ETSI TS 126 264 V19.0.0 (2025-10)

TECHNICAL SPECIFICATION

Universal Mobile Telecommunications System (UMTS);
LTE;
5G;
IMS-based AR Real-Time Communication
(3GPP TS 26.264 version 19.0.0 Release 19)

Reference
RTS/TSGS-0426264vj00

Keywords
5G,LTE,UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

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

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

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**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 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.

In the present document, modal verbs have the following meanings:

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

can indicates that something is possible

cannot indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

will indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

will not indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

might indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

might not indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

is (or any other verb in the indicative mood) indicates a statement of fact

is not (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

1 Scope

The present document focuses on IMS-based conversational AR (Augmented reality) services. AR services can overlay media (e.g., video, audio, text, etc.) on top of the user's real perception. Conversational AR services as described by the present document typically include a bidirectional conversational A/V connection in addition to other non-real-time AR media for collaboration or communication between two or more users.

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 26.114: "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction".

[3] 3GPP TS 26.119: "Media Capabilities for Augmented Reality".

[4] 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2".

[5] 3GPP TS 24.229: "IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3".

[6] 3GPP TS 26.565: "Split Rendering Media Service Enabler".

[7] ISO/IEC 23090-14 AMD 2, Information technology — Coded representation of immersive media — Part 14: Scene description — Amendment 2: Support for haptics, augmented reality, avatars, Interactivity, MPEG-I audio, and lighting

[8] 3GPP TS 26.522: "5G Real-time Media Transport Protocol Configurations".

[9] 3GPP TR 26.813: Avatar Representation and Communication.

[10] ISO/IEC DIS 23090-39, Information technology — Coded representation of immersive media — Part 39: Avatar Representation Format.

[11] IETF RFC 9110: "HTTP Semantics".

[12] IETF RFC 9112: "HTTP/1.1".

[13] OpenAPI, "Open API specification", Online: https://spec.openapis.org/oas/v3.1.0.

[14] 3GPP TS 29.501: "5G System; Principles and Guidelines for Services Definition".

[15] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[16] IETF RFC 3611, "RTP Control Protocol Extended Reports (RTCP XR)".

[17] 3GPP TS 29.571: "5G System; Common Data Types for Service Based Interfaces; Stage 3".

[18] IETF RFC 9113: "HTTP/2".

[19] 3GPP TS 26.512: "5G Media Streaming (5GMS); Protocols".

3 Definitions of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in 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 TR 21.905 [1].

Animation data: Skeletal, blend shape set, and other animation-related information.

Animation stream: Timed animation data sequence used to animate the base avatar.

AR data: Collection of information to be exchanged among participants in a call with AR experience. It includes AR media and AR metadata.

AR media: Media (e.g., audio, video, text or image) that will be rendered by the AR-MTSI client as an overlay over the user's real perception. This includes traditional 2D media (e.g., a 2D audio stream rendered to be perceived by the user to originate from their left side) and 3D media (e.g., spatial audio and volumetric video).

AR metadata: Data that provides information on AR media and its rendering. This includes pose, spatial descriptions and scene descriptions.

AR-MTSI client: DCMTSI client supporting AR capabilities as defined by this specification.

AR MF: AR-MTSI client implemented by functionality included in the MF.

AR-MTSI client in terminal: An AR-MTSI client that is implemented in a terminal or UE. The term "AR-MTSI client in terminal" is used in this document when entities such as AR MF/MRF is excluded.

Asset: Independently accessible component of an avatar.

Avatar: Digital representation of a user.

Base avatar model: Personalized and animatable model of the user.

Avatar representation: Subset of a base avatar model of a user selected for a particular avatar call.

Avatar call: IMS call that uses an avatar representation for at least one participant in the call.

Split rendering: The procedure in which a UE offloads some of the media processing related to rendering tasks to a media function as considered for network centric AR IMS session procedures in TS 23.228 [4]

3.2 Symbols

Void

3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 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 TR 21.905 [1].

ADC	Application Data Channel
BAR	Base Avatar Repository
BDC	Bootstrap Data Channel
AS	Application Server
DC	Data Channel
DCSF	Data Channel Signalling Function
IBM	Inverse Bind Matrix
I-CSCF	Interrogating-CSCF
IMS	IP Multimedia Core Network Subsystem

LBS	Linear Blend Skinning
MF	Media Function
P-CSCF	Proxy-CSCF
S-CSCF	Serving-CSCF

4 System description

4.1 General

Typical conversational AR scenarios as envisioned in this document consist of an immersive AR call that may include the following conversational components:

- Real-time speech/audio that can comprise mono, stereo, and/or spatial audio.
- Real-time 2D video or 360-degree video that can be rendered as rectangular or spherical overlay in the AR experience.
- A real-time volumetric video of the user or an object that can be rendered in AR or MR.

In addition to the above conversational media, non-real-time objects may be exchanged over the data channel as well.

Both two-party and multiparty calls are possible. The AR experience may be unidirectional, i.e., only one party receives AR media and renders it, or it may be bidirectional, i.e., both parties receive and transmit AR media. The term AR-MTSI client includes both:

- an AR-MTSI client in terminal which is an AR device as defined in TS 26.119 [3] e.g., AR glasses, phone, Head Mounted Display (HMD) that has an XR Runtime for rendering an AR experience.
- AR MF/ that provides support for AR conversational services.

As an AR-MTSI client in terminal is a DCMTSI client in terminal with additional features for AR communication, the following requirements for a MTSI client terminal also apply for an AR-MTSI client in terminal:

- the interworking requirements in clause 12 of TS 26.114 [2],
- the jitter buffer management requirements in clause 8 of TS 26.114 [2],
- the packet loss handling requirements in clause 9 of TS 26.114 [2],
- the media and rate adaptation requirements in clause 10 and 17 of TS 26.114 [2], and
- the network preference management object in clause 15 of TS 26.114 [2],

NOTE: If an AR-MTSI client in terminal supports functionalities for MSMTSI client in terminal as specified in Annex S of TS 26.114 [2], the media and rate adaptation requirements in Annex S.8 of TS 26.114 [2] also apply for an AR-MTSI client in terminal.

4.2 Terminal architecture

The detailed XR client architecture is not in the scope of this specification. The XR baseline client architecture can be found in TS 26.119 [3]. The pre/post-processor component in terminal provides AR capabilities for processing output of peripherals and the input/output of encoders/decoders, which may include:

- XR runtime
- Scene manager
- Presentation engine
- XR source management

The AR-MTSI client has XR Runtime capabilities for rendering AR experience, e.g., spatial localization and mapping, etc., and can support local AR rendering and network-assisted split rendering based on client's capabilities. A UE may support multiple microphones, cameras or sensors.

An AR-MTSI client supports the protocol stack of a basic MTSI client as described in clause 4.2 of TS 26.114 [2]. For the specific AR communication instance, AR-MTSI client can select different IMS media channel to deliver AR data to IMS network or peer UE. In general, AR media components with real-time characteristics are transported via RTP session and AR metadata is transported via data channel or RTP session with AR media.

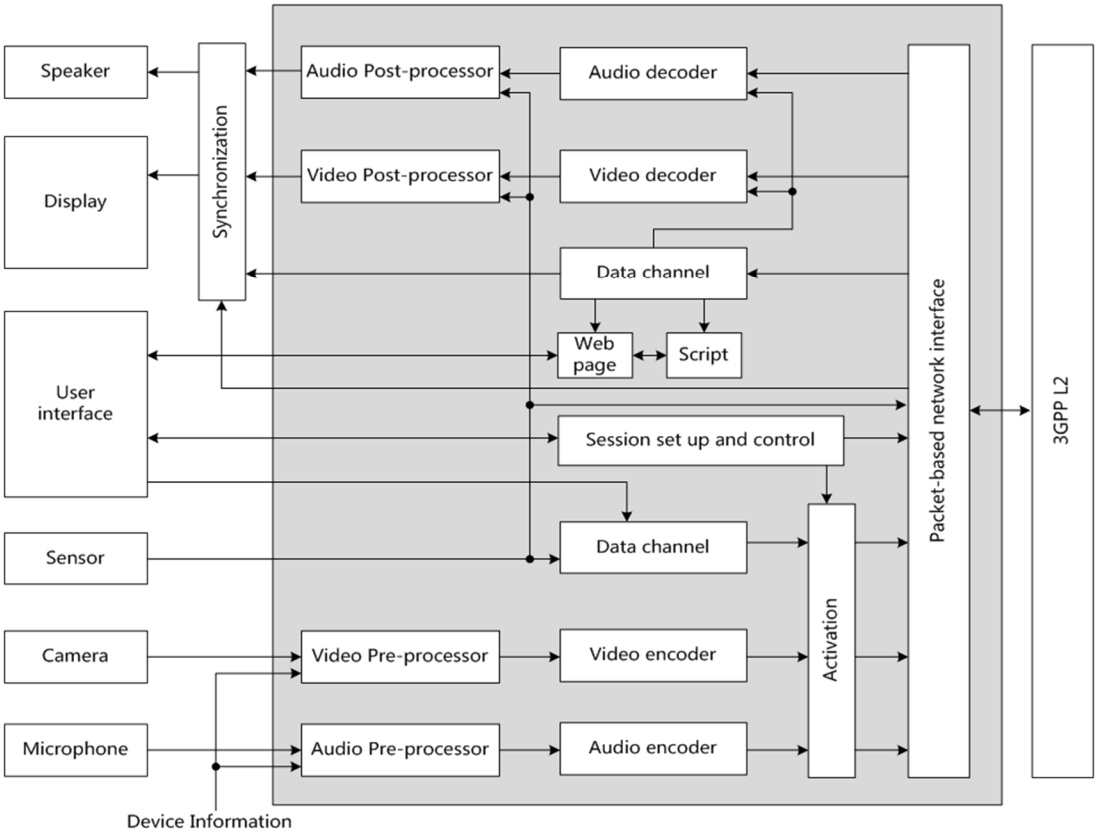


Figure 4.2.1: Functional components of an AR-MTSI client in terminal

4.3 End-to-End Reference Architecture

The end-to-end architecture to support AR communication over IMS can be found in TS 23.228 Annex AC [4]. The following Figure 4.3.1 is a simplified version showing the media functions within the scope of this specification.

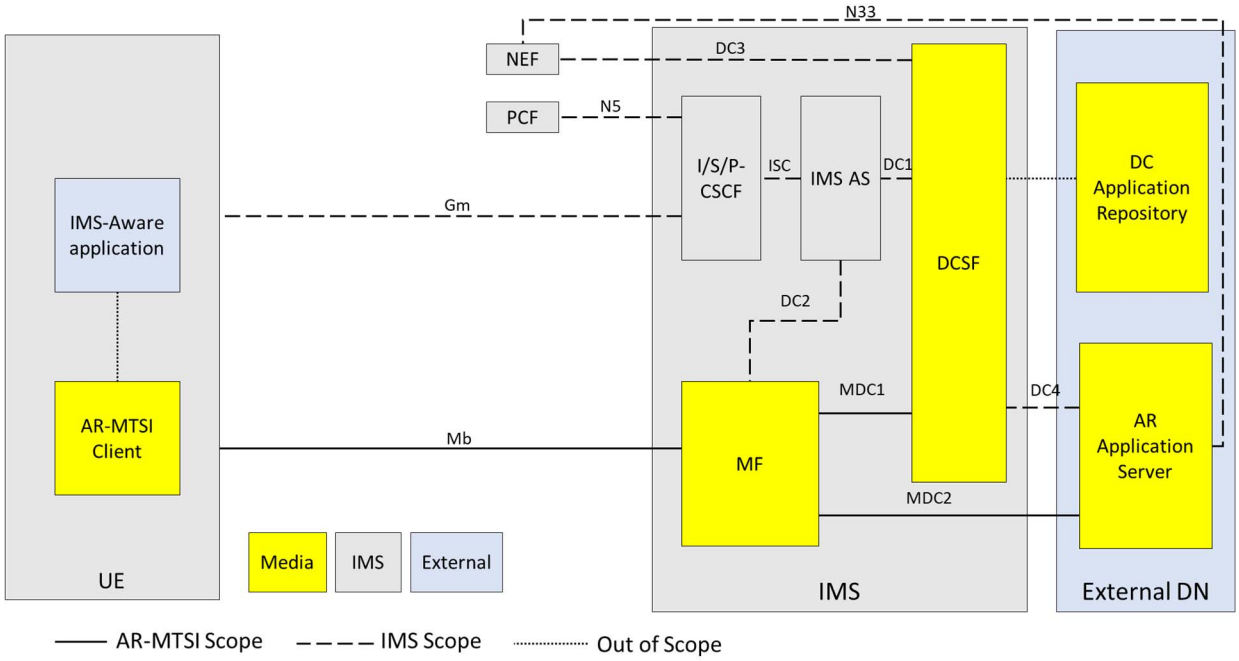


Figure 4.3.1: Generalized IMS DC Architecture to support AR communication

NOTE 1: General control-related elements over Gm interface, such as SIP signalling (TS 24.229 [5]), fall outside the scope of this specification, albeit parts of the session setup handling and session control for AR conversational media at Gm reference point, such as the usage of SDP and setup and control of the individual media streams between clients, are defined in this specification.

NOTE 2: DC Application Repository may be in external DN but can also be in operator domain. The DC Application Repository holds the application(s) that can be used in AR communication sessions and is out of scope of 3GPP.

AR Application Server (AR AS):

- AR Application Server is responsible for AR service control related to AR communication, including AR session media control and AR media capability negotiation with the UE.

NOTE 3: AR Application Server is a specific DC Application Server and is out of scope of 3GPP.

NOTE 4: The UE can download the AR metadata from AR AS through application data channel.

NOTE 5: Whether DC AS used in an avatar call is an AR AS is FFS.

DCSF:

- The DCSF receives event reports from the IMS AS, and decides whether AR communication service is allowed to be provided during the IMS session. Additionally, the DCSF interacts with the AR AS for DC resource control.

MF:

- Support AR conversational service by providing transcoding for terminals with limited capabilities. Additionally, the MF may collect spatial and media descriptions from UEs and create scene descriptions for symmetrical AR call experiences.

- Provide remote rendering for AR-MTSI clients in terminals with limited capabilities based on rendering negotiation. For remote rendering the AR-MTSI client provides AR metadata, e.g., pose data, as defined in clause 6 of this specification.

- For sessions that use avatar call, it optionally provides avatar animation functionality to the AR-MTSI clients with limited avatar animation capabilities.

IMS AS:

- The IMS AS receives the media control instructions from the DCSF and accordingly interacts with the UE for connecting the UE's audio/video media termination to the MF [4], and interacts with MF for data channel media resource management for AR media processing.

BAR:

- Manages and provides access to a user’s base avatar models to be used during an IMS session with avatar call.

5 Immersive AR Media

5.1 General

An AR-MTSI client supports simultaneous transfer of multiple media components with real-time characteristics. An AR-MTSI client supports the core media components in TS 26.114 [2] for a conversational AR scenario including text, image, video and speech (also referred to as audio).

5.2 Speech

AR-MTSI client in terminal offering speech communication shall follow clause 5.2.1 in TS 26.114 [2]. In order to support minimum service interoperability, an AR-MTSI client in terminal shall implement the UE codec and media handling requirements as specified in TS 26.114 [2].

5.3 Video

AR-MTSI client in terminal offering video communication shall follow clause 5.2.2 in TS 26.114 [2] and may render it as based on AR metadata (in clause 6) and media configuration (in clause 7). In order to support minimum service interoperability, an AR-MTSI client in terminal shall implement the UE codec and media handling requirements as specified in TS 26.114 [2].

Specifically, the AR-MTSI client in terminal may support Overlays and Scene Description-Based Overlays (as described in TS 26.114 [2] in clause Y.6.4 and Y.6.9) to render video elements in parts of the AR environment. This may result into rendering the video stream (or parts of the video stream) in a sub-area of the display device. Further, the UE may negotiate a stream characteristic most suitable for the sub-area and may renegotiate the stream characteristics in case the sub-area changes.

5.4 Real-time text

AR-MTSI client in terminal offering real-time text shall follow clause 5.2.3 in TS 26.114 [2] and may render it as defined in the AR metadata (in clause 6).

5.5 Still images

AR-MTSI client in terminal supporting still images shall follow clause 5.2.4 in TS 26.114 [2] and may render it as defined in the AR metadata (in clause 6).

5.6 Avatars

5.6.1 General

An AR-MTSI client in terminal supporting Avatar call services over IMS data channel shall support the Avatar Representation Format (ARF) as specified in [10] and shall have at least one base avatar stored in the Base Avatar Repository (BAR) in one of the two supported container formats defined in [10].

An AR-MTSI Rx client that supports avatars shall support the ARF container in both ISOBMFF and ZIP formats.

NOTE 1: The ARF specification is still under development. Some details may change.

The base avatar model shall comply with the ARF specification [10]. In addition, the ARF document as specified in clause 6 of [10] shall include the following information:

- A list of the supported animations, which includes at least one animation type (e.g., face or landmark animation),
- At least one asset with at least one level of detail.

All data of relevant assets shall be contained in the ARF container of the base avatar model. All data of relevant assets selected for an avatar representation shall be contained in the ARF container of that avatar representation.

NOTE 1: Evaluation of MPEG-ARF is FFS. Interoperability aspects of ARF have not been fully evaluated and are FFS.

NOTE 2: URNs for specific animation frameworks can be defined in operator-specific profiles or through industry fora.

5.6.2 3D Avatar Format

5.6.2.1 General

An AR-MTSI client that supports 3D avatars shall support the ARF base avatar format as specified in [10] with the requirements in clause 5.6.1.

5.6.2.2 3D Avatar simple profile

The 3D avatar container shall consist of the following mandatory components:

At least one Skeleton component as defined in [10] that defines the hierarchical joint structure for body animation, with support for at least partial humanoid joint configurations. Inverse Bind Matrices (IBMs) shall be provided for each joint of the skeleton.

At least one Skin component as defined in [10] that references both the skeleton and associated meshes to enable skeletal deformation.

3D mesh geometry data that conforms to the binary glTF (GLB) format version 2.0, with support for triangle-based topology.

Skinning weight data provided as dense tensors in the format specified in Annex E of [10] with a maximum of 4 joint influences per vertex.

Texture data components that conform to still image formats as defined in clause 5.5.

When facial animation is supported, at least one BlendshapeSet component as defined in [10] which should include:

A minimum of 50 blend shapes,

Shape key data as meshes in GLB format, restricted to vertex positions, polygon/face information, normals, and tangents.

Data items of the 3D avatar should signal no compression or protection schemes by default.

NOTE 1: Compression aspects of ARF are for FFS.

NOTE 2: Content protection aspects are for FFS

5.6.3 2D Avatar Format

5.6.3.1 General

An AR-MTSI client that supports 2D avatars shall support the ARF base avatar format as specified in [10] with the requirements specified in clause 5.6.1.

5.6.2.2 2D Avatar simple profile

The 2D avatar container shall consist of the following mandatory components:

At least one 2D mesh representation that consists of a single planar mesh or quad suitable for texture mapping, conforming to the binary glTF (GLB) format.

At least one static image asset for the base avatar representation that conforms to still image formats as defined in clause 5.5.

At least one LandmarkSet component as defined in [10] for facial animation.

Data items of the 2D avatar should signal no compression or protection schemes.

Support for animation using voice-based animation through a pre-trained and fine- tuned model for the user may be supported.

6 AR Metadata

6.1 General

Real-time scene creation for an AR conference with two or more participants may be done by the MF to create a symmetric experience for all participants. For an MF to create a scene, it may request the following information from the UEs:

- spatial description of the space surrounding the UE e.g., the occlusion-free space around the user in which the AR media will be rendered.
- media properties indicating the AR media that the UE will be sending, and thus have to be incorporated in the scene.
- receiving media capabilities of the UEs, which may include
  - UE media decoding capabilities
  - UE hardware capabilities (e.g., the display resolution)
- information based on detecting the location, orientation, and capabilities of physical world devices, eligible for usage in an audio-visual communications session

Based on this information the MF creates a scene which includes:

- defining the placement of the user and the AR media in that scene, including e.g., the position, size, depth from the user, anchor type, and recommended resolution (or quality)
- specific rendering properties for the AR media, e.g., for a 2D object to be rendered with a billboarding effect

The MF can then share the scene with the participant UEs using a supported scene description format. This scene description may be different for different UEs.

NOTE: The scene as sent by the MF allows the UE to 1) select and request any related media (for example, in a quality and bitrate based on the rendering characteristics or network connection), 2) render the complete scene on a (virtual) display device, and 3) update the rendering and requested media dynamically (e.g., according to the movement and view orientation of the user).

Avatar-specific metadata is defined in clause 6.3.2.

6.2 Metadata data channel message format

For the carriage of metadata defined in this clause the AR-MTSI clients shall use the data channel. The data channel sub-protocol shall be identified as “3gpp-ar-metadata”, which shall be included in the dcmap attribute of the SDP.

The transmission order for the data channel shall be set to in-order and the transmission reliability shall be set to reliable.

The metadata message format shall be set to text-based and the messages shall be UTF-8 encoded JSON messages.

A data channel message may carry one or more AR metadata messages as defined in Table 6.2-1.

Table 6.2-1 AR Metadata Messages Format

Name	Type	Cardinality	Description
messages	Array(Message)	1..n	A list of AR metadata messages. Each message shall be formatted according to the Message data type as defined in Table 6.2-2.

Each metadata message shall follow the format specified in Table 6.2-2.

Table 6.2-2 AR Metadata Message Data Type

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	A URN that identifies the message type.
payload	object	1..1	The message payload depends on the message type.
sendingAtTime	number	0..1	The wall clock time when the AR metadata message is transmitted. (clause 9.3.2.1 in TS 26.565 [6])

6.3 Spatial descriptions

6.3.1 Spatial description format

6.3.1.1 General

A spatial description format is used for defining the physical space around a UE or trackable in which virtual content can be inserted. This clause includes the supported formats and the method for exchanging the information between AR-MTSI clients.

6.3.1.2 Available visualization space

An AR-MTSI client in terminal may send available visualization space, user position and other trackable poses to AR MF for scene creation and update.

The available visualization space defines an occlusion-free space around the user for rendering the AR scene as a geometric primitive. The format for available visualization space is defined in clause 12.4 of TS 26.119 [3]. The type of the message containing visualization space as a payload shall be "**urn:3gpp:ar:v1:visualization-space**". The availableVisualizationSpace object [3] shall contain a *xrSpaceId*. The *xrSpaceId* is used for determining the local coordinate axis of the visualization space. The *xrSpaceId* shall be a unique identifier for an XR space of one AR-MTSI client in terminal. If the visualization space is sent, then initial user pose shall be sent. The user pose and visualization space are in reference to the same *xrSpaceId*.

If the visualization space is anchored to another trackable (instead of the user) not anchored around the user or if the viewer is not the centre of the visualization space, an initial pose for a trackable as defined in clause 6.3.1.3 may be used.

6.3.1.3 Initial Pose

Trackable is a real-world object (e.g., the UE, floor, controllers, table etc.) that the UE can detect, which can be used as a reference to anchor virtual objects to the real world.

The AR-MTSI client in terminal that sends the available visualization space may also send at least one pose for a trackable within the visualization space. The AR-MTSI client in terminal may send additional poses for anchoring virtual objects. The poses shall be sent using the format defined in clause 12.2 of TS 26.119 [3]. The poseInfo (as defined in Table 12.2-1 of TS 26.119 [3]) shall contain an *xrSpaceId* that is the same as the one used for visualization space. The poses may additionally contain a label string to identify the type of anchor. The labels are application-dependent, but for example, user, floor, left controller etc., can be used as labels. The type of the message for a pose sent for scene creation shall be set to "**urn:3gpp:ar:v1:initial-pose**".

6.3.2 Avatar Animation Stream Format

An AR-MTSI client or MF that supports avatar animation in Avatar call services over IMS data channel shall support the exchange of avatar animation data over the data channel according to the sample formats described in clause 8 of [10].

NOTE: Support for other means to transport animation streams (e.g., over the media channel) may be added in the future.

When the data channel is used to send animation data, the metadata data channel message format defined in clause 6.2 shall be used and the avatar animation messages shall have the type "urn:3gpp:avatar:v1:animation" and the format shown in Table 6.3-1.

Table 6.3-1: Message format for avatar animation messages

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp:avatar:v1:animation
message	Object	1..1	Message content
subtype	number	1..1	An identifier of the subtype of the animation message. Value 1 indicates a facial animation, value 2 indicates a joint animation, and value 3 indicates a landmark animation, 10 indicates audio, 11 indicates video. Other values are reserved for future use.
payload	Object	1..1	The avatar animation sample format corresponding to the message subtype.

No compression scheme is defined for animation samples.

An AR-MTSI client that offers an avatar animation stream should notify the remote client with a message over the data channel with the URN **urn:3gpp:avatar:v1:animation:stopped** when the animation stream becomes temporarily unavailable**.**

Format of the stopped message is shown in Table 6.3-2.

Table 6.3-2: Stopped message for avatar animation

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	A URN that identifies the message type: urn:3gpp:avatar:v1:animation:stopped
animationStream	URI	1..1	A URN that identifies the stopped animation stream
reason	string	0..1	An optional field indicating the cause of disruption in animation stream. Possible values include: "device error", "low-confidence for sensor data", "network issues", etc.
startTime	number	0..1	start time of the suspension of the animation data
endTime	number	0..1	end time of the suspension of the animation data if known
mid	number	0..1	MID value of the audio stream used as fallback when animation stream stopped.

When the animation stream becomes available again, the AR-MTSI client that offers the avatar animation stream should notify the remote client with a message over the data channel with the URN **urn:3gpp:avatar:v1:animation:resumed**.

Format of the resumed message is shown in table 6.3-3.

Table 6.3-3: Resumed message for avatar animation

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	A URN that identifies the message type: **urn:3gpp:avatar:v1:animation:resumed**
animationStream	URI	1..1	A URN that identifies the resumed animation stream.

6.4 Scene descriptions

6.4.1 General

An AR-MTSI client in terminal that is a compliant device type of TS 26.119 [3] shall support the capabilities requirements for scene description as described in clause 10 of TS 26.119 [3] for its respective device type.

When used in an AR call, the scene description should be the entry point to the AR session (after establishment of a regular call/conference) and shall be exchanged over the data channel as described in TS 26.114 [2]. The Scene Description is exchanged over a stream with a stream id in the range 1 to 1000 and shall be provided by the AR AS through the MF to the AR-MTSI client in terminal. In this case, no web application needs to be downloaded in this case.

Based on the information in the Scene Description, the UE may decide to add additional media streams through a re-INVITE. However, at least the RTP session for the voice of every participant should be present and should be linked to an audio source in the scene description.

NOTE: Support for advanced audio codecs, such as IVAS, in scene description is for further study.

Each participant should be associated with their own camera node, identified through the node name, which is also provided as part of the SDP through the “sd-nodes” attribute of media session of the data channel that carries the Scene Description.

The “sd-nodes” attribute shall conform to the following ABNF syntax:

```
att-field           = "sd-nodes"
att-value           = participant-label SP node-desc *("," node-desc)
participant-label   = char-val  ; char-val is defined in RFC 7405
node-desc           = node-name ["/avatar"]
node-name           = char-val  ; char-val is defined in RFC 7405
```

The AR MF should apply pose updates from the received pose information of each participant to their respective camera nodes, as negotiated by the SDP sd-nodes attribute.

A scene description of an AR session may be sent from the AR MF/MRF to the AR-MTSI clients in terminal.

An AR MF that supports scene description shall support:

- The capability to generate a scene description file that conforms to the SD-Rendering-glTF-Core capability as defined in TS 26.119 [3].
- The capability to generate and update a scene description file that conforms to the SD-Rendering-glTF-Ext1 as specified in TS 26.119 [3].

An AR MF that supports scene description may additionally support the generation of scene description files and updates that conform to the **SD-Rendering-glTF-Ext2** capabilities as defined in TS 26.119 [3].

An AR MF that supports scene description may additionally support the generation of scene description files and updates that conform to the **SD-Rendering-glTF-Interactive** capabilities as defined in TS 26.119 [3].

In addition, an AR MF that supports scene description shall support the referencing of RTP streams in the scene description through the MPEG_media extension as defined in ISO/IEC 23090-14 AMD 2 [7]. The external media shall be RTP media streams supported by an AR-MTSI client and signalled in the SDP.

When scene description is not used as the entry point, the scene description shall be sent by the AR MF/MRF to the AR-MTSI client in terminal over the application data channel. The type of the message containing the scene description shall be set to "**urn:3gpp:ar:v1:sd**".

An AR MF that supports scene descriptions should create and distribute the scene for an AR call with audio and video streams based on the visualization space, viewer position and AR media properties. The AR MF should create the scene description for each participant (AR-MTSI client in terminal) such that the shared experience is symmetrical for the different users in the call, e.g., to maintain relative position of users and objects.

6.4.2 Integration in Scene Description

An AR-MTSI client supporting avatar calls with scene description shall support the integration of MPEG-ARF avatars as defined in Annex B of [10]. When avatars are used in a call with scene description, the avatar representation shall be integrated using the MPEG_node_avatar extension as specified in [7] and extended in Annex B of [10].

The scene description containing avatar nodes shall be exchanged over the data channel as described in [2] clause 6.2.10, using a stream with a stream ID in the range 1 to 1000. Each participant's avatar shall be associated with their own avatar node in the scene, identified through the node name, which shall be provided as part of the SDP through the "sd-nodes" attribute of the media session of the data channel that carries the scene description, with clear marking of the node as an avatar node as defined in clause 6.4.

The MPEG_node_avatar extension shall include:

The avatar type set to the URN defined in clause 4.2 of ISO/IEC 23090-39

The ARFContainer URL pointing to the avatar in the BAR

Animation stream definitions for blendshapes, joints, landmarks, or multiplexed animation data

An MF that supports avatar scene description shall support:

The capability to reference ARF containers stored in the BAR through the MPEG_node_avatar extension

The capability to reference animation streams that are delivered over application data channels

The capability to synchronize avatar animations with other scene elements and RTP media streams

When scene description is used as the entry point for an avatar-enhanced call, the MF shall apply animation updates received from each participant to their respective avatar nodes, as negotiated by the SDP avatar-nodes attribute. The animation data shall be transmitted through the application data channel established for avatar animation.

6.5 Network media rendering

6.5.1 General

The AR-MTSI client in terminal supporting network media rendering (of AR Media objects or 3D scenes) shall support metadata formats for split rendering as specified in clause 8.3 of TS 26.565 [6].

NOTE: In case network media rendering is used, AR Media objects or 3D scenes are rendered in a split rendering server (AR AS) based on the XR pose information and actions of the AR-MTSI client in terminal. The split rendering server renders a view of the scene (or object) based on the viewing angles and XR pose, provided by the AR-MTSI client in terminal. The resulting images are sent as video streams from the split rendering server to the AR-MTSI client in terminal, which renders the videos at their respective position in AR.

6.5.2 Pose Format

When the network media rendering is activated, the AR-MTSI client in terminal periodically transmits a set of pose predictions to the AR AS. The pose prediction format shall conform to the payload of the message whose type is "**urn:3gpp:split-rendering:v1:pose**" as specified in clause 8.3.2.2 of TS 26.565 [6].

6.5.3 Action Format

The action sets and actions are negotiated during the AR media rendering negotiation. The AR-MTSI client in terminal reports any changes to action state as it occurs by sending updated actions to the AR AS after the network media rendering is activated. When the AR-MTSI client in terminal sends updated actions to the AR AS, the action format shall conform to the payload of the message whose type is "**urn:3gpp:split-rendering:v1:action**" as specified in clause 8.3.2.3 of TS 26.565 [6].

7 Media configurations

7.1 General

The media configuration requirements for MTSI clients in terminals specified in TS 26.114 [2], clause 6, also apply for AR-MTSI client in terminal.

An SDP framework for AR data exchange for AR communication is presented to negotiate codec support for AR media, AR metadata, as well as RTP/RTCP signalling necessary for AR media rendering processing.

AR-MTSI client in terminal shall use RTP for the real-time transport of AR media for AR communication. Any AR media as an overlay may refer to the overlay configuration described in clause Y.6.4.3 of TS 26.114 [2].

AR-MTSI client in terminal shall use data channels for exchange of AR metadata and rendering negotiation. The SDP attribute "*3gpp_armetadata_types*" should be used to indicate the types of AR metadata which defined in clause 6 (e.g. pose, action and scene description) within the data channel.

The syntax for the SDP attribute is:

```
att-field       = "3gpp_armetadata_types"
att-value       = message_type *("," message_type)
message_type    = char-val
                    ; URN identifying the message type of AR metadata
                    ; char-val is defined in RFC 7405
```

Poses as part of AR metadata may be transmitted via RTP session in a RTP header extension as specified in clause 4.3 of TS 26.522 [8].

The data channel requirements for an MTSI client in terminal specified in TS 26.114 [2], clause 6.2, also apply for AR-MTSI client in terminal that supports Avatar calls.

AR-MTSI client in terminal that supports Avatar calls shall use application data channel for transport of animation data.

7.2 Network media rendering configuration

The AR-MTSI client in terminal shall indicate its support for AR calls by including the “webrtc-datachannel” in the “+sip.sub-type” Contact header field.

A new Contact header field parameter, “+sip.3gpp-ar-support” is used to indicate the level of support for AR calls. The possible values for the “3gpp-ar-support” parameter are:

- “**ar-capable**”: indicates that the terminal is fully capable of receiving and rendering AR media as described by the capabilities in [2] clause 9.2.

- “**ar-assisted**”: indicates that the terminal is capable of transmitting AR metadata on the uplink. However, the UE has no support for processing and rendering a 3D scene. The participation in an AR call requires the deployment of network rendering. The rendered view(s) are controlled by the pose information that is shared by the terminal.

In the absence of the “+sip.3gpp-ar-support”, it shall be assumed that the terminal has no support for AR calls. In this case, the MTSI client can only participate in the AR call if network rendering is offered.

An AR-MTSI terminal that intends to participate in an AR call shall register with the “**ar-capable**” value for the “+sip.3gpp-ar-support” parameter and shall offer/answer an SDP that includes a data channel with the sub-protocol “mpeg-sd”. Any updates that the AR-MTSI terminal intends to share, including pose updates, will be sent as scene updates to the AR AS. An AR-MTSI terminal that intends to participate in an AR call with the support for network rendering shall register with the “**ar-assisted**” value for the “+sip.3gpp-ar-support” parameter and shall offer/answer an SDP that includes a data channel with the sub-protocol “3gpp-sr-metadata” as defined in [6]. Pose updates that are to be used for the rendering are shared as pose predictions with the MF.

As specified in Annex AC.9 of TS 23.228 [4], the AR application server may provide network assisted rendering. An AR-MTSI client in terminal can decide to request network media rendering based on user selection and its status such as power, signal, computing power, internal storage, etc. The AR-MTSI client in terminal shall complete an AR media rendering negotiation with the AR AS before it initiates subsequent procedures to activate the network media rendering. The data channel should be established for rendering negotiation with SDP offer/answer between AR-MTSI client in terminal and MF with the sub-protocol “3gpp-sr-conf”, and continue to be used for rendering re-negotiation until the end of the AR communication.

An **AR-assisted** terminal that intends to deploy network rendering for AR media rendering, shall use the negotiation processes between the AR-MTSI client in terminal and the AR AS to determine the split rendering configuration. The split rendering configuration shall be in JSON format as specified in clause 8.4.2 of TS 26.565 [6]. The exchange of the configuration information shall take place using the established application data channel. The split rendering configuration message shall be formatted according to clause 8.4.2.2 of TS 26.565 [6] and shall have the type: “**urn:3gpp:split-rendering:v1:configuration**”. The output description message shall be formatted according to clause C.1.4 of TS 26.565 [6] and shall have the type: “**urn:3gpp:split-rendering:v1:output**”.

For a terminal that does not support AR calls, the IMS AS may trigger network rendering on behalf of the terminal upon receiving an (re)INVITE for an AR call. The output format for the rendered media shall be conformant to clauses 10.4.3 and 10.4.4 of TS 26.119 [3]. The MF that performs the remote rendering shall select a suitable rendering viewpoint for the session, e.g. a selected viewpoint in the scene or the initial viewpoint for the participant as assigned by the AR AS in the scene description. In case no network rendering can be setup, the IMS AS should reject the call.

The IMS AS detects support for AR capabilities based on the presence or absence of the “+sip.3gpp-ar-support” parameter of the Contact Header Field in the REGISTER message.

7.3 Avatar animation and rendering configuration

7.3.1 Avatar capability configuration

The AR-MTSI client in terminal shall indicate its support for avatar calls by including the “webrtc-datachannel” in the “+sip.sub-type” Contact header field.

A new Contact header field parameter, “+sip.3gpp-avatar-support” is used to indicate the level of support for Avatar calls. The possible values for the “3gpp-avatar-support” parameter are:

- “**avatar-capable**”: indicates that the terminal is fully capable of receiving, animating and rendering of avatar.

- "**avatar-assisted**": indicates that the UE has no support for animating or rendering an avatar

NOTE: The SIP register message for avatar calls is FFS. It may be needed for indicating which UEs require avatar assistance and which ones are avatar capable.

7.3.2 Network Animation and Rendering

When an AR-MTSI client initiates or receives a call with avatar media, the IMS Application Server (IMS AS) shall evaluate whether network-based avatar animation and rendering is required. The IMS AS forwards avatar-related INVITE requests to the Avatar-capable AR AS for capability assessment and media routing decisions.

The AR AS shall invoke network-based animation and rendering through an MF when:

the receiving MTSI client has not registered the "+sip.3gpp.avatar-support" feature tag,

the receiving MTSI client has registered the "+sip.3gpp.avatar-support" feature tag with "avatar-assisted" value,

the receiving MTSI client's registered capabilities indicate insufficient resources for avatar animation,

the offered animation frameworks are not supported by the receiving MTSI client.

When network animation and rendering is invoked, the sending AR-MTSI client shall use the negotiation processes with the AR AS to determine the avatar animation and rendering configuration. The exchange of the configuration information shall take place using an established application data channel. The application data channel established for animation and rendering negotiation of the avatar with SDP offer/answer between the sending AR-MTSI client in terminal and the MF with the sub-protocol "3gpp-avatar-negotiation" and continue to be used for animation and rendering re-negotiation until the end of the avatar call. In this case, the MF shall establish a video stream with the receiving MTSI client using standard video codecs as specified in [2].

When network animation and rendering is invoked, the AR AS shall allocate an MF capable of real-time avatar rendering and configure it with the appropriate rendering parameters based on the receiving UE's video capabilities. The IMS AS shall modify the SDP to route avatar animation data to the MF instead of the receiving UE, effectively inserting the MF into the media path between the sending and receiving MTSI clients.

The MF performing network-based avatar animation and rendering shall fetch the ARF container from the BAR using the reference provided in the SDP and load the avatar model for rendering. The MF shall receive animation streams through the data channel from the sending AR-MTSI client and apply these animation samples to the avatar model in real-time. The animated avatar shall be rendered as a 2D video stream or stereoscopic 3D video stream based on the receiving UE's capabilities.

For the receiving MTSI client, the avatar data channel media description shall be replaced with a video media description including standard video codec negotiation as specified in [2], and avatar-specific attributes that are not applicable to video streams shall be removed.

The MF should ensure lip-sync between avatar animation and associated audio streams.

7.4 SDP Negotiation and Signaling of Avatars

7.4.1 General

The SDP is used for establishing the Avatar negotiation application data channel (ADC) between an AR-MTSI client and its remote peer (MF or AR-MTSI client).

7.4.2 SDP signalling

7.4.2.1 Establishment of ADC for avatar negotiation

For avatar negotiation over an established IMS application data channel, the data channel sub-protocol shall be identified as "3gpp-avatar-negotiation".

The data channel shall be ordered and reliable.

7.4.2.2 Negotiation of fallback mechanism

An AR-MTSI client receiving an SDP offer for an avatar call shall:

1. Examine the avatar capabilities in the offer, and
2. If avatar rendering is supported, the AR-MTSI client shall:

a. Fetch the avatar model using the provided reference from the BAR via the MF

b. Verify compatibility with at least one offered animation framework

c. Accept the data channel for animation parameters

3. If avatar rendering is not supported, the following process is used:

a. Network-based rendering may be invoked as specified in 7.3.2, or

b. Sender-centric rendering may be invoked, where the sending UE proceed with the rendering and the receiving UE receives a video stream representing the animated avatar, or

c. A fallback mechanism to regular non-avatar video call may be invoked with necessary media renegotiation, or

d. A fallback to audio only call may take place, or

e. If none of the above is supported, the session may be rejected.

When avatar rendering is not supported at the receiving UE and network-based rendering is invoked by the IMS network, the IMS AS shall renegotiate the connection by sending a modified SDP offer by including a new “video” media “m=” line to the receiving MTSI client.

When avatar rendering is not supported at the receiving UE and network-based rendering is not invoked by the IMS network, then the IMS AS shall delete the application data channel media description for avatar animation data from the SDP answer. The SDP answer shall not include a media description (“m=” line) for the avatar animation negotiation data channel. Upon receiving this SDP answer, the sending MTSI client may invoke sender-centric rendering.

When avatar rendering is not supported at the receiving UE and sender-centric rendering is invoked, the sending MTSI client shall renegotiate the connection by sending a modified SDP offer by including a new “video” media “m=” line to the receiving MTSI client. The new “video” media “m=” line shall include “a=avatar” attribute line to indicate that this video stream is generated by animating the avatar model.

The following cases can happen:

1. When avatar rendering is not supported at the receiving UE and sender-centric rendering is invoked, the receiving entity may accept the “video” media “m=” line with the associated attribute lines, indicating that the receiving UE accepts to receive a video of the animated avatar.
2. The receiving entity may reject the avatar video by deleting the “a=avatar” attribute line from the SDP offer and sending the modified SDP answer to the sending UE, indicating that it wishes to receive a regular video call.
3. Alternatively, the receiving UE may also remove the “video” media “m=” line from the SDP answer. This may indicate that the receiver wishes to have an audio only call or
4. The receiving UE may also remove all media “m=” lines from the SDP answer to reject the communication.

In cases 2 and 3 , the sending UE may renegotiate the call with a regular video call by adding a new “video” media “m=” line without including the “a=avatar” attribute line.

An SDP offer may contain multiple “m=” lines, each corresponding to one of the above cases 1 and 2. The receiver may accept either the “video” media “m=” line that contains the “a=avatar” attribute line or the “video” media “m=” line that does not contain the “a=avatar” attribute line in the SDP answer.

When an avatar video call is negotiated between a sending UE and a receiving UE and the sending UE or the receiving UE (whoever acts as an invoker) decides to send only regular video at a later point in time in the Avatar call session, then the invoker shall renegotiate the session by sending a modified SDP offer that includes a new “video” media “m=”

line and it shall not contain the “a=avatar” attribute line. The invoker shall remove the old “video” media “m=” line with “a=avatar” attribute line from the modified SDP offer.

When a regular video call is negotiated between a sending UE and a receiving UE and the sending UE or the receiving UE (whoever acts as an invoker) decides to switch to an avatar video call at a later point in time in the Avatar call session, then the invoker shall renegotiate the session by sending a modified SDP offer that includes a new “video” media “m=” line and it shall contain the “a=avatar” attribute line in the corresponding “video” media “m=” line. In this case, the invoker shall remove the old “video” media “m=” line from the modified SDP offer. The other UE may accept the modified SDP offer or may reject it.

7.4.2.3 Usage of the label attribute and label parameter

AR-MTSI clients supporting avatar calls shall use the "label" attribute by including it under the corresponding media streams (for AV media), or the “label” parameter by including it under the DCMAP attribute for corresponding data channel m-lines, in order to describe the media data to be delivered. These labels and their values should be the same in both the SDP offer and answer, and are used for descriptive purposes during media re-negotiation after avatar negotiation.

The label values defined for avatar calls are as follows:

a=label:3gpp-avatar-proc-{process}

as a label attribute for RTP media streams:

a=dcmap:0 label="3gpp-avatar-proc-{process}"

as a label parameter under the DCMAP attribute for data channel streams.

- **process**: the name of the process in the avatar workflow (see clause 7.1 of TR 26.813 [9]) which outputs the data to be delivered using the media stream or data channel under which this label attribute or parameter is included. Defined values (avatar processes) include:
  - "cp": the data transported is the output of the capture posture process which captures the user’s posture for creation of an animation stream; this output is typically either captured video, sensor data, or both.
  - "gc": the data transported is the output of the generate command process which generates animation stream data; this output is typically blendshape data or joint data.
  - "aa": the data transported is the output of the animateAvatar process which animates a 3D base avatar using animation stream data; this output is typically a file or patch containing the animated 3D base avatar data.
  - "ms": the data transported is the output of the manageScene process which either creates or updates the scene description for an avatar call; this output is typically a glTF file in the form of JSON or a JSON PATCH.
  - "rs": the data transported is the output of the renderScene process which renders the avatar (or scene description including any avatars); this output is typically 2D video.

Avatar workflow processes:

- Capture posture: the process of capturing the data representing the user’s posture which is used as input data to generate animation stream data.
- Generate command: the process of generating animation stream data.
- Animate avatar: the process of animating a 3D base avatar using animation stream data.
- Manage scene: the process of creating or updating a scene description for an avatar call.
- Render scene: the process of rendering the animated avatar.

If avatar animation capability negotiation has already been completed (animation mode decided), the AR-MTSI client supporting avatar calls shall include the label attribute or label parameter with the suitable value under each corresponding m-line of the SDP offer/answer, according to the required RTP stream(s) and ADC(s), respectively, which need to be established as defined in Figure A.2.4-1. E.g. if network rendering mode has been decided, the AR-

MTSI client shall include the label attribute with value "a=label:3gpp-avatar-proc-rs" under the video m-line media stream intended to transport the rendered avatar to the receiver client.

7.4.3 Avatar negotiation over ADC

7.4.3.1 Avatar negotiation protocol

Messages sent over the ADC for avatar negotiation shall use the AR metadata message format defined in clause 6.2.

The message type for avatar negotiation messages shall be set to "urn:3gpp:avatar:v1:negotiation". The payload of the messages shall be as defined in Table 7.4-1.

Table 7.4-1: Payload format of the avatar negotiation message type

Name	Type	Cardinality	Description
subtype	number	1..1	An identifier of the subtype of the message payload. The allowed values are listed in Table 7.4-2.
content	object	1..1	A URN that identifies the message type.

Table 7.4-2 defines all supported sub-types of the avatar negotiation message type:

Table 7.4-2: Subtypes of avatar negotiation messages

Value	Message Name
1	REQUEST
2	ACCEPT
3	REJECT
4	ACK

The REQUEST is sent by the UE or by the MF when acting as the negotiation coordinator.

The ACCEPT and REJECT messages are sent by the UE or MF to indicate that the received REQUEST is accepted or rejected.

The ACK is issued by the DC-AS or MF when it decides to perform animation and rendering on behalf of the receiver.

7.4.3.2 Message Formats

The following tables describe the message formats exchanged for avatar negotiation.

The REQUEST message payload is defined in Table 7.4-3.

Table 7.4-3: REQUEST payload

Name	Type	Cardinality	Description
avatarProfile	string	1..1	Indicates the avatar profile as defined in clause 5.6. NOTE: allowed values for the avatarProfile are FFS.
arfContainer	object	1..1	Reference to ARF container of the Avatar representation that is offered for the call. The container shall be retrievable from the BAR.
animationSourceData	array(URI)	1..1	Provides a list of supported animation data for the avatar. The following values are allowed: A subset of the AnimationFrameworks as declared in the ARF container of the base avatar Audio with the URN "urn:3gpp:avatar:v1:animation:audio" Video with the URN "urn:3gpp:avatar:v1:animation:video" At least one animation source data shall be included.
renderingLocation	array(string)	1..1	Indicates the possible location(s) for animating and rendering of the offered avatar. The possible values are: “sender”, “receiver” or “mf”. An MF may add “mf” in the offered list if it is capable of animating and rendering the offered avatar.
animationFallback	array(URI)	0..1	A list of URNs indicating the supported animation data to be used as a fallback for animating the avatar when the selected animation stream stops. If the animationFallback is an RTP stream, the MID value of the fallback stream shall be sent as part of the urn:3gpp:avatar:v1:animation:stopped message over ADC.

The ACCEPT message payload is defined in Table 7.4-4

Table 7.4-4: ACCEPT payload

Name	Type	Cardinality	Description
renderingLocation	string	1..1	Indicates the selected location for the rendering and animation of the offered avatar. If the location is set to “mf”, it has to be confirmed by the transmission of an ACK message from the MF to the receiver.
animationSourceData	array(URI)	1..1	Includes the selected subset of the animation frameworks that the receiver prefers to receive for performing the avatar animation. If an audio or a video source is used, the receiver or MF shall use one of the audio or video stream from the IMS session to animate the avatar.
animationFallback	URI	0..1	Selected URN from the values offered in the animationFallback field of the REQUEST message to be used as a fallback for animating the avatar when the selected animation stream stops. If the animationFallback is an RTP stream, the MID value of the fallback stream shall be sent as part of the **urn:3gpp:avatar:v1:animation:stopped** message over ADC.

If a fallback stream is indicated in the animationFallback field of the ACCEPT message and the animation fallback is an RTP stream, then the SDP media description of the fallback RTP stream shall include a MID value.

The ACK message payload is defined in Table 7.4-5.

Table 7.4-5: ACK payload

Name	Type	Cardinality	Description
selectedRenderingLocation	string	1..1	Indicates the selected rendering location as a final decision.

The REJECT message payload is defined in Table 7.4-6.

Table 7.4-6: REJECT payload

Name	Type	Cardinality	Description
code	string	1..1	Machine-readable error code. NOTE 1: If the REJECT message needs to be associated with a particular REQUEST is FFS. NOTE 2: Definition of Error codes is FFS.
detail	string	0..1	Human-readable diagnostic text.

The arfContainer typer is defined in Table 7.4-7.

Table 7.4-7: Component: arfContainer

Name	Type	Cardinality	Description
uri	string	1..1	Authorization-bound reference to the base avatar in BAR or an operator repository. Typical schemes: bar://… or https://….
auth	object	0..1	Ephemeral authorization for dereferencing URI (e.g., bearer token); present when required by BAR policy.

The auth object of the arfContainer is defined in Table 7.4-8.

Table 7.4-8: Sub-component: arfContainer.auth

Name	Type	Cardinality	Description
scheme	string	1..1	Authentication scheme; "bearer" is supported. NOTE: Other HTTP defined authentication schemes can be added and is FFS.
token	string	1..1	Opaque credential used to access and download the ARF container.
expiresAt	number	0..1	Expiry time in milliseconds since epoch for the token.

8 AR Data Transport

8.1 General

The data transport requirements for MTSI clients in terminals specified in TS 26.114 [2], clause 7, also apply for AR-MTSI clients in terminals.

8.2 RTP transport

Additionally to the requirements specified in TS 26.114 [2], clause 7, the RTP Header Extension for PDU Set Marking (clause 4.2) and RTP Header Extension for XR Pose (clause 4.3) specified in TS 26.522 [8] also apply for AR-MTSI clients in terminals.

8.3 RTCP usage

Additionally to the requirements specified in TS 26.114 [2], clause 7, the Transmission of timing information data for QoE measurements specified in TS 26.522 [8], clause 5.2, also applies for AR-MTSI clients in terminals.

8.4 Data Channel Transport of Avatar Data

8.4.1 General

An AR-MTSI client that supports avatars shall use the avatar ADC for the transport of avatar control and animation data.

The SDP attribute *a=3gpp_armetadata_types* shall enumerate all avatar-related message URNs that the endpoint is prepared to send or receive on that ADC.

8.4.2 Transport of Base Avatar

When a base avatar or selected assets are delivered in-session over a data channel, the sender and receiver shall establish a second application data channel whose sub-protocol is "http".

The http channel shall be reliable and in-order.

8.4.3 Transport of Animation Streams

The format of the avatar animation streams is defined in clause 6.3.2.

9 Quality of Experience

9.1 General

Quality of Experience (QoE) requirements for MTSI clients in terminals specified in TS 26.114 also apply for terminals to be specified by this specification. Further, extensions to those QoE requirements are for future studies (and expected once extensions are made to the AR media formats).

9.2 Avatar-related QoE

9.2.1 Timing Information for Avatar Animation and Rendering

In avatar calls, compared to traditional remote or split rendering for AR, one important parameter to estimate the user quality of experience is the *posture to render to photon time delay*, defined as the time duration between the *posture-capture-time* or *animation-data-generation-time* and the *actual-display-time*. The calculation or measurement of the timestamps related to this delay is dependent on the entity performing the avatar animation and rendering, which is also influenced by the latencies involved in generating and delivering the animation data required at the corresponding entity.

The *posture-capture-time* (T10) is measured in the sending UE, as the time when the sender UE user’s pose is captured in order to generate the animation data.

The *animation-data-generation-time* (T11) is measured either in the sending UE or the MF, as the time when the animation data is generated from the source data.

The *avatar-animation-time* (T12) is measured, depending on the animation mode, either in the sending UE, MF or receiving UE, as the time when the base avatar is animated and rendered.

The *actual-display-time* (T2) is measured in the receiving UE, as the time when the rendered avatar is displayed to the user.

These timestamps may be delivered to each corresponding entity via feedback messages in order to facilitate better quality of experience. Better QoE may be provided to the user either through an adjustment in pose correction (in the

receiving UE), or by other means such as the re-negotiation of a more suitable entity for animation data generation and/or avatar animation and rendering.

9.2.2 RTCP message-based transmission of avatar timing information

9.2.2.1 General

The avatar timing information data recorded at the MF or at the RTP sender (UE) can be transmitted to another entity by enhancing the RTCP XR packets, which are specified in IETF RFC 3611 [16]. The RTCP XR report is identified by payload type (PT) equal to 207, which refers to an extended report block message. For transmission of timing information data using RTCP XR messages, the block type (BT) defined in RFC 3611 [16] can be extended with a value TBD.

NOTE: The block type value for the QoE timing information RTCP XR message will be assigned by IANA and the specification will be updated with that block type value later.

9.2.2.2 Extended report block for avatar QoE timing information

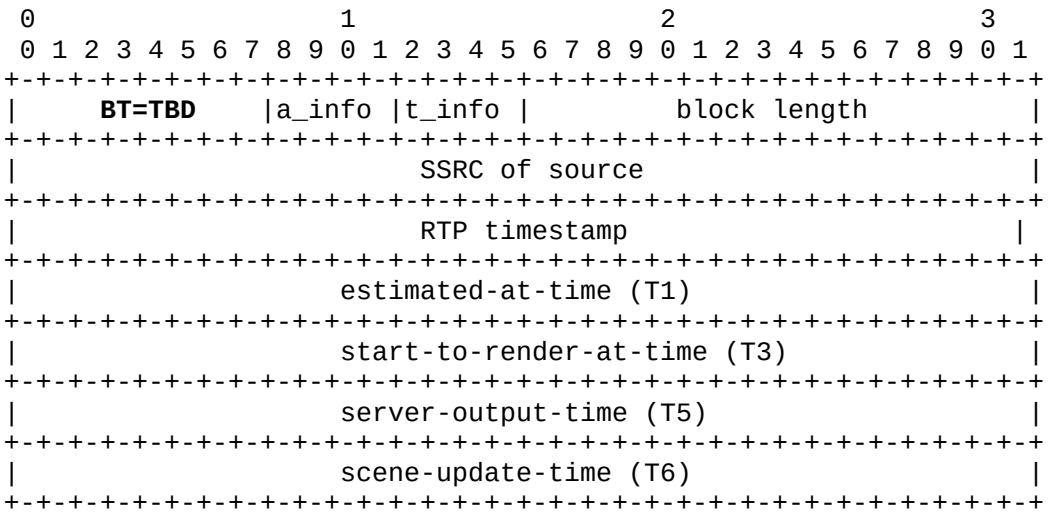
This extended report block type permits detailed reporting of avatar timing information recorded at the MF or UE. These reports can be used, for example, for calculating the QoE metrics such as the *posture to render to photon time delay* at the UE.

The avatar timing information required for measuring QoE metrics may be expressed in the same units as in the RTP timestamps of RTP data packets. This is so that, for each packet, one can establish the relation between the media data flowing and the corresponding avatar QoE timing information recorded at the MF or UE for a specific avatar media frame.

For optimum use of the RTCP bandwidth, the RTCP XR block payload may contain the whole or part of the avatar timing information required to calculate the QoE metrics. The `a_info` field present in figure 9.2.2.2-1 represents the avatar timing information in an RTCP XR block report. The `t_info` field represents respectively the timing information as defined in clause 5.2.2.2 of TS 26.522 [8]. When a bit is set to 'ONE' in the `a_info` field the respective avatar time information shall be present in the payload. When a bit is set to 'ZERO' in the `a_info` field, the respective avatar time information shall not be present in the payload. When an XR block report contains both `a_info` and `t_info` related timing information, the timing information present shall follow the order T1, T3, T5, T6, T10, T11, T12 followed by T2.

The identifiers of all actions that were processed for the rendering of a frame at a specific time are reported in the RTP HE for XR pose defined in clause 4.3 of TS 26.522 [8]. The synchronization between the various avatar timing information present in the below XR report and other timing information as defined in clause 5.2.2.2 of TS 26.522 [8] and any other action identifiers present is performed using the RTP timestamp information present in the RTP header of relevant packets and the `RTP timestamp` field present in the below RTCP XR report block.

The avatar-related QoE timing information Report Block has the following format:



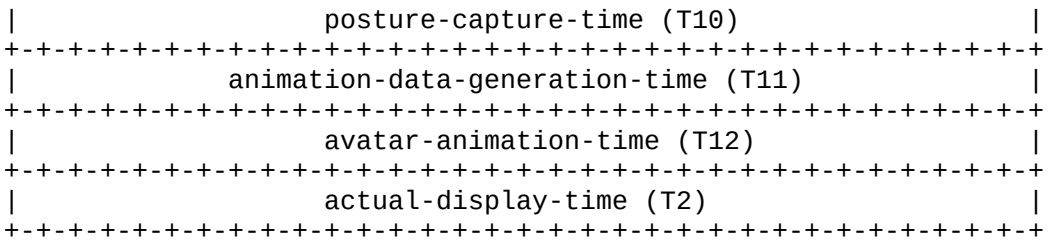


Figure 9.2.2.2-1: RTCP XR block format for avatar-related QoE timing information data

The semantics of the fields in avatar-related QoE time information Extended Report (RTCP XR) block are as follows:

- `block type (BT)` [8 bits]: A QoE time information Report Block is identified by a constant value.
- `block length` [16 bits]: The length of this report block, including the header, in 32-bit words minus one.
- `resv` [3 bits]: This field is reserved for future definition. In the absence of such definition, the bits in this field shall be set to zero by the sender and shall be ignored by the receiver.
- `t_info` [4 bits]: This field bits represent the timestamps that are present in the RTCP XR report block. When T1 is present in the RTCP XR report, first bit (least significant bit) is set to 1. When the LSB is set to 0, T1 information shall not be present. When T3, T5 and T6 are present in the RTCP XR block data, bits 2, 3 and 4 are set to 1 respectively. When T1, T3, T5 and T6 are present in an RTCP XR block data, the `t_info` field value shall be b1111. The timing information when present shall follow the order T1, T3, T5 followed by T6. For example, when the `t_info` field value is b0101, the RTCP XR block carries the T1 information followed by T5. T3 and T6 timing information will not be present in that RTCP XR block content.
- `a_info` [4 bits]: This field bits represent the avatar related timestamps that are present in the RTCP XR report block. When T10 is present in the RTCP XR report, first bit (least significant bit) is set to 1. When the LSB is set to 0, T10 information shall not be present. When T11, T12 and T2 are present in the RTCP XR block data, bits 2, 3 and 4 are set to 1 respectively. When T10, T11, T12 and T2 are present in an RTCP XR block data, the `a_info` field value shall be b1111. The avatar timing information when present shall follow the order T10, T11, T12 followed by T2. For example, when the `a_info` field value is b0101, the RTCP XR block carries the T10 information followed by T12. T11 and T2 avatar timing information will not be present in that RTCP XR block content. When both t_info and a_info indicate the presence of certain timestamp information, the timing information when present shall follow the order T1, T3, T5, T6, T10, T11, T12 followed by T2.
- The transmission frequency of T10, T11, T12 and T2 time information in RTCP XR report block can be negotiated during the configuration phase.
- `SSRC of source` [32 bits]: The SSRC of the RTP data packet source being reported upon by this report block.
- `RTP timestamp` [32 bits]: This field represents the RTP timestamp of the media frame at which the corresponding avatar QoE timing information date was recorded. This correspondence may be used for synchronization between the media data and the avatar QoE timing information measurements recorded at the MF or corresponding UE for a specific media frame.

  `estimated-at-time (T1)` [32 bits]: This field represents the time when the pose estimation was made. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.
- `start-to-render-at-time (T3)` [32 bits]: This field represents the time when the renderer in the split rendering server started to render the associated media frame. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.
- `server-output-time (T5)` [32 bits]: This field represents the recorded time at the output of the split rendering server. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.
- `scene-update-time (T6)` [32 bits]: This field represents the time when the Scene manager processes the interaction task according to the actions in the action message from the UE and updates the scene. This time

information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.

- `posture-capture-time (T10)` [32 bits]: This field represents the time when the sender UE user's pose is captured in order to generate the animation data. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.

- `animation-data-generation-time (T11)` [32 bits]: This field represents the time when the animation data is generated from the source data. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.

- `avatar-animation-time (T12)` [32 bits]: This field represents the time when the base avatar is animated and rendered, either in the sending UE, MF, or receiving UE, depending on the animation mode. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.

- `actual-display-time (T2)` [32 bits]: This field represents the time when the rendered avatar is displayed to the user in the receiving UE. This time information is expressed in the same units and with the same random offset as the RTP timestamps in data packets.

NOTE: How the time information specified can be measured is FFS.

9.2.3 SDP signaling and attributes

RFC 3611 [16] defines the SDP attribute "`rtcp-xr`" that can be used to signal to participants in a media session that they should use the specified RTCP XR blocks. The parameter `avatar-qoe-metrics` is defined under the "`rtcp-xr`" attribute to indicate the sending of avatar related QoE RTCP XR blocks as defined in Figure 9.2.2.2-1.

Annex A (normative): Call flows for IBACS

A.1 IMS AR communication Call Flows

A.1.1 General

Figure A.1.1-1 illustrates a high-level call flow for AR communication.

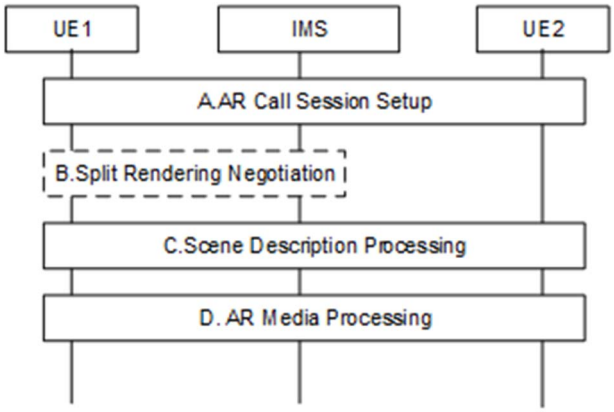


Figure A.1.1-1: High-level call flow for AR communication

The following steps are performed:

A. AR Call Session Setup: UE1 initiates an AR call, and the AR call session is established between UE1 and UE2, data channels along with audio and video channels are established between UE and network

B. Split Rendering Negotiation: When the UE has poor rendering capability which is not able to satisfy the requirements of AR communication, the split rendering negotiation is involved between the UE and IMS. The split may be adjustable during a session. This split rendering negotiation step can be executed/re-executed during the session when the UE's status changed and/or user selection.

C. Scene Description Processing: Prepare and generate scene description updates for the AR call session, this procedure can be done either on UE or IMS network.

D. AR Media Processing: AR media & Metadata transmission and AR media rendering for the AR call session, if split rendering is enabled, this procedure can be done by both UE and IMS network.

A.1.2 AR Call Session Setup

The AR call session procedure shall be followed as specified in clause AC.7 of TS 23.228 [4].

A.1.3 Split Rendering Negotiation

Figure A.1.3-1 illustrates a detailed call flow for split rendering procedure.

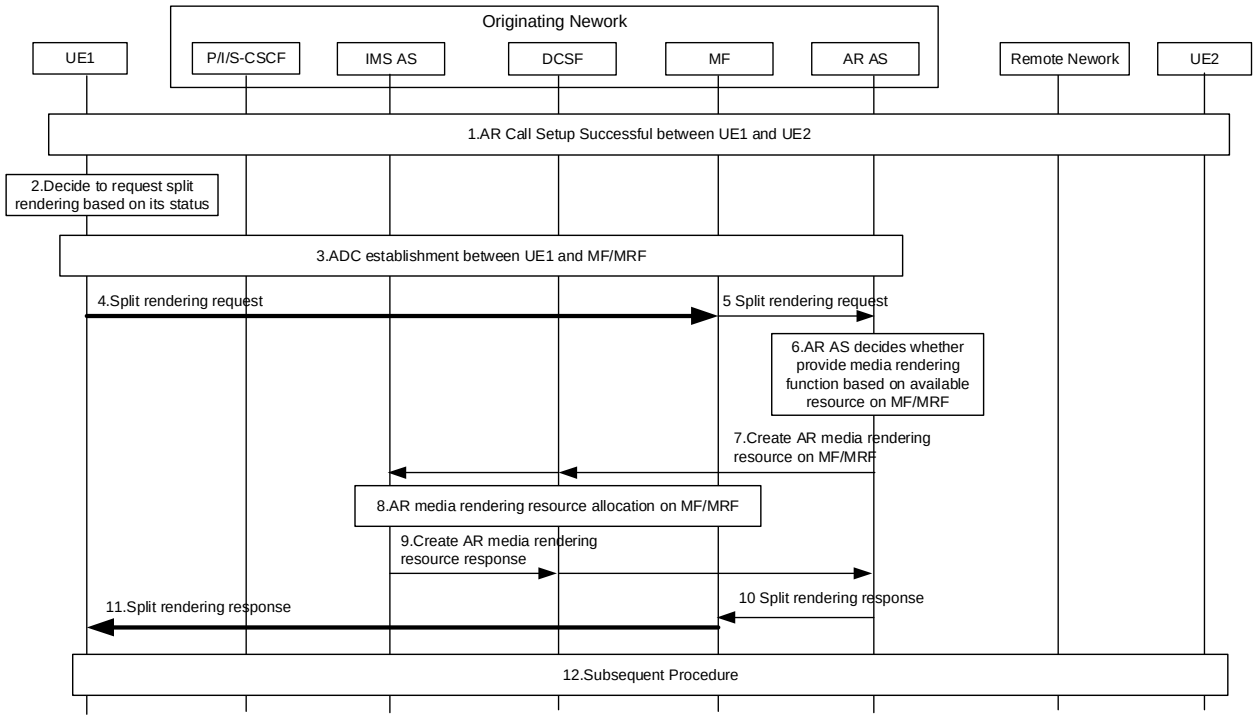


Figure A.1.3-1: Call Flow for Split Rendering Negotiation

The steps are as follows:

1. The UE1 initiates an AR communication session and establishes audio and video session connections with the UE2. Then the bootstrap and application data channels are established for the UE1 and UE2.

2. When the UE1 discovers that its media capabilities cannot meet the AR communication related media rendering requirements, the UE1 decides to start split rendering call flow. Then the UE1 calculates which AR objects can be rendered by itself based on its status, and decides which part of the AR objects to be rendered in the UE1 and the others to be rendered in the IMS network.

3. The UE1 initiates the application data channels between the UE1 and the MF, for the split rendering request and metadata transmission.

4-5. The UE1 sends a Split Rendering Request to the MF through the established application data channel, the request includes the information of the AR objects to be rendered in IMS network, the MF transfers the request to the AR AS.

Note: The use of data channel for split rendering procedure (steps 2, 4 and 5) is optional. Alternatively, the MF or AR AS may decide to use split rendering for media delivery based on, e.g., device capability and media properties.

6. The AR AS may decide whether to provide AR media rendering function based on the request message received from the UE1, the media rendering resource available on the MF and the split rendering provisioning status for the UE1.

7. The AR AS sends a request to the DCSF to create required AR media rendering resource on the MF for the AR objects that should be rendered in IMS network, the DCSF transfers the request to the IMS AS.

NOTE: The requested AR media rendering resources are such that they can accommodate expected changes in scene description for the session.

8. The IMS AS sends the media resource allocation request to the MF, to reserve AR media rendering resource for the UE1.

9. When the MF resources are allocated successfully, the IMS AS returns a successful response to the DCSF, the DCSF transfers the response to the AR AS.

10-11 The AR AS returns a successful Split Rendering Response carrying the result to the MF. The MF then transfers the response/notification message with split rendering configuration information to the UE1 through the application data channel.

12. Subsequent procedures continue.

A.1.4 Scene Description Processing

Figure A.1.4-1 illustrates a detailed call flow for scene description processing procedure.

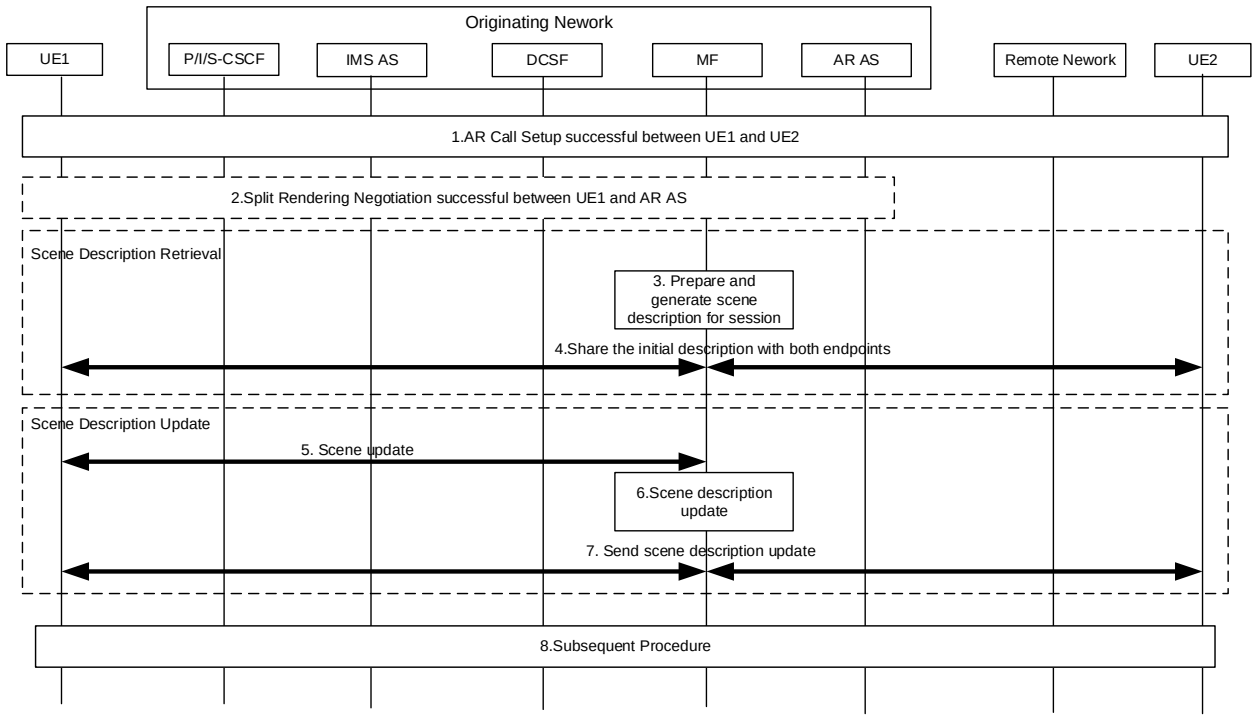


Figure A.1.4-1: Call Flow for Scene Description Processing

The steps are as follows:

1. The UE1 initiates an AR communication session and establishes audio and video session connections with the UE2. Then the bootstrap and application data channels are established for the UE1 and UE2.
2. The split rendering negotiation procedure has been finished.
3. MF prepares the scene description based on media descriptions and assets for the call.
4. MF delivers the scene description to the UEs.
5. A UE may trigger a scene update e.g., when a new object is added/removed in the scene, or a spatial information update is sent or UE capabilities change. The figure shows the update is triggered by UE1, but this can be either UE.
6. The MF will process the new information and creates a scene description update. It is also possible for the MF to initiate an update without an update from the UEs, for example if link conditions change.
7. MF distributes scene description update to all UEs.

NOTE: Spatial data related updates may be required for collaborative AR calls, e.g., when multiple users are physically collocated and also part of the same AR experience. The type of spatial description updates is for further study.

8. Subsequent procedures continue.

A.1.5 AR Media Processing

Figure A.1.5-1 illustrates a detailed call flow for AR media processing procedure.

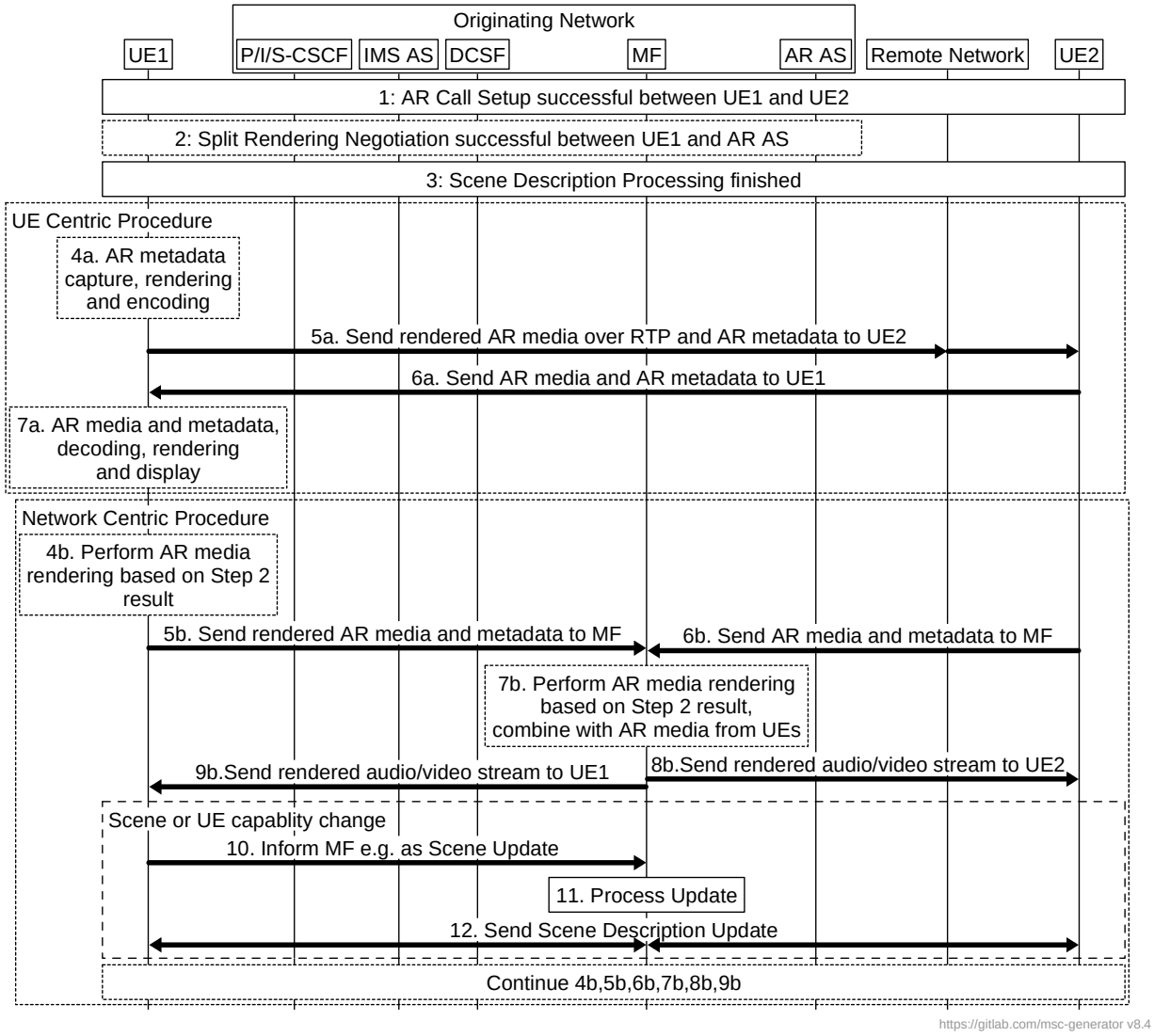


Figure A.1.5-1: Call Flow for AR Media Processing

The steps are as follows:

1. The UE1 initiates an AR communication session and establishes audio and video session connections with the UE2. Then the bootstrap and application data channels are established for the UE1 and UE2.

2. The split rendering negotiation procedure has been finished.

3. The scene description processing has been finished.

4a. The UE1 captures the AR metadata, performs AR media rendering locally and then encodes rendered AR media, e.g., as audio/video media stream.

5a. The UE1 sends the rendered AR media and/or AR metadata to the peer through the established media connection(s).

6a. The UE1 receives the AR media and/or AR metadata from the peer through established media connection(s).

7a. The UE1 decodes the AR metadata and performs AR media rendering locally and displays it on its screen.

Network Centric Procedure:

4b. The UE1 retrieves AR metadata such as pose and user input locally, and renders the part of AR objects that should be done in the UE1 according to the result in step 2.

5b. The UE1 sends AR media to the MF, the AR media can include the audio/video media stream rendered by the UE1, and the AR metadata which is used for the AR objects rendering that should be done in the IMS network.

6b. The UE2 may also send AR media and/or AR metadata to the MF/MRF, e.g., the viewport.

7b. The MF renders the part of AR objects based on the AR metadata received. Then the MF decodes the audio/video media stream received from the UE1, and combines with the AR media rendered by the MF.

8b. The MF sends the rendered audio/video media stream to UE2.

9b. The MF sends the rendered audio/video media stream to the UE1.

10. A UE may trigger a scene update as described in clause 9.1.3. for example, if objects are added/removed from the scene or if UE capabilities change and it can no longer render some AR objects as negotiated in step 2.

11. The MF processes the new information and creates a scene description update. It is also possible for the MF to initiate an update without an update from the UEs, for example if network conditions change.

12. The MF distributes the scene description update to all UEs.

A.2 Avatar Call Flows

A.2.1 General Avatar Call Flow

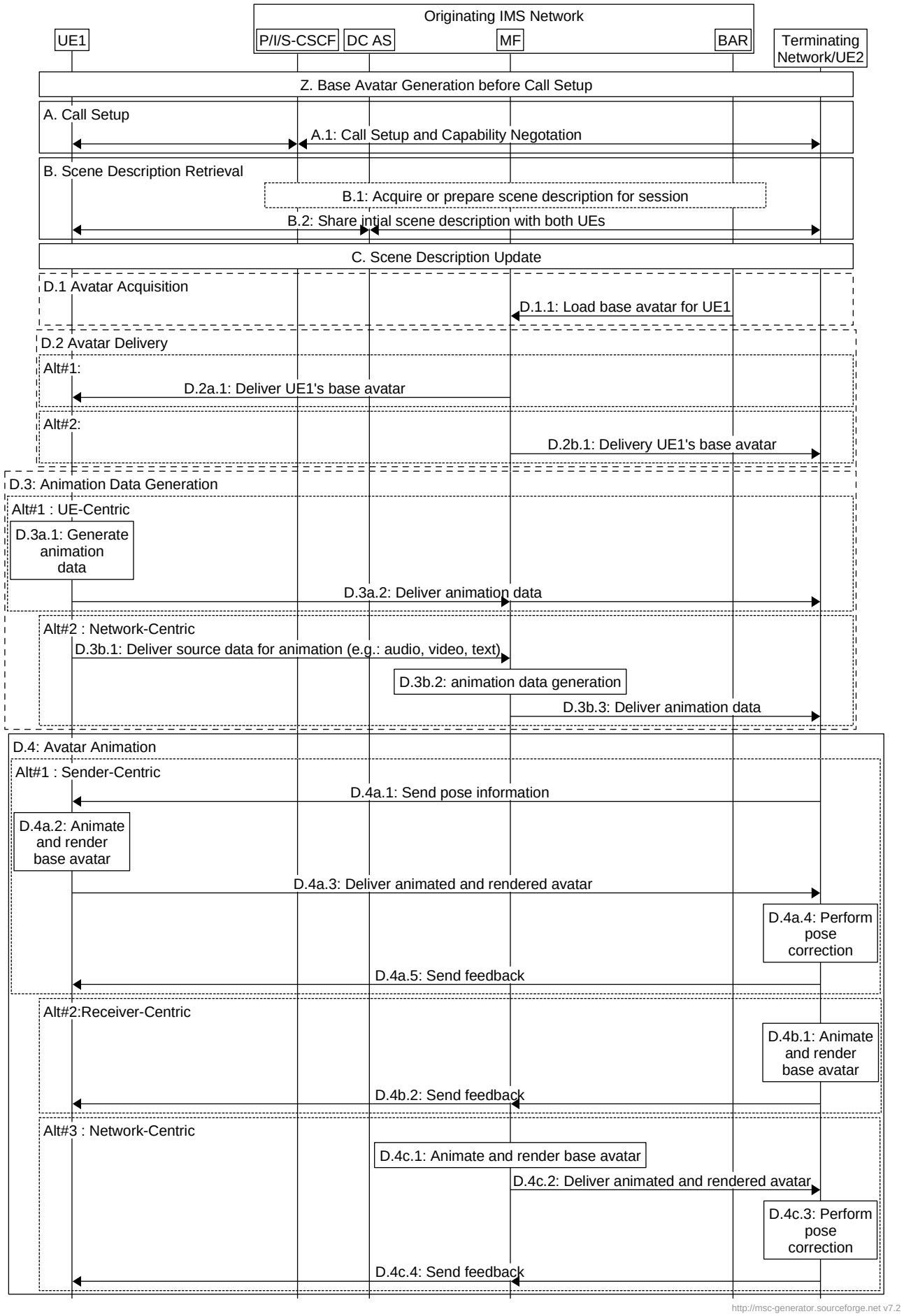

Originating IMS Network
UE1
P/I/S-CSCF
DC AS
MF
BAR
Terminating Network/UE2
Z. Base Avatar Generation before Call Setup
A. Call Setup
A.1: Call Setup and Capability Negotation
B. Scene Description Retrieval
B.1: Acquire or prepare scene description for session
B.2: Share intial scene description with both UEs
C. Scene Description Update
D.1 Avatar Acquisition
D.1.1: Load base avatar for UE1
D.2 Avatar Delivery
Alt#1:
D.2a.1: Deliver UE1's base avatar
Alt#2:
D.2b.1: Delivery UE1's base avatar
D.3: Animation Data Generation
Alt#1 : UE-Centric
D.3a.1: Generate animation data
D.3a.2: Deliver animation data
Alt#2 : Network-Centric
D.3b.1: Deliver source data for animation (e.g.: audio, video, text)
D.3b.2: animation data generation
D.3b.3: Deliver animation data
D.4: Avatar Animation
Alt#1 : Sender-Centric
D.4a.1: Send pose information
D.4a.2: Animate and render base avatar
D.4a.3: Deliver animated and rendered avatar
D.4a.4: Perform pose correction
D.4a.5: Send feedback
Alt#2:Receiver-Centric
D.4b.1: Animate and render base avatar
D.4b.2: Send feedback
Alt#3 : Network-Centric
D.4c.1: Animate and render base avatar
D.4c.2: Deliver animated and rendered avatar
D.4c.3: Perform pose correction
D.4c.4: Send feedback
http://msc-generator.sourceforge.net v7.2

Figure A.2.1-1: IMS Avatar Delivery and Animation Flow

Z. Base Avatar Generation Before Call Setup

The base avatar is generated before step A. Call Setup and Capability Negotiation. It may be uploaded to the BAR by using the Avatar management interface defined in annex B.

A. Call Setup and Capability Negotiation

A.1 An audio/video session is established between UE1 and UE2 and parameters of the session are negotiated.

The list of Avatar ID(s) and/or Avatar Representations is downloaded to the UE by the following options:

Pre-configured in the UE: The Avatar ID List and/or Avatar Representations is provisioned or downloaded to the UE before a data channel for avatar call is setup.,

Through bootstrap data channel: The Avatar ID List is fetched by the DC AS from the BAR when the associated Avatar call application is downloaded and transferred from the DC AS to the DCSF and downloaded to UE through bootstrap data channel (see details in Annex AC 11.3.1 in TS 23.228 [4].

Through application data channel: The Avatar ID List is fetched by the DC AS from the BAR and downloaded to the UE through application data channel).

NOTE 1: Further details of avatar selection and negotiation are defined in clause A.2.3.

B. Scene Description Retrieval

The MF and the participating UEs retrieve the scene description. The scene description may be shared by the MF with the UEs, in case of a shared experience, or the UEs may have their own scene descriptions.

C. Scene Description Update

A scene update trigger occurs, e.g., if an object is added to or removed from a scene or if spatial information is updated. The update trigger may originate from the MF itself or the UEs. The UEs may update their scene descriptions independently or the MF may generate an updated scene description and share it with the UEs.

NOTE 2: The steps B and C are not needed for 2D avatar.

D.1. Avatar Acquisition

D.1.1: The MF loads the base avatar for UE1 from BAR.

D.2. Avatar Delivery

Alternative #1: Sender-centric

D.2a.1: The MF delivers the base avatar of UE 2 to UE1 through data channel.

Alternative #2: Receiver-centric

D.2b.1: The MF delivers the base avatar of UE1 to UE2 through data channel.

D.3. Animation Data Generation

Based on the capability negotiation result in step A, the UE or network may generate animation data.

Alternative #1: UE centric animation data generation

D.3a.1: The UE1 generates the animation data based on the source data (e.g., audio, video, text) or using an XR runtime. The animation data may be transformed from the source data (e.g., from audio to text), or the same as the source data.

D.3a.2: UE1 delivers the animation data to the entity actuating avatar animation through data channel. The animating entity may be the MF or UE2.

Alternative #2: Network centric animation data generation

D.3b.1: UE1 sends source data for animation data generation to the MF over RTP (audio, video, text) or data channel (text).

D.3b.2: The MF processes the received source data to generate animation data during the session. The animation data may be transformed from the source data (e.g., from audio to text, video to motion data), or the same as the source data.

D.3b.3: The MF delivers animation data over data channel to the UE2 animating the base avatar. If network centric avatar animation is used, this step will be skipped. The animation data may be delivered to UE1 as well.

D.4. Avatar Animation

Based on the capability negotiation result in step A, the UE or network may animate the avatar.

Alternative #1a: Sender-centric avatar animation

[Optional] D.4a.1: UE2 delivers its pose information to UE1 for viewer-dependent avatar animation and rendering.

D.4a.2: UE1 animates and renders the base avatar using animation data. The animation data is generated by UE1 in step D.3a.1.1

D.4a.3: UE1 delivers the animated and rendered avatar to UE2. The animated and rendered avatar may be delivered as a 2D video through RTP.

D.4a.4: UE2 corrects the rendered video (for latency compensation) from UE1 before displaying as rendered avatar.

D.4a.5: UE2 delivers a report of timing information, including its actual display time to UE1 for the monitoring of the UE1 centric rendering service.

Alternative #1b: Receiver-centric avatar animation

D.4b.1: UE2 animates and renders the base avatar using animation data. The animation data may be generated by the MF, following steps D.3b.1 to D.3b.2 and received by UE2 in step D.3b.3 or it may be generated by UE1 in step D.3a.1 and received by UE2 in step D.3a.2.

D.4b.2: UE2 delivers a report of timing information, including its actual display time to UE1 (and MF) for the monitoring of the UE1 centric rendering service.

Alternative #1c: Network-centric avatar animation

D.4c.1: The MF animates and renders the UE1’s base avatar using animation data. The animation data may be generated by the MF, following step D.3b.1 and D.3b.2 or it may be received from UE1 following steps D.3a.1 and D.3a.2.

D.4c.2: The MF delivers the animated and rendered avatar to the UEs. In the figure, delivery to UE2 is shown as example. The animated and rendered avatar may be delivered as a 2D video through RTP.

D.4c.3: UE2 corrects the rendered video (for latency compensation) from the MF before displaying as rendered avatar.

D.4c.4: UE2 delivers a report of timing information, including its actual display time to UE1 (and MF) for the monitoring of the network-centric rendering service.

NOTE 3: Rendering is not needed for 2D avatar.

NOTE 4: Network centric modes include both MF rendering mode and DC AS rendering mode. Alternatively, DC AS also can perform the above functions if the DC AS rendering mode is chosen. This specification currently only considers MF based rendering mode.

A.2.2 Avatar Management Call Flow

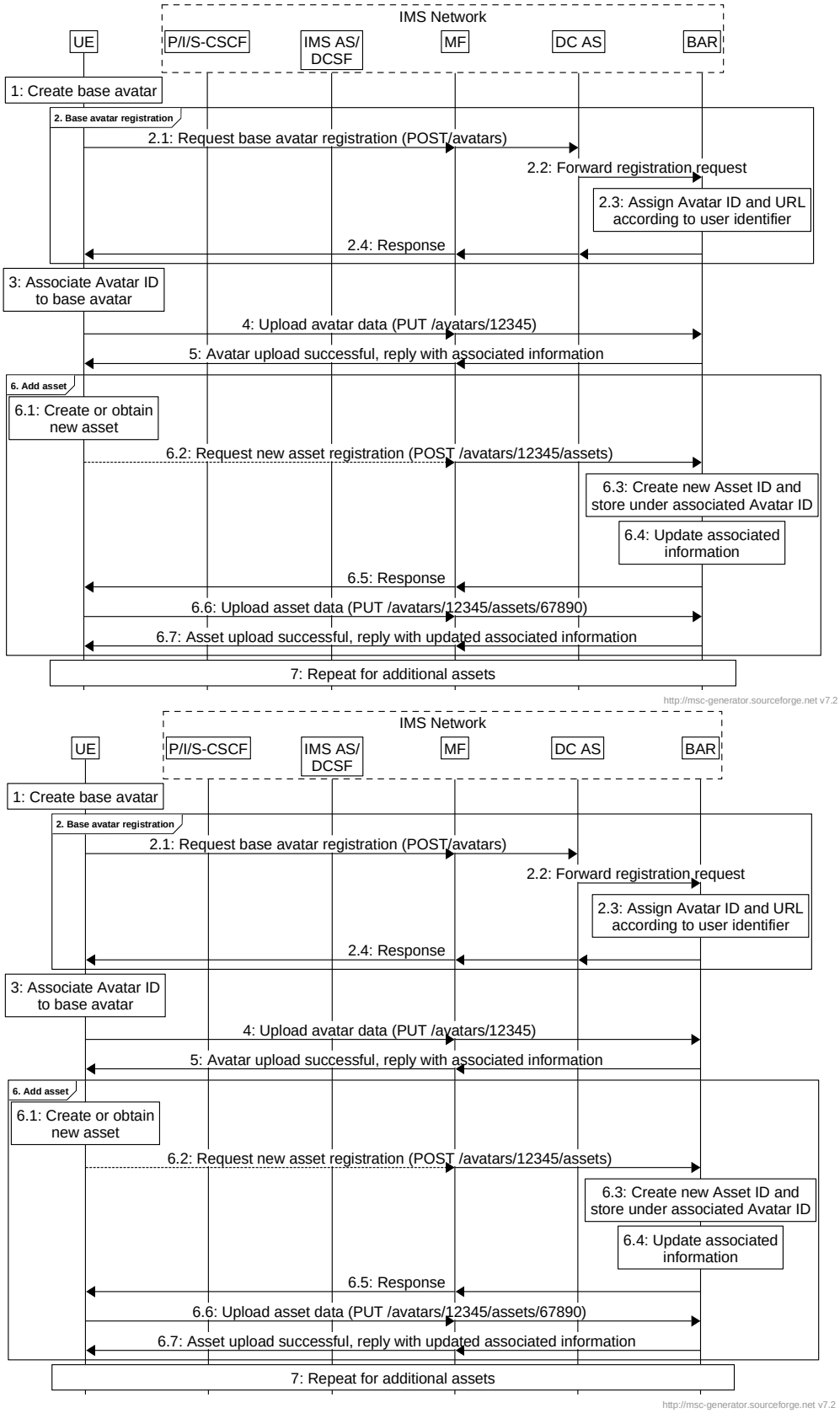


Figure A.2.2-1: Avatar management call flow via IMS network for registering and uploading base avatar and associated assets

Figure A.2.2-1 depicts the call flow procedure for registering and uploading a user's base avatar and associated assets. The main steps in the call flow are as follows:

1. The UE creates the base avatar.
2. Base avatar registration (the use of an Avatar ID assigned by the BAR is required for the secure upload of the base avatar by the UE):

   2.1 The UE sends registration request to the MF/DC AS via application data channel to request the registration of its base avatar.

   2.2 The DC AS forwards the registration request to the BAR.

   2.3 The BAR assigns a unique Avatar ID and URL for the base avatar of the UE according to the UE identifier known via the DC AS.

   NOTE: How the BAR obtains and maps user identifiers to Avatar IDs is FFS.

   2.4 The BAR sends a registration response containing the Avatar ID and URL for the registered base avatar to the UE via the MF/DC AS.
3. The UE associates the assigned Avatar ID to the Base Avatar data created from step 1.
4. The UE uploads the base avatar data to the BAR via the MF using the application data channel.
5. The BAR replies with associated information for the registered base avatar.
6. Adding new assets to the base avatar:

   6.1 The UE creates or obtains a new asset for adding to the base avatar.

   6.2 The UE sends an asset registration request for the new asset to the BAR via the MF using the application data channel.

   6.3 The BAR creates a new Asset ID for the new asset and associates it to the Avatar ID of the base avatar.

   6.4 The BAR updates the associated information corresponding to the Avatar ID.

   6.5 The BAR sends an asset registration response containing the Asset ID to the UE via the MF.

   6.6 The UE uploads the asset data to the BAR via the MF.

   6.7 The BAR replies with updated associated information.
7. Steps 6.1 to 6.7 are repeated for the registration of additional assets.

A.2.3 Avatar Selection and Negotiation Call Flow

For avatar call over the IMS data channel, the avatar ID list (a list of the base avatars available in the BAR) is obtained by the UE using one of following options:

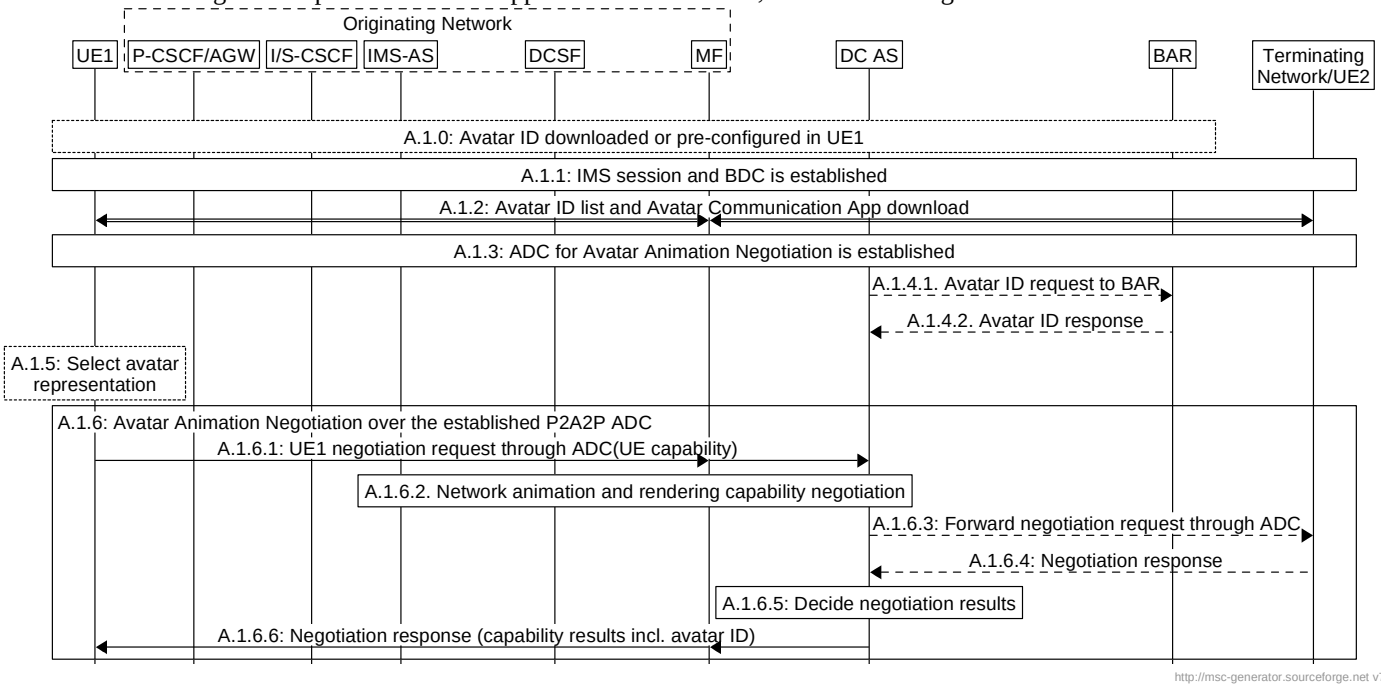
- Pre-configured in the UE: The Avatar ID List and/or Avatar Representations are provisioned or downloaded to the UE before any session for the avatar call is established.
- Through bootstrap data channel: The Avatar ID List is fetched by the DC AS from the BAR when the associated Avatar call application is downloaded and transferred from the DC AS to the DCSF and downloaded to UE through the bootstrap data channel.
- Through application data channel: The Avatar ID List is fetched by the DC AS from the BAR and downloaded to the UE through an application data channel.

Three avatar animation modes are defined for avatar call over the IMS data channel:

- Sender-centric: the sender UE animates and renders its base avatar before sending it to the receiving UE as 2D video.
- Receiver-centric: the receiving UE animates and renders the sender UE's base avatar.

- Network-centric: the MF animates and renders the sender UE's base avatar before sending it to the receiving UE as 2D video.

The decision of which avatar animation mode to use for avatar call is dependent on the outcome of the capability and Avatar Animation Negotiation procedure via an application data channel, as detailed in Figure A.2.3-1.



A.2.3-1: Avatar selection and negotiation call flow

The avatar animation negotiation procedure is based on the avatar type (2D or 3D) and the capability information of the sender/receiver UEs and the MF. The capability information includes the animation data type(s) (e.g., text, expression data, and motion signals for joints) supported by either UEs or the MF. For network centric mode, after avatar animation negotiation, the IMS AS instructs the MF to download UE1's base avatar from the BAR, generate animation data from the source data received from UE1, and animate UE1's base avatar using the animation data received from UE1 or generated by the MF itself.

A.1.0: (optional) An Avatar ID List is pre-downloaded, or pre-configured in UE1.

NOTE 1: Step A.1.0 is optional; in this step the Avatar ID List is provisioned or downloaded to the UE before any session for the avatar call is setup. The UE and the BAR may interact by means out of the scope of 3GPP.

A.1.1: An IMS session is established between UE1 and UE2, and a bootstrap data channel is established between UE1, the MF, and the DC AS.

A.1.2: The Avatar ID List and the Avatar call App are downloaded to UE1 via the BDC (see details in AC 11.3.1 in TS 23.228 [4]).

A.1.3: A P2A2P application data channel for Avatar Animation Negotiation is established between UE1, the MF DC AS, and UE2. If the Avatar ID List has not been downloaded in either steps A.1.0 or A.1.2, it may optionally be requested via this ADC.

NOTE 2: TS 23.228 [4] only defines MF allocation during the initial IMS session procedure and not during the IMS session modification procedure. Whether and how a suitable MF with avatar capabilities can be re-allocated after avatar negotiation is FFS.

A.1.4.1. (optional): If an Avatar ID list is not obtained in A.1.0 or A.1.2, UE1 may send an Avatar ID List request to the DC AS via MF; the DC AS then sends the request to the BAR.

A.1.4.2 (optional) If the BAR receives an Avatar ID List request, the BAR (generates and) sends the Avatar ID List to UE1 via the DC AS and MF

NOTE 3: Steps A.1.4.1 and A.1.4.2 are optional. Whether and which user identity(ies) should be used by the user of the sending UE (UE1) and/or the receiving UE (UE2) for the download of the avatar representations in the case of a receiving UE centric rendering mode will be decided by SA WG3 and the procedure will be aligned with SA WG3's decision.

A.1.5: UE1 selects an avatar representation to be used for the avatar call using the list of available avatar representations known to UE1 via the Avatar ID List.

A.1.6: Avatar animation negotiation takes place via the established P2A2P ADC.

A.1.6.1: UE1 sends an avatar animation negotiation request using the ADC through the MF to the DC AS. The message carries parameters which may include: an avatar ID associated with the selected avatar representation selected in step A.1.4, animation data types (e.g., text, expression data, or motion signals for joints) supported by UE1, and related rendering requirements or capability information.

A.1.6.2: (optional) To facilitate the negotiation of the rendering mode, the DC AS may interact with the current serving MF to check the MF capability through the DC1, DC2, and DC3/DC4 interfaces The MF sends a response with its avatar capability information to the DC AS that enable the DC AS to make further decisions.-For example, to determine whether network-centric rendering is supported and which rendering mode (MF or DC AS) to use.

NOTE 4: Whether the interaction between the DC AS and the MF is supported needs confirmation from SA2. In particular, the MF response with its supported capabilities to the DC AS.

A.1.6.2: (optional) Through an established P2A application data channel, MF/DC AS sends a capability negotiation request to UE2. The message may include the same information as in described in A.1.5.1.

A.1.6.3: (optional) UE2 sends a capability negotiation response to the MF/DC AS. The message carries the capability negotiation result related to UE2's preference.

A.1.6.4: The DC AS gets the avatar type (2D or 3D) associated with the avatar ID from base avatar retrieved from BAR or to be generated by the MF and confirms the avatar animation negotiation result based on the avatar type and the capabilities supported by UE1, the MF, and UE2. The capability negotiation result includes the rendering mode and animation method (e.g., by audio, text, or expression data and motion signals for joints).

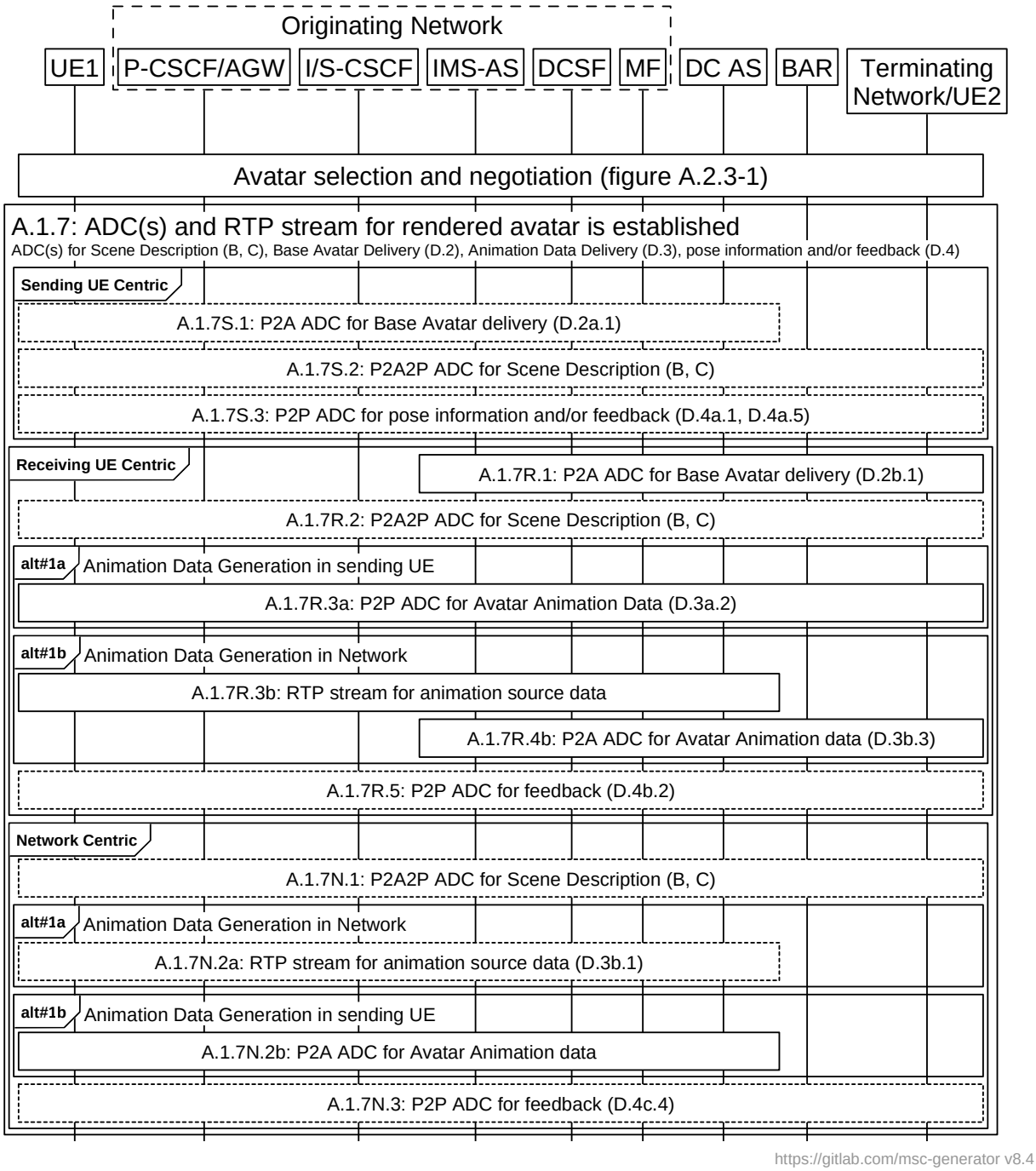
A.1.6.5: The DC AS sends the capability negotiation response to UE1 through the MF. The message carries the capability negotiation result.

15th Change

A.2.4 ADC and RTP Stream Establishment for Avatar Call

As a result of the avatar selection and negotiation procedure as described in clause A.2.3, depending on the configuration negotiated, certain application data channels and/or RTP streams are established between different endpoints (UE1, MF, UE2) for the delivery of different avatar data components as shown in Figure A.2.1-1. These include (where necessary) the delivery of scene description, base avatar, animation source data, animation data, pose information and feedback information.

The decision of which avatar animation mode to use for avatar call is dependent on the outcome of the capability and Avatar Animation Negotiation procedure, as detailed in Figure A.2.3-1.



A.2.4-1: ADC(s) and RTP stream establishment call flow

Avatar selection and avatar animation mode negotiation takes place as described in clause A.2.3. Depending on the configuration negotiated, delivery channels are established between certain endpoints as shown in Figure A.2.4-1.

A.1.7: Depending on the avatar negotiation result for the selected avatar representation, media re-negotiation (via SDP) takes place in order to establish the necessary ADCs and RTP streams to deliver the various avatar data. In the case where media processing in the network is required but not supported by the allocated MF, an additional MF supporting such processing may be allocated. Avatar data may include: scene description, base avatar, animation data, pose information, and feedback information.

Sending UE Centric:

A.1.7S.1: P2A ADC (UE1) is established for base avatar delivery (Figure A.2.1-1 step D.2a.1).

A.1.7S.2: P2A2P ADC is established for scene description retrieval and update (Figure A.2.1-1 steps B and C).

A.1.7S.3: P2P ADC is established for pose information and/or feedback information respectively (Figure A.2.1-1 steps D.4a.1 and D.4a.5).

Receiving UE Centric:

A.1.7R.1: P2A ADC (UE2) is established for base avatar delivery (Figure A.2.1-1 step D.2b.1).

A.1.7R.2: P2A2P ADC is established for scene description retrieval and update (Figure A.2.1-1 steps B and C).

alt#1a Animation Data Generation in sending UE

A.1.7R.3a: P2P ADC is established for avatar animation data delivery (Figure A.2.1-1 step D.3a.2).

alt#1b Animation Data Generation in Network

A.1.7R.3b: RTP stream is established for animation source data delivery (Figure A.2.1-1 step D.3b.1).

A.1.7R.4b: P2A ADC (UE2) is established for avatar animation data delivery (Figure A.2.1-1 step D.3b.2).

A.1.7R.5: P2P ADC is established for feedback information (Figure A.2.1-1 step D.4b.2)

Network Centric:

A.1.7N.1: P2A2P ADC is established for scene description retrieval and update (Figure A.2.1-1 steps B and C).

alt#1a Animation Data Generation in Network

A.1.7N.2a: RTP stream is established for animation source data delivery (Figure A.2.1-1 step D.3b.1).

alt#1b Animation Data Generation in sending UE

A.1.7N.2b: P2A ADC is established for avatar animation data delivery (Figure A.2.1-1 step D.3a.2).

A.1.7N.3: P2P ADC is established for feedback information (Figure A.2.1-1 step D.4c.4)

16th Change

Annex B:
Base Avatar Management Interface

B.1 Mbar_Management service

B.1.1 Overview

This clause defines the BAR management API offered by the BAR and used by the DC AS or MF to manage avatar related data in the BAR. A summary of the resource structure is shown in Table B.1-1 below.

Table B.1-1: Resource structure of Mbar_Management APIs

HTTP request path element hierarchy	Description	Allowed HTTP methods					Resource structure definition clause
		Create	Retrieve	Update	Destroy	Non-RESTful operation	
avatars	Avatar collection	POST					B.1.2
{avatarId}	Avatar resource		GET	PUT, PATCH	DELETE		

assets	Asset collection	POST					B.1.3
{assetId}	Asset resource		GET	PUT	DELETE		
associatedInfo	Associated Information resource		GET				B.1.4
representations	Avatar Representation collection	POST					B.1.5
{avatarRepresentationId}	Avatar Representation resource		GET	PUT, PATCH	DELETE		
sessions	Session collection	POST					B.1.6
{sessionId}	Session resource		GET	PUT	DELETE		

B.1.2 HTTP resource URIs and paths

The resource URI used in each HTTP request to the API shall have the structure defined in subclause 4.4.1 of TS 29.501 [14], i.e.:

{apiRoot}/{apiName}/{apiVersion}/{apiSpecificResourceUriPart}

with the following components:

- *{apiRoot}* shall be set as described in TS 29.501 [14].
- *{apiName}* shall be set as defined by the following clauses.
- *{apiVersion}* shall be set to "v1".
- *{apiSpecificResourceUriPart}* shall be set as described in the following clauses.

B.1.3 Usage of HTTP

B.1.3.1 HTTP protocol version

The version-independent semantics of HTTP specified in RFC 9110 [11] shall be followed.

Support of HTTP/1.1 (per RFC 9112 [12]) over TLS is mandatory and support of HTTP/2 (per RFC 9113 [18]) over TLS is recommended.

B.1.3.2 HTTP response codes

Guidelines for HTTP *4xx (Client Error)* status codes in response to the invocation of the BAR APIs defined in this annex are specified in clause 4.8 of TS 29.501 [14].

B.1.4 Common API data types

B.1.4.1 General

The data types defined in this clause are intended to be used by more than one of the BAR APIs.

B.1.4.2 Simple Data Types

This annex refers to some common simple data types defined in the OpenAPI specification [13]. These data types are listed in Table B.1-2. Additionally, Table B.1-3 specifies common simple data types used within the BAR API, including a short description of each. In cases where types from other specifications are reused, a reference is provided.

Table B.1-2: Reused OpenAPI simple data type.

Type Name	Description
boolean	As defined in OpenAPI Specification [13]
integer	As defined in OpenAPI Specification [13]
number	As defined in OpenAPI Specification [13]
string	As defined in OpenAPI Specification [13]
object	As defined in OpenAPI Specification [13]
array	As defined in OpenAPI Specification [13]
NOTE: Data types defined in OpenAPI Specification [13] do not follow the UpperCamel convention for data types in 3GPP TS 29.501 [14].	

Table B.1-3: Reused data types from other specifications.

Type name	Type definition	Description	Reference
ResourceId	`string`	String chosen by the BAR to serve as an identifier in a resource URL.	TS 26.512 [19] Table 6.4.2-1.
Url	`string`	Uniform Resource Locator, conforming with the URI Generic Syntax.	IETF RFC 3986 [15]

B.1.5 Avatars API

B.1.5.1 Overview

The Avatars API is used by the DC AS or MF to manage Base Avatars (including related assets and associated information) in the BAR, providing operational functions such as Base Avatar creation, retrieval, update and deletion.

B.1.5.2 Resource structure

The Avatars API is accessible through the following URL base path:

{apiRoot}`/3gpp-mbar-management/`*{apiVersion}*`/avatars/`

Table B.1-4 specifies the operations and the corrresopnding HTTP methods that are supported by this API. In each case, the sub-resource path specified in the second column of the table shall be appended to the above URL base path.

Table B.1-4: Operations supported by the Avatars API

Operation name	Sub-resource path	Allowed HTTP method(s)	Description
Create Avatar		`POST`	Creates a new avatar resource in the BAR.
Get Avatar	*{avatarId}*	`GET`	Used to retrieve a previously created or uploaded base avatar in the BAR.
Update Avatar		`PUT, PATCH`	Used to upload or update Base Avatar data corresponding to an Avatar ID.
Delete Avatar		`DELETE`	Removes and deletes a Base Avatar, as well as its related assets and associated information.

B.1.5.3 Data model

B.1.5.3.1 Avatar resource

Table B.1-5: Definition of Avatar resource

Property name	Data type	Cardinality	Usage	Description
avatarId	`ResourceId`	1..1	C: RO R: RW U: –	A unique identifier assigned to a Base Avatar by the BAR on creation.
ownerId	`string`	1..1	C: RW R: RO U: –	A unique identifier identifying the subscriber (owner) associated with the base avatar specified by *avatarId* in this resource.
assetIds	`array(ResourceId)`	0..1	C: RO R: RO U: –	A list of assets associated with the Base Avatar.
avatarContainer	`URL`	0..1	C: RW R: RO U: RW	Payload containing the Base Avatar data and associated assets. This provides access to the full binary avatar container, including all of the contained assets. For creation and update operations, the URL shall point to a multi-part mime part with MIME type “model/vnd.mpeg.arf+zip”.
associatedInfo	`AssociatedInfo`	0..1	C: RO R: RO U: RO	Associated information related to the Base Avatar.

B.1.6 Assets API

B.1.6.1 Overview

The Assets API is used by the DC AS or MF to manage individual assets of the base avatar in the BAR, providing operational functions such as asset creation, retrieval, update and deletion.

B.1.6.2 Resource structure

The Assets API is accessible through the following URL base path:

{apiRoot}`/3gpp-mbar-management/`*{apiVersion}*`/avatars/`*{avatarId}*`/assets/`

Table B.1-6 specifies the operations and the corrresopnding HTTP methods that are supported by this API. In each case, the sub-resource path specified in the second column of the table shall be appended to the above URL base path.

Table B.1-6: Operations supported by the Assets API

Operation name	Sub-resource path	Allowed HTTP method(s)	Description
Create Asset		POST	Creates a new asset resource in the BAR.
Get Asset	*{assetId}*	GET	Used to retrieve a previously created or uploaded asset in the BAR.
Update Asset		PUT	Used to upload or update asset data corresponding to an Asset ID.
Delete Asset		DELETE	Removes and deletes an asset.

B.1.6.3 Data model

B.1.6.3.1 Asset resource

Table B.1-7: Definition of Asset resource

Property name	Data type	Cardinality	Usage	Description
assetId	ResourceId	1..1	C: RO R: RW U: –	A unique identifier assigned to an asset by the BAR on creation. The assetId is scoped by the avatarId.
namespace	string	1..1	C: RW R: RO U: RW	A namespace defining the intended usage of the asset, as exemplified by names such as "human/head" or "accessory/hat"
LoD	array(string)	0..1	C: RW R: RO U: RW	A list of available LoDs for the corresponding asset. NOTE: The labels for LoDs are FFS.
assetData	array(URL)	0..1	C: RW R: RO U: RW	List of URLs that point to the asset data. The primary URL shall point into an ARF document that describes all components of the asset. For creation/update of an asset, all components shall be provided as part of a multi-part mime body.
associatedInfo	AssociatedInfo	0..1	C: RO R: RO U: RO	Associated information related to the Base Avatar.

B.1.7 Associated Information API

B.1.7.1 Overview

The Associated Information API is used by the DC AS or MF to fetch Associated Information related to a Base Avatar from the BAR.

B.1.7.2 Resource structure

The Associated Information API is accessible through the following URL base path:

{apiRoot}/3gpp-mbar-management/*{apiVersion}*/avatars/*{avatarId}*/associatedInfo/

Table B.1-8 specifies the operations and the corrresopnding HTTP methods that are supported by this API. In each case, the sub-resource path specified in the second column of the table shall be appended to the above URL base path.

Table B.1-8: Operations supported by the Associated Information API

Operation name	Sub-resource path	Allowed HTTP method(s)	Description
Get Associated Information		GET	Used to retrieve associated information corresponding to a Base Avatar (identified by the AvatarID) in the BAR.

B.1.7.3 Data model

B.1.7.3.1 Associated information resource

Table B.1-9: Definition of Associated Information resource

Property name	Data type	Cardinality	Description
associatedInfo	Object	1..1	A list of assets associated with the Base Avatar.
avatarId	ResourceId	1..1	A unique identifier assigned to a Base Avatar by the BAR on creation.
avatarMetadata	Object	1..1	Metadata related to the Avatar,
assetIds	array(ResourceId)	1..1	A list of assets associated with the Base Avatar.
assetLoDs	array(Object)	1..1	A list of available LoDs for the corresponding asset. The resulting size in bytes shall be associated with each LoD. NOTE 1: LoDs descriptions/labels and their associated complexity is FFS. NOTE 2: Clarification about how LoD would be described and what that would mean in terms of complexity is FFS.
selectionInfo	Object	0..1	Provides information that the user can use to select this avatar. This may contain a name, a nickname of the asset, usage context e.g. casual, work, and images of renditions of the asset.
supportedAnimations	array(string)	1..1	A list of the URNs that identify the supported animation frameworks by this base avatar.
infoUpdatedAt	number	1..1	A timestamp (in wall clock time) describing the time of the last update to the associated information for the corresponding Base Avatar. This field is updated whenever an asset is modified by the owner of the Base Avatar or BAR. Users may utilize the infoUpdatedAt field to verify the latest validity of previously downloaded Base Avatar data. Comparing the infoUpdatedAt value with the downloaded time allows users to determine if the downloaded data requires updating,

B.1.8 Avatar Representations API

B.1.8.1 Overview

The Avatar Representations API is used by the DC AS or MF to select specific Base Avatars (including related assets and associated information) in the BAR for use during an avatar call. An owner of a Base Avatar may select a base avatar and certain associated assets in order to create an Avatar Representation resource which can be stored in the BAR.

B.1.8.2 Resource structure

The Avatar Representation Selection API is accessible through the following URL base path:

{apiRoot}/3gpp-mbar-management/*{apiVersion}*/avatars/*{avatarId}*/representations/

Table B.1-10 specifies the operations and the corrresopnding HTTP methods that are supported by this API. In each case, the sub-resource path specified in the second column of the table shall be appended to the above URL base path.

Table B.1-10: Operations supported by the Avatar Selection Instruction API

Operation name	Sub-resource path	Allowed HTTP method(s)	Description
Create Avatar Representation		POST	Creates an Avatar Representation resource in the BAR.
Get Avatar Representation	*{avatarRepresentationId}*	GET	Used to retrieve a previously created Avatar Representation resource in the BAR.
Update Avatar Representation		PUT, PATCH	Used to update Avatar Representation resource corresponding to an Avatar Representation ID.
Delete Avatar Representation		DELETE	Removes and deletes an Avatar Representation.

B.1.8.3 Data model

B.1.8.3.1 Avatar representation resource

Table B.1-11: Definition of Avatar representation resource

Property name	Data type	Cardinality	Usage	Description
avatarRepresentationId	ResourceId	1..1	C: RO R: RW U: –	A unique identifier assigned to an Avatar Representation by the BAR on creation.
ownerId	string	1..1	C: RW R: RO U: –	A unique identifier identifying the subscriber (owner) associated with the base avatar specified in this resource.
assetIds	array(ResourceId)	1..1	C: RW R: RO U: RW	Identifies the assets to be shared.
assetLoDs	array(Object)	1..1	C: RW R: RO U: RW	Identifies the allowed LoD's for each asset selected.
publishTime	number	1..1	C: RO R: RO U: –	Describes the issue time or latest update time (in wall clock time) of the Avatar Representation.

B.1.9 Sessions API

B.1.9.1 Overview

The Sessions API is used by the DC AS or MF to provide session-scoped configuration and authorization for avatar usage during an avatar call. A session resource is created to bind the base avatar to be used, a subset of the assets and LoDs selected for that session. An existing avatar representation resource is bound to a session resource to define the selected base avatar and assets. When the identifiers for *ownerId* and the session *creatorId* match, a token is issued by the BAR.

NOTE: Creating a session resource intended only for a specific user is for FFS.

B.1.9.2 Resource structure

The Sessions API is accessible through the following URL base path:

{apiRoot}/3gpp-mbar-management/*{apiVersion}*/sessions/

Table B.1-12 specifies the operations and the corrresopnding HTTP methods that are supported by this API. In each case, the sub-resource path specified in the second column of the table shall be appended to the above URL base path.

Table B.1-12: Operations supported by the Sessions API

Operation name	Sub-resource path	Allowed HTTP method(s)	Description
Create Session		POST	Creates an Session resource in the BAR.
Get Session	*{sessionId}*	GET	Used to retrieve a previously created Session resource in the BAR.
Update Session		PUT	Used to update Session resource corresponding to a Session ID.
Delete Session		DELETE	Removes and deletes the Session and revokes its associated access token.
Get Session Token	*{sessionId}*/token	GET	Returns current token metadata for the Session.
Rotate Session Token		POST	Issues a new short lived access token for the Session and invalidates the previous token.
Revoke Session Token		DELETE	Revokes the current Session token without deleting the Session.

B.1.9.3 Data model

B.1.9.3.1 Session resource

Table B.1-13: Definition of Session resource

Property name	Data type	Cardinality	Usage	Description
sessionId	ResourceId	1..1	C: RO R: RW U: –	Unique identifier assigned by the BAR upon session creation.
avatarId	ResourceId	1..1	C: RO R: RW U: –	A unique identifier assigned to a base avatar by the BAR.
avatarRepresentationId	ResourceId	1..1	C: RO R: RW U: –	A unique identifier assigned to an Avatar Representation by the BAR on creation.
ownerId	string	1..1	C: RW R: RO U: –	A unique identifier identifying the subscriber (owner) associated with the base avatar specified by *avatarId* in this resource.
creatorId	string	1..1	C: RW R: RO U: –	A unique identifier identifying the creator of the session resource.
callStartTime	number	1..1	C: RW R: RO U: RW	The wall clock time at the start time of an avatar call period.
callEndTime	number	1..1	C: RW R: RO U: RW	Specifies the end time (in wall clock time) of an avatar call period. *callEndTime* may be extended while both the owner and intended user of the Base Avatar are online. NOTE: DC AS may collect ping within time out to determine connectivity.

publishTime	`number`	1..1	C: RO R: RO U: –	Describes the issue time or latest update time (in wall clock time) of the session resource.
accessToken	`SessionAccess`	1..1	C: RO R: RW U: –	Access token and validity for dereferencing BAR resources within the scope of this session.

B.1.9.3.2 SessionAccess type

Table B.1-14: Definition of SessionAccess type

Property name	Data type	Cardinality	Description
token	SessionToken	1..1	The bearer token that authorizes session-scoped access to BAR resources referenced by this session.
issuedAt	number	1..1	Token issuance time (ms since epoch).
expiresAt	number	1..1	Absolute expiry (ms since epoch). Tokens are short-lived and shall not exceed operator policy TTL.

B.1.9.3.3 SessionToken type

Table B.1-15: Definition of SessionToken type

Property name	Data type	Cardinality	Description
scheme	string	1..1	Token scheme; "bearer" shall be supported.
value	string	1..1	Opaque token value. Its internal structure is operator-specific.

Annex <B> (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2022-08	SA4#120-e	S4-221017				Initial skeleton	0.0.1
2022-08	SA4#120-e	S4-221202				Version agreed during SA4#120-e	0.1.0
2023-08	SA4#125	S4-231474				Scope, references, definitions, and general descriptions	0.2.0
2023-08	SA4#125	S4-231476				Spatial description format	0.2.0
2023-11	SA4#126	S4-231783				Draft TS 26.264 v0.2.1 implemented S4aR230099 on end-to-end reference architecture and S4aR230121 on the usage of a scene	0.3.0
2023-11	SA4#126	S4-231784				Add general requirements applied both AR-MTSI and MTSI clients and remove unnecessary clauses	0.3.0
2023-11	SA4#126	S4-231785				Reuse Split Rendering Formats for session setup and negotiation and metadata defined in TS 26.565	0.3.0
2023-11	SA4#126	S4-231967				Generalized reference architecture based on IMS DC architecture in TS 23.228	0.3.0
2024-02	SA4#127	S4-240393				Agreements in SA4#127: S4-240163, S4-240165, S4-240370, S4-240323	0.4.0
2024-03	SA#103	SP-240024				Version 1.0.0 created by MCC	1.0.0
2024-04	SA4#127-bis-e	S4-240804				Agreements in SA4#127-bis-e: S4-240655, S4-240763, S4-240823	1.1.0
2024-05	SA4#128	S4-241259				Agreements in SA4#128: S4-201186, S4-241216, S4-241219, S4-241217	1.2.0
2024-06						Version 2.0.0 created by MCC	2.0.0
2024-06						Version 18.0.0 created by MCC after TSG approval	18.0.0
2024-09	SA#105	SP-241119	0001		D	Editorial Corrections to TS 26.264	18.1.0
2025-03	SA#107	SP-250138	0002	1	F	Remove MRF from IMS data channel architecture	18.2.0
2025-09	SA#109	SP-250926	0006	2	B	CR for Avatar calls in IMS	19.0.0

History

Document history		
V19.0.0	October 2025	Publication