
- CacheAdd & AddContent for large files in case of on-request transactions: during in case of large files, where the download duration or the file partitioning duration or the FEC redundancy duration is of unknown duration, the Content Provider should have a separate cache procedure (i.e. CacheAdd) and send procedure (AddContent

to delivery session instance). During session provisioning, the BM-SC provides a unique delivery session instance id to the Content Provider, so that subsequence Cache and Add commands can be uniquely associated to the a delivery session instance.

A.1.3 Content Provisioning

This provisioning step targets the correct content to the correct transmission BM-SC(s) and then ensuring ingested the traffic flow using the content ingestion points provided to the Content Provider during Session Provisioning.

There are different types of provisioning for live DASH and on-request ingestion and delivery, requiring differing ingestion and transmission. The Content Provider received the content ingestion point identifies (e.g. a WebDAV Folder URL or a delivery session instance id) during the session provisioning step.

For on-request delivery of large files like binary data or VOD clips, the Content Provider may define a transmission start time or may provide a relative sequence for each file in the delivery session instance. The start time may be provided with the AddContent command.

Content Provisioning is traffic related and requires the corresponding availability and reliability.

Annex B: Abstract Representation of API using IDL

B.1 Introduction

This section describes abstract representation of protocol interface API using the chosen IDL

For describing the interface, different interface description languages that are in use today are presented. In clause 4.1, the most commonly used interface description languages with a recommendation are presented. In clause 4.2, the recommended IDL used to describe the interface are presented.

B.2 Survey of Interface Description Languages

An IDL is a specification language for describing the programming interface of a system. For describing the interface, a protocol independent IDL to abstractly describe the capabilities and services of the interface can be used. In this section, commonly used IDL languages are presented, and the section ends by providing a recommended IDL for describing the interface.

1) Protocol Buffers:

   Protocol Buffers is an IDL for describing structured data and programs for generating software code that can translate the description to an actual software implementation, which can then be used to parse the above structured data. The client and server share the data structures (called messages) and data formats represented by Protocol Buffers IDL. Upon program invocation, the IDL representation of messages and data formats are translated to program code that both the client and server understand and use them for their communication needs.

   For the interface, Protocol Buffers IDL can be used to represent the message and data formats that all entities clearly understand and have decoding logic to parse those messages and data.

2) Apache Thrift:

   Apache Thrift is an IDL to define "Thrift" types based on the services and interfaces implemented by the server. The Thrift IDL description is then converted to software code using the Thrift code generator which can then be used by the client to call the server implemented services. As a result, the server supported services, represented using Thrift IDL, can be translated to software interfaces which can then be invoked by the client.

   All entities can use Thrift IDL to describe their supported services which can then be translated into software interfaces that the other parties can invoke using a message protocol such as HTTP.

3) Apache Avro:

   Avro is a higher level IDL used to describe protocol definitions. Each protocol definition describes a set of types (data) and messages (for interaction between the agents). Avro enables definition of user defined data types (e.g., based on enumeration, primitive types, and reference types) and messages (RPC messages, message formats etc.). Different agents in the system provide and share protocol definitions so all agents have a common interface to access services and interfaces provided by other agents.

   All entities can use Avro to describe their protocol definitions (data types, messages, message formats etc.) so each party has the detailed descriptions of the services supported by the other party.

4) Web IDL:

   Web IDL is an interface definition language that is used to define interfaces for APIs in the web platform. The APIs in the web platform include one or more IDL fragments that describe the interfaces (state and behaviour that objects can exhibit). IDL fragments allows for description of different kinds of definitions such as interfaces, partial interface definitions, dictionaries, partial dictionary definitions, exceptions, typedefs and implements statements. Different objects in the system can provide IDL fragments to describe such kinds of definitions.

All entities modelled as web applications can provide IDL fragments describing their interfaces to specify the state and behaviour they exhibit. With this kind of IDL representation, the two entities are aware of each other's web API (interface) and therefore can interact with each other using a web message protocol such as HTTP.

5) WSDL:

WSDL is a description language for defining web services. It uses XML protocol and often used with SOAP to describe the web services supported by a web application. WSDL describes web services as a set of end points operating on messages encoded using XML documents. The service end point definitions (e.g., messages and operations on those messages) are abstractly described so any client can fetch the service definition using network protocols and use the service. WSDL also provides extensibility where related end points can be grouped to form an abstract endpoint (services).

If SOAP is being used as an interface specification language to specify the interface, then WSDL can be used as a service description language to describe the web service endpoints supported by the entities. Based on end point service definitions (services), the service provider entities will be able to fetch the definitions and use a network message protocol (e.g., HTTP with SOAP) to consume those services.

6) OMG IDL:

OMG is an interface description language that allows for interface specification which the clients can use to call in their programs. The object implementations on the server are abstracted in the interface description, but provide sufficient information for the client (e.g., number and type of parameters, method name, return type etc.) to call the methods given in the description.

The protocol interface can use OMG IDL to describe the methods and procedures supported by the interface. These methods and procedures can be abstractly described using OMG IDL without specifying implementation details. With such kind of an interface description, all entities will be able to invoke the procedures described in OMG IDL for the protocol interface.

Recommendation: After surveying all the available interface description languages, it is believed that using OMG IDL as the interface description language for describing the interface is appropriate. OMG IDL descriptions can be provided at the API level with detailed information about the APIs to the client without burdening the client with implementation details of the said APIs. Further, since the OMG IDL interface description can be mapped to almost all of the client implementation languages, it is easy for the client to understand and start using the above interface.

B.3 OMG IDL (Object Management Group Interface Description Language)

OMG IDL is a description language that allows for interface specification which the clients can use to call in their programs. The server system's identifiers (e.g., variables) and procedures (e.g., functions and parameters) are defined using the OMG IDL interface specification using which the client application developers can build the client applications.

OMG IDL provides all the constructs to describe an interface (e.g., literals, attributes, types, grammar, specification rules etc.). OMG IDL also provides capabilities for general language level features (e.g., inheritance, type declaration, exception handling etc.) while generating the interface description. Once the interface description is generated (or written), the interface can then be mapped to a language that the client can understand so the interface methods can be invoked during client execution.

An application's IDL specification is usually built using one or more modules. Each module provides a description of the set of identifiers and procedures of different participants involved in the architecture. For example, a basic module on a server application can be defined as follows:

```
module ExampleModule {

// Identifiers and procedures of different participants in the system

}
```

Within the *EampleModule* module described above, a set of interfaces can be defined with each interface corresponding to a participant in the system. For example, the two interfaces "*Client*" and "*Server*" describe the identifiers and procedures by the client system and server system respectively in a traditional client server model.

```
module ExampleModule {

    interface Client {

        //Identifiers and procedures of client system

    };


    Interface Server {

    // Identifiers and procedures of server system

    };

}
```

Each of the *Client* and *Server* interfaces in the IDL description above can individually define the identifiers and procedures for the respective application/system. For example, the Client interface can define identifiers and procedures as shown below:

```
interface Client {

    long a;     // identifier/ variable

    long sum (in long b, in long c); // procedure that returns the sum of two numbers

};
```

In addition to defining basic interfaces, OMG provides capabilities for:

- extending the interfaces to generate derived interfaces (using the inheritance mechanism)
- different types of abstract and local interfaces
- different types of declarations such as value declarations, constant declarations, type declarations, exception declarations, operation declarations, attribute declarations, event declarations, and component declarations
- scoping rules and name resolution

Once, all the interfaces are defined and the IDL description is completed, the description can be mapped to client and/or server's language. The mapped description can then be used directly by the client in its implementation without complete understanding of the detailed server's implementation. As a results, this enables the client and server development to progress in parallel as long as they adhere to the agreed upon interface description and specification.

Annex C: API Modelling Languages

C.1 Introduction

This Annex contains an overview of different REST API modelling languages, which are currently used in the industry.

C.2 RESTful API Modeling Language (RAML)

RAML is a vendor independent open specification language that is based on JSON. RAML is used to describe RESTful APIs.

RAML provides a structured format that allows for the description of the API, the endpoints, and the HTTP methods that are used for the API communication, as well as the data format for the exchanged data over the API. RAML covers the whole development stage extending from the specification design, the documentation generation, as well as the implementation of the API and its testing.

A RAML file is a text file that starts with a line that indicates the version number of the RAML specification that is used in this file. A RAML document has a root section that describes the basic information about the API such as its title and version number. It also defines the assets that are used elsewhere in the RAML document. A brief listing of the key nodes in the root of the RAML document is given in the following table:

Table C.2-1

Name	Description
Title	The title of the API.
description	A description of the API.
Version	The version of the API.
baseUri	A URI that will be used as the base for all relative URIs in the RAML document.
Protocols	Indicates what protocols are supported by the API, e.g. HTTPS.
mediaType	Provides the mime type of all request and response payloads. For example "application/json".
Types	Provides a list of the resource types used by this API. Resource types are similar to classes and provide patterns that can be reused.
Traits	Provides a list of the traits used in this API. A trait defines methods that can be reused and extended.

An example of a RAML document is provided here for information:

```
#%RAML 1.0
title: API with Types
types:
  User:
    type: object
    properties:
      firstname: string
      lastname:  string
      age:       number
/users/{id}:
  get:
    responses:
      200:
        body:
          application/json:
            type: User
```

In this example, users are defined as resources that follow the type User and that are identifier through an id value. It shows also that successful get request will return the user information encoded in JSON as part of the response body.

C.3 Open API Initiative (successor of Swagger)

Open API is a widely adopted vendor independent open specification language that is based on JSON (YAML, being a superset of JSON, can be used as well to represent an Open API specification file). Open API is used to describe RESTful APIs. Starting January 1st 2016 the Swagger Specification has been donated to the Open API Initiative (OAI) and is the foundation of the Open API Specification.

Open API provides a structured format that allows for the description of the API, the endpoints, and the HTTP methods that are used for the API communication, as well as the data format for the exchanged data over the API. Open API covers the whole development stage extending from the specification design, the documentation generation, as well as the implementation of the API and its testing.

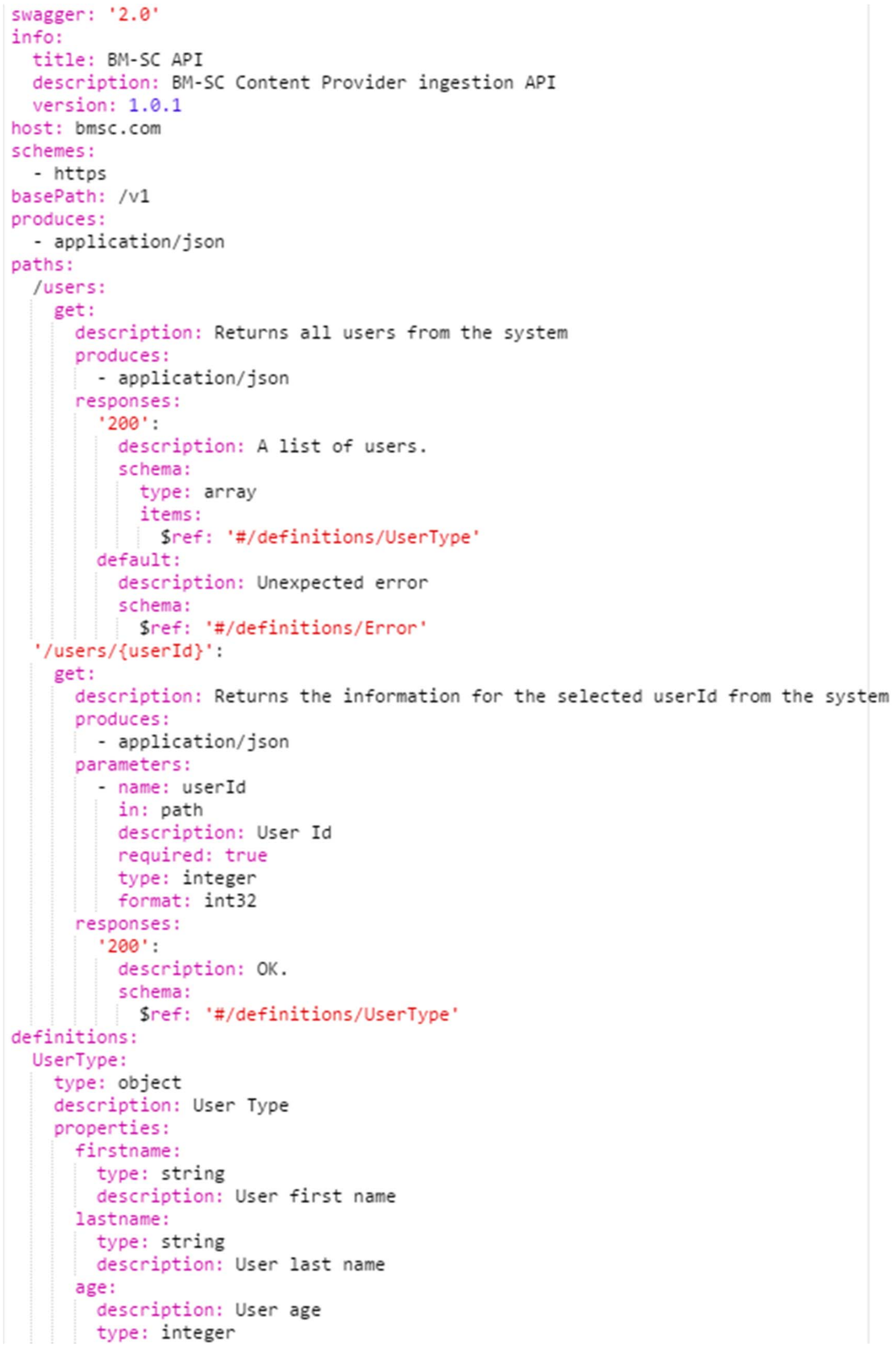
An Open API file is a JSON file (single file or many if needed) that starts with the version number of the Open API specification that is used in this file. An Open API document has an "info" section that describes the basic information about the API such as its title, description and version number. A brief listing of the key nodes used in the Open API document is given in the following table:

Table C.3-1

Field Name	Type	Description
swagger	`string`	Required. Specifies the Swagger Specification version being used. It can be used by the Swagger UI and other clients to interpret the API listing. The value is "2.0".
info	Info Object	Required. Provides metadata about the API. The metadata can be used by the clients if needed.
host	`string`	The host (name or ip) serving the API. This is the host only and does not include the scheme nor sub-paths. It MAY include a port. If the host is not included, the host serving the documentation is to be used (including the port). The host does not support path templating.
basePath	`string`	The base path on which the API is served, which is relative to the host. If it is not included, the API is served directly under the host. The value starts with a leading slash (/). The basePath does not support path templating.
schemes	[`string`]	The transfer protocol of the API. Values are taken from the list: "http", "https", "ws", "wss". If the schemes is not included, the default scheme to be used is the one used to access the Swagger definition itself.
consumes	[`string`]	A list of MIME types the APIs can consume. This is global to all APIs but can be overridden on specific API calls. Value has to be as described under Mime Types.
produces	[`string`]	A list of MIME types the APIs can produce. This is global to all APIs but can be overridden on specific API calls. Value has to be as described under Mime Types.
paths	Paths Object	Required. The available paths and operations for the API.
definitions	Definitions Object	An object to hold data types produced and consumed by operations.
parameters	Parameters Definitions Object	An object to hold parameters that can be used across operations. This property does not define global parameters for all operations.
responses	Responses Definitions Object	An object to hold responses that can be used across operations. This property does not define global responses for all operations.
securityDefinitions	Security Definitions Object	Security scheme definitions that can be used across the specification.
security	[Security Requirement Object]	A declaration of which security schemes are applied for the API as a whole. The list of values describes alternative security schemes that can be used (that is, there is a logical OR between the security requirements). Individual operations can override this definition.
tags	[Tag Object]	A list of tags used by the specification with additional metadata. The order of the tags can be used to reflect on their order by the parsing tools. Not all tags that are used by the Operation Object have to be declared. The tags that are not declared may be organized randomly or based on the tools' logic. Each tag name in the list is unique.
externalDocs	External Documentation Object	Additional external documentation.

An example of an Open API document is provided here for information (in JSON then YAML). In this example, users are defined as resources that follow the type UserType and that are identified through an id value. It shows also that successful get request will return the user information encoded in JSON as part of the response body.

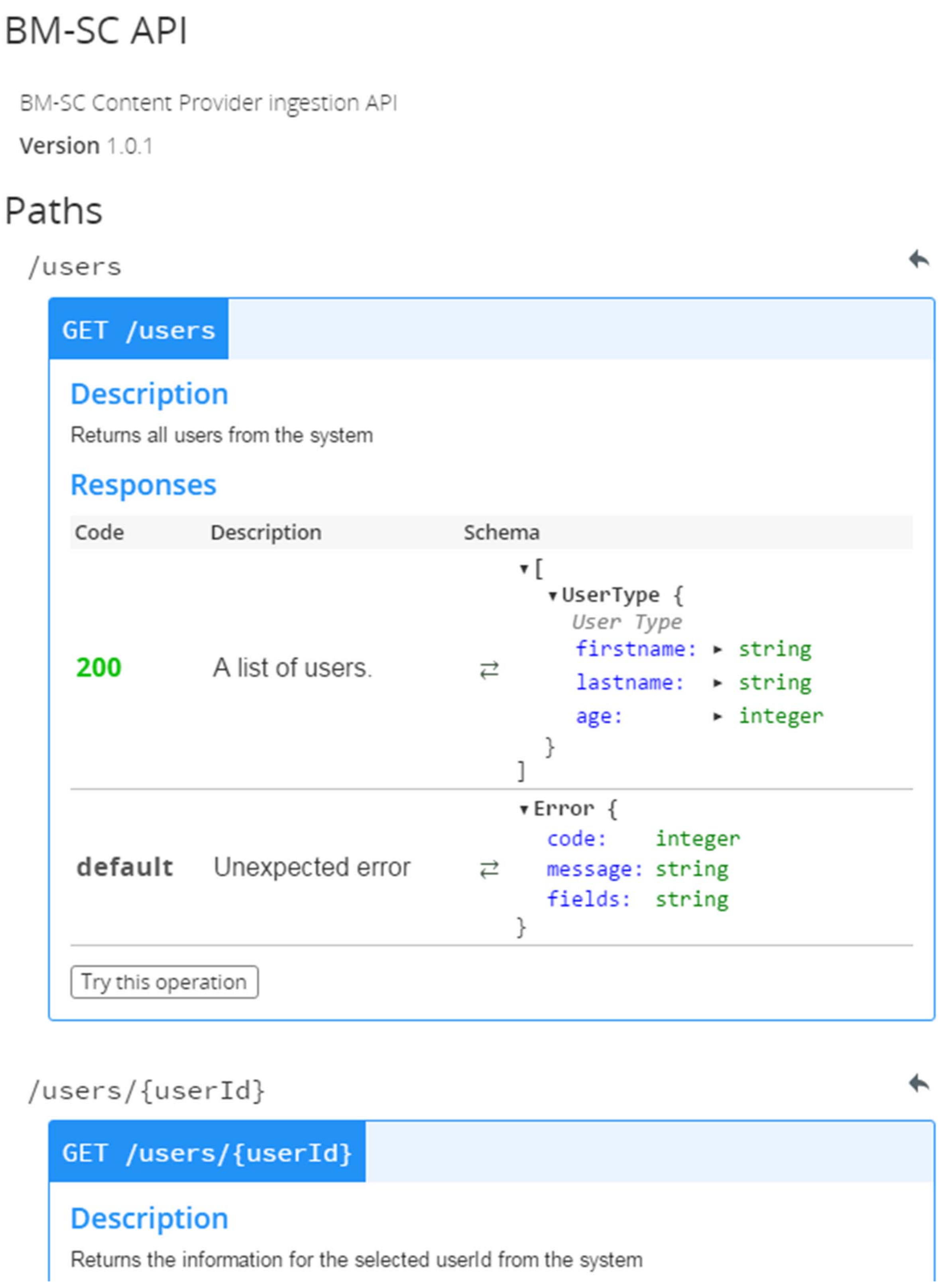
```
{
   "swagger": "2.0",
   "info": {
      "title": "BM-SC API",
      "description": "BM-SC Content Provider ingestion API",
      "version": "1.0.1"
   },
   "host": "bmsc.com",
   "schemes": [
      "https"
   ],
   "basePath": "/v1",
   "produces": [
      "application/json"
   ],
   "paths": {
      "/users": {
         "get": {
            "description": "Returns all users from the system",
            "produces": [
               "application/json"
            ],
            "responses": {
               "200": {
                  "description": "A list of users.",
                  "schema": {
                     "type": "array",
                     "items": {
                        "$ref": "#/definitions/UserType"
                     }
                  }
               },
               "default": {
                  "description": "Unexpected error",
                  "schema": {
                     "$ref": "#/definitions/Error"
                  }
               }
            }
         }
      },
      "/users/{userId}": {
         "get": {
            "description": "Returns the information for the selected userId from the system",
            "produces": [
               "application/json"
            ],
            "parameters": [
               {
                  "name": "userId",
                  "in": "path",
                  "description": "User Id",
                  "required": true,
                  "type": "integer",
                  "format": "int32"
               }
            ],
            "responses": {
                "200": {
                    "description": "OK.",
                    "schema": {
                        "$ref": "#/definitions/UserType"
                    }
                }
            }
        }
    }
},
"definitions": {
    "UserType": {
        "type": "object",
        "description": "User Type",
        "properties": {
            "firstname": {
                "type": "string",
                "description": "User first name"
            },
            "lastname": {
                "type": "string",
                "description": "User last name"
            },
            "age": {
                "description": "User age",
                "type": "integer"
            }
        }
    }
}
}
```



```yaml
swagger: '2.0'
info:
  title: BM-SC API
  description: BM-SC Content Provider ingestion API
  version: 1.0.1
host: bmsc.com
schemes:
  - https
basePath: /v1
produces:
  - application/json
paths:
  /users:
    get:
      description: Returns all users from the system
      produces:
        - application/json
      responses:
        '200':
          description: A list of users.
          schema:
            type: array
            items:
              $ref: '#/definitions/UserType'
        default:
          description: Unexpected error
          schema:
            $ref: '#/definitions/Error'
  '/users/{userId}':
    get:
      description: Returns the information for the selected userId from the system
      produces:
        - application/json
      parameters:
        - name: userId
          in: path
          description: User Id
          required: true
          type: integer
          format: int32
      responses:
        '200':
          description: OK.
          schema:
            $ref: '#/definitions/UserType'
definitions:
  UserType:
    type: object
    description: User Type
    properties:
      firstname:
        type: string
        description: User first name
      lastname:
        type: string
        description: User last name
      age:
        description: User age
        type: integer
```

Figure C.3-1

Below is an example of a tool suite (Swagger Editor) that displays both the API source file and the automatically generated documentation for this source file:



BM-SC API

BM-SC Content Provider ingestion API

Version 1.0.1

Paths

/users

GET /users

Description

Returns all users from the system

Responses

Code	Description	Schema
200	A list of users.	▾[▾UserType { *User Type* firstname: ▸ string lastname: ▸ string age: ▸ integer }]
default	Unexpected error	▾Error { code: integer message: string fields: string }

Try this operation

/users/{userId}

GET /users/{userId}

Description

Returns the information for the selected userId from the system

Figure C.3-2

Annex D: MB2 Reference Point

D.1 Introduction

As part of Release 13, 3GPP developed and specified the Group Communication System Enabler for LTE (GCSE_LTE) and MB2 reference point in 3GPP TS 23.468 [6]. The MB2 control and data plane interfaces (MB2-C and MB2-U respectively) enable a Group Communication Service Application Server (GCS AS) to establish a connection with a BM-SC and feed data into the BM-SC to be transmitted over MBMS broadcast. The GCS AS acts as an external Content Provider using the BM-SC for broadcasting content to the users.

The MB2 reference point between the GCS AS and BM-SC provides the ability:

- To the GCS AS application to request for allocation of a set of new TMGIs or renewal of expiration time for already allocated TMGIs.
- To the GCS AS application to request for de-allocation of a set of TMGIs (and their corresponding bearers) irrespective of their expiration time.
- To the GCS AS application to request for activation, deactivation, and modification of MBMS bearers and their corresponding resources.
- To BM-SC to notify the GCS AS application of status of MBMS bearers.
- To BM-SC to notify the GCS AS application regarding the timer expiry of a TMGI.

Based on the procedures requested by the GCS AS using the MB2 interface, the BM-SC invokes corresponding procedures on the SGmb/Sgi-mb interface to the MBMS GW.

D.2 Shortcomings of MB2

The MB2 reference point between the GCS AS and BM-SC provides capabilities for TMGI management (allocation/deallocation/modification), MBMS bearer control (activation/deactivation/modification) and status notification procedures. However, there are clear drawbacks/limitations to the MB2 reference point such as the following:

- MB2 only allows for setup of MBMS services using group delivery method. Other different delivery methods such as download delivery and streaming delivery are not supported.
- The procedures defined for the MB2 reference point do not allow the GCS AS to benefit from some of the functionality that is offered by the BM-SC to monitor and improve the QoS over the broadcast channel such as QoE reporting, consumption reporting etc. The GC1 interface, which is not defined in the specification 3GPP TS 23.468 [6], may have to be overloaded with similar functionality, leading to redundancy and inefficiency.
- The procedures defined for the MB2 reference point do not allow the GCS AS to delegate responsibility of some functions to BM-SC that are already supported at BM-SC. Instead these functions are duplicated at GCS AS further complicating the GCS AS functionality. For example, functions that could be delegated include FEC activation, service announcement etc.
- Further, it is left to the GCS AS and the GCS application on the UE to implement new features/enhancements (e.g., FEC). Based on this design, many GCS AS and GCS applications on the UE may have to implement the same set of features instead of using features already provided at the BM-SC

NOTE: GCS realizes Service Announcement over GC1. A Service Announcement Channel (SACH) as defined in Annexes L2, L3 of TS 26.346 or other Service Announcement / Discovery functions are not used.

In addition to the above, the MB2 reference point was developed keeping in mind the basic requirements for a group communication service. However, the detailed requirements of different group communication services differ and cannot be accommodated with the procedures supported by MB2 reference point. For example, the requirements for a group communication service such as MCPTT (for public safety) will be different from that of broadcast TV. It is difficult to extend the MB2 reference point to support the varied set of requirements, particularly in presence of group communication services with different criticalities.

Also, the MB2 reference point is based on Diameter Base protocol. A protocol which is simpler, dynamic, and flexible than Diameter could be an ideal protocol for defining the interface between the external Content Provider and the BM-SC.

Annex E: Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2017-03	75	SP-170034				Presented to SA#75 for approval	1.0.0
2017-03	75					Version for Release 14	14.0.0
2018-06	80					Version for Release 15	15.0.0
2020-07	-	-	-	-	-	Update to Rel-16 version (MCC)	**16.0.0**

History

Document history		
V16.0.0	September 2020	Publication