



**LTE;
5G;
Support of 5G glass-type Augmented Reality / Mixed Reality
(AR/MR) devices
(3GPP TR 26.998 version 17.1.0 Release 17)**



Reference

RTR/TSGS-0426998vh10

Keywords

5G,LTE

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

Important notice

The present document can be downloaded from:

<http://www.etsi.org/standards-search>

The present document may be made available in electronic versions and/or in print. The content of any electronic and/or print versions of the present document shall not be modified without the prior written authorization of ETSI. In case of any existing or perceived difference in contents between such versions and/or in print, the prevailing version of an ETSI deliverable is the one made publicly available in PDF format at www.etsi.org/deliver.

Users of the present document should be aware that the document may be subject to revision or change of status.

Information on the current status of this and other ETSI documents is available at

<https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>

If you find errors in the present document, please send your comment to one of the following services:

<https://portal.etsi.org/People/CommitteeSupportStaff.aspx>

If you find a security vulnerability in the present document, please report it through our
Coordinated Vulnerability Disclosure Program:

<https://www.etsi.org/standards/coordinated-vulnerability-disclosure>

Notice of disclaimer & limitation of liability

The information provided in the present deliverable is directed solely to professionals who have the appropriate degree of experience to understand and interpret its content in accordance with generally accepted engineering or other professional standard and applicable regulations.

No recommendation as to products and services or vendors is made or should be implied.

No representation or warranty is made that this deliverable is technically accurate or sufficient or conforms to any law and/or governmental rule and/or regulation and further, no representation or warranty is made of merchantability or fitness for any particular purpose or against infringement of intellectual property rights.

In no event shall ETSI be held liable for loss of profits or any other incidental or consequential damages.

Any software contained in this deliverable is provided "AS IS" with no warranties, express or implied, including but not limited to, the warranties of merchantability, fitness for a particular purpose and non-infringement of intellectual property rights and ETSI shall not be held liable in any event for any damages whatsoever (including, without limitation, damages for loss of profits, business interruption, loss of information, or any other pecuniary loss) arising out of or related to the use of or inability to use the software.

Copyright Notification

No part may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm except as authorized by written permission of ETSI.

The content of the PDF version shall not be modified without the written authorization of ETSI.

The copyright and the foregoing restriction extend to reproduction in all media.

© ETSI 2022.
All rights reserved.

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 Web server (<https://ipr.etsi.org/>).

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™** and **LTE™** 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 Report (TR) 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 under <http://webapp.etsi.org/key/queryform.asp>.

Modal verbs terminology

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

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

Contents

Intellectual Property Rights	2
Legal Notice	2
Modal verbs terminology.....	2
Foreword.....	6
Introduction	6
1 Scope	7
2 References	7
3 Definitions, symbols and abbreviations	10
3.1 Definitions	10
3.2 Symbols.....	11
3.3 Abbreviations	11
4 Introduction to Glass-type AR/MR Devices	13
4.1 General	13
4.2 Device Functional Architecture.....	14
4.2.1 Device Functions	14
4.2.2 Generic reference device functional structure device types.....	16
4.2.2.1 Overview.....	16
4.2.2.2 Type 1: 5G STandalone AR (STAR) UE.....	17
4.2.2.3 Type 2: 5G EDGe-Dependent AR (EDGAR) UE.....	18
4.2.2.4 Type 3: 5G WireLess Tethered AR UE	19
4.2.3 AR Runtime.....	21
4.2.4 Scene Manager.....	22
4.2.5 XR Spatial Computing.....	22
4.2.6 5G Media Access Function.....	24
4.3 Basic Processes for delivering an AR experience	25
4.3.1 Introduction.....	25
4.3.2 AR Media Delivery Pipeline.....	26
4.3.3 XR Spatial Compute Pipeline	28
4.4 AR content formats and codecs	29
4.4.1 Overview	29
4.4.2 Scene Graph and Scene Description	30
4.4.3 Metadata	30
4.4.3.1 User pose information.....	30
4.4.3.2 Camera Parameters.....	31
4.4.4 Media Formats/Primitives in AR Scenes	31
4.4.5 Compression Formats	32
4.4.5.1 Elementary stream.....	32
4.4.5.2 Storage and Delivery Formats.....	33
4.4.6 Multiple Media Decoders management and coordination	33
4.4.7 XR Spatial Description	34
4.4.7.1 Overview	34
4.4.7.2 Camera and sensor information.....	35
4.4.7.3 Visual features, Keyframes and Spatial Maps.....	35
4.4.7.4 Spatial Anchors and Trackables	37
4.5 Key Performance Indicators and Metrics for AR	37
4.5.1 Summary of TR 26.928	37
4.5.2 Updated KPIs for AR.....	39
4.5.3 Typical Latencies in networked AR Services	41
4.6 Related Work.....	42
4.6.1 3GPP.....	42
4.6.2 MPEG	43
4.6.3 ETSI Industry Specification Group	44
4.6.4 Work related to AR Runtime	46

4.6.4.1	OpenXR	46
4.6.4.2	WebXR	48
4.6.5	MPEG Scene Description	49
4.6.6	MPEG-I Video Decoding Interface for Immersive Media.....	50
4.6.7	MPEG-I Carriage of Point Cloud Compression Data	50
4.6.8	Web Real-Time Communication (WebRTC)	51
4.6.8.1	WebRTC as an OTT application	51
4.6.8.2	WebRTC framework for RTC Media Service Enablers.....	52
4.6.9	Joint workshop with Khronos and MPEG on "Streamed Media in Immersive Scene Descriptions"	52
5	Core Use Cases.....	53
6	Mapping to 5G System Architecture.....	54
6.1	General	54
6.2	Immersive media downlink streaming	55
6.2.1	Introduction.....	55
6.2.2	Relevant use cases	55
6.2.3	Architectures.....	56
6.2.3.1	STAR-based	56
6.2.3.2	EDGAR-based	56
6.2.4	Procedures and call flows	57
6.2.4.1	STAR-based media streaming.....	57
6.2.4.2	EDGAR-based media streaming	59
6.2.5	Content formats and codecs	60
6.2.6	KPIs and QoS	61
6.2.7	Standardization areas	62
6.3	Interactive immersive services	62
6.3.1	Introduction.....	62
6.3.2	Relevant use cases	62
6.3.3	Architectures.....	63
6.3.3.1	STAR-based	63
6.3.3.2	EDGAR-based	63
6.3.4	Procedures and call flows	63
6.3.4.1	STAR-based interactive immersive service	63
6.3.4.2	EDGAR-based interactive immersive service.....	66
6.3.5	Content formats and codecs	67
6.3.6	KPIs and QoS	67
6.3.7	Standardization areas	69
6.4	5G cognitive immersive service	69
6.4.1	Introduction.....	69
6.4.2	Relevant use cases	69
6.4.3	Architectures.....	70
6.4.3.1	STAR-based	70
6.4.3.2	EDGAR-based	70
6.4.4	Procedures and call flows	70
6.4.6	KPIs and QoS	73
6.5	AR conversational services	74
6.5.1	Introduction.....	74
6.5.2	Relevant use cases	75
6.5.3	Basic architecture and call flows	75
6.5.3.1	Symmetrical case	77
6.5.3.2	Asymmetrical case	77
6.5.4	Instantiation #1: MTSI-based architecture extension.....	78
6.5.5	Instantiation #2: DCMTSI-based architecture extension with immersive media processing	81
6.5.6	Content formats and codecs	84
6.5.7	Summary of AR conversational instantiations.....	84
6.5.8	Standardization areas	85
6.6	Shared AR conversational experience	85
6.6.1	Introduction.....	85
6.6.2	Relevant use cases	86
6.6.3	Basic architecture.....	86
6.6.4	Generic Call flow.....	87

6.6.5	Various instantiations	89
6.6.6	Content formats and codecs	90
6.6.7	Standardization areas	90
7	Considerations on Devices Form-factor	90
7.1	General	90
7.2	Battery/Power consumption	91
7.3	Camera	91
7.4	Display	91
7.5	Heat dissipation	92
7.6	Weight	92
8	Potential New Work and Study Area	93
8.1	General	93
8.2	5G Generic Architectures for Real-Time Media Delivery	93
8.3	5G-Media Service Enablers.....	94
8.4	5G Real-time Communication.....	95
8.5	Media Capabilities for Augmented Reality Glasses	95
8.6	Split Rendering Media Service Enabler with AR profile	96
8.7	Tethering AR Glasses.....	97
8.8	IMS-based AR conversational services	97
8.9	Audio Media Pipelines for AR Experiences.....	98
9	Conclusions	98
Annex A:	Collection of Glass-type AR/MR Use Cases	100
A.1	Use Case 16: AR remote cooperation	100
A.2	Use Case 17: AR remote advertising	102
A.3	Use Case 18: Streaming of volumetric video for glass-type MR devices	104
A.3.1	Use case description	104
A.3.2	Call flow for STAR UE.....	107
A.3.3	Call flow for EDGAR UE	109
A.4	Use Case 19: AR Conferencing.....	111
3.2.1	AR Conferencing (1:1)	111
3.2.1	AR Conferencing (1:many).....	111
A.5	Use Case 20: AR IoT control	112
A.6	Use Case 21: AR gaming	114
A.7	Use Case 22: Shared AR Conferencing Experience.....	116
Annex B:	Change history	118
History		119

Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
 - 1 presented to TSG for information;
 - 2 presented to TSG for approval;
 - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

Augmented Reality (AR) and Mixed Reality (MR) promise to provide new experiences for immersive media services. The form factors of the devices for these services are typically not expected to deviate significantly from those of typical glasses, resulting in less physical space for the various required components such as sensors, circuit boards, antennas, cameras, and batteries, when compared with typical smartphones. Such physical limitations also reduce the media processing and communication capabilities that may be supported by AR/MR devices, in some cases requiring the devices to offload certain processing functions to a tethered device and/or a server.

This report addresses the integration of such new devices into 5G system networks and identifies potential needs for specifications to support AR glasses and AR/MR experiences in 5G.

The focus of this document is on general system aspects, especially targeting visual rendering on glasses, and may not be equally balanced or equally precise on all media types (e.g. on haptics, GPUs, audio). For example, extrapolations on architectural aspects derived for primarily visual media pipelines to audio pipelines may require confirmation based on further study.

1 Scope

The present document collects information on glass-type AR/MR devices in the context of 5G radio and network services. The primary scope of this Technical Report is the documentation of the following aspects:

- Providing formal definitions for the functional structures of AR glasses, including their capabilities and constraints,
- Documenting core use cases for AR services over 5G and defining relevant processing functions and reference architectures,
- Identifying media exchange formats and profiles relevant to the core use cases,
- Identifying necessary content delivery transport protocols and capability exchange mechanisms, as well as suitable 5G system functionalities (including device, edge, and network) and required QoS (including radio access and core network technologies),
- Identifying key performance indicators and quality of experience factors,
- Identifying relevant radio and system parameters (required bitrates, latencies, loss rates, range, etc.) to support the identified AR use cases and the required QoE,
- Providing a detailed overall power analysis for media AR related processing and communication.

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 TR 26.928: "Extended Reality (XR) in 5G"
- [3] Wireless Broadband Alliance, "5G and Wi-Fi RAN Convergence", April 2021.
- [4] Khronos Group, The OpenXR Specification, 1.0, <https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html>
- [5] W3C, WebXR Device API, W3C Working Group Draft, <https://www.w3.org/TR/webxr/>
- [6] ISO/IEC 23090-13:2022 DIS: "Information technology — Coded representation of immersive media — Part 13: Video Decoding Interface for Immersive Media"
- [7] Microsoft Azure Kinect™ DK documentation, <https://docs.microsoft.com/en-us/azure/kinect-dk/>
- [8] Google ARCore™: Use Depth in your Android app, <https://developers.google.com/ar/develop/java/depth/developer-guide>
- [9] Microsoft Azure Kinect™ DK documentation: Use Azure Kinect Sensor SDK image transformations, <https://docs.microsoft.com/en-us/azure/kinect-dk/use-image-transformation#overview>

- [10] Daniel Wagner, Louahab Noui, Adrian Stannard, "Why is making good AR displays so hard?", LinkedIn Blog, August 7, 2019, <https://www.linkedin.com/pulse/why-making-good-ar-displays-so-hard-daniel-wagner/>
- [11] Daniel Wagner, "MOTION TO PHOTON LATENCY IN MOBILE AR AND VR", Medium Blog, August 20, 2018, <https://medium.com/@DAQRI/motion-to-photon-latency-in-mobile-ar-and-vr-99f82c480926>
- [12] Yodayoda, "Why loop closure is so important for global mapping", Medium Blog, December 24, 2020, <https://medium.com/yodayoda/why-loop-closure-is-so-important-for-global-mapping-34ff136be08f>
- [13] 3GPP TS 22.261: "Service requirements for the 5G system"
- [14] 3GPP TR 22.873: "Study on evolution of the IP Multimedia Subsystem (IMS) multimedia telephony service"
- [15] 3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia telephony; Media handling and interaction"
- [16] 3GPP TR 38.838: "Study on XR (Extended Reality) evaluations for NR"
- [17] ISO/IEC 23090-2:2021: "Information technology — Coded representation of immersive media — Part 2: Omnidirectional media format"
- [18] ISO/IEC 23090-3:2021: "Information technology — Coded representation of immersive media — Part 3: Versatile video coding"
- [19] ISO/IEC 23090-5:2021: "Information technology — Coded representation of immersive media — Part 5: Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)"
- [20] ISO/IEC 23090-8:2020: "Information technology — Coded representation of immersive media — Part 8: Network based media processing"
- [21] ETSI GS ISG ARF 003 v1.1.1 (2020-03): "Augmented Reality Framework (ARF) AR framework architecture"
- [22] Khronos Group, The GL Transmission Format (glTF) 2.0 Specification, <https://github.com/KhronosGroup/glTF/tree/master/specification/2.0/>
- [23] ISO/IEC 23090-14:2022: "Information technology — Coded representation of immersive media — Part 14: Scene Description for MPEG-I Media"
- [24] ISO/IEC 23090-10:2021: "Information technology — Coded representation of immersive media — Part 10: Carriage of Visual Volumetric Video-based Coding Data"
- [25] ISO/IEC 23090-18:2021: "Information technology — Coded representation of immersive media — Part 18: Carriage of Geometry-based Point Cloud Compression Data"
- [26] 3GPP TS 26.501: "5G Media Streaming (5GMS); General description and architecture"
- [27] H. Chen, Y. Dai, H. Meng, Y. Chen and T. Li, "Understanding the Characteristics of Mobile Augmented Reality Applications," 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2018, pp. 128-138.
- [28] S. Kang, H. Choi, "Fire in Your Hands: Understanding Thermal Behavior of Smartphones", The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)
- [29] T. Chihara, A. Seo, "Evaluation of physical workload affected by mass and center of mass of head-mounted display", Applied Ergonomics, Volume 68, pp. 204-212, 2018
- [30] Google Draco: <https://google.github.io/draco/>
- [31] T.Ebner, O.Schreer, I. Feldmann, P.Kauff, T.v.Unger, "m42921 HHI Point cloud dataset of boxing trainer", MPEG 123rd meeting, Ljubljana, Slovenia

- [32] Scene understanding, <https://docs.microsoft.com/en-us/windows/mixed-reality/scene-understanding>
- [33] Serhan Gül, Dimitri Podborski, Jangwoo Son, Gurdeep Singh Bhullar, Thomas Buchholz, Thomas Schierl, Cornelius Hellge, “Cloud Rendering-based Volumetric Video Streaming System for Mixed Reality Services”, Proceedings of the 11th ACM Multimedia Systems Conference (MMSys'20), June 2020
- [34] Scene lighting: <https://docs.microsoft.com/en-us/azure/remote-rendering/overview/features/lights>
- [35] PBR material: <https://docs.microsoft.com/en-us/azure/remote-rendering/overview/features/pbr-materials>
- [36] Colour Material: <https://docs.microsoft.com/en-us/azure/remote-rendering/overview/features/color-materials>
- [37] S. N. B. Gunkel, H. M. Stokking, M. J. Prins, N. van der Stap, F.B.T. Haar, and O.A. Niamut, 2018, June. Virtual Reality Conferencing: Multi-user immersive VR experiences on the web. In Proceedings of the 9th ACM Multimedia Systems Conference (pp. 498-501). ACM.
- [38] Dijkstra-Soudarissanane, Sylvie, et al. "Multi-sensor capture and network processing for virtual reality conferencing." Proceedings of the 10th ACM Multimedia Systems Conference. 2019.
- [39] VRTogether, a media project funded by the European Commission as part of the H2020 program, <https://vrtogether.eu/>, November 2020.
- [40] MPEG131 Press Release: Point Cloud Compression – WG11 (MPEG) promotes a Video-based Point Cloud Compression Technology to the FDIS stage: <https://multimediacommunication.blogspot.com/2020/07/mpeg131-press-release-point-cloud.html>
- [41] 3GPP TR 23.701: “Study on Web Real Time Communication (WebRTC) access to IP Multimedia Subsystem (IMS); Stage 2”
- [42] 3GPP TR 23.706: “Study on enhancements to Web Real Time Communication (WebRTC) access to IP Multimedia Subsystem (IMS); Stage 2”
- [43] 3GPP TS 24.371: “Web Real-Time Communications (WebRTC) access to the IP Multimedia (IM) Core Network (CN) subsystem (IMS); Stage 3; Protocol specification”
- [44] IETF RFC 8831: WebRTC Data Channels
- [45] IETF RFC 8864: Negotiation Data Channels Using the Session Description Protocol (SDP)
- [46] IETF RFC 8827: WebRTC Security Architecture
- [47] ISO/IEC 23090-6:2021: “Information technology — Coded representation of immersive media — Part 6: Immersive media metrics”
- [48] 3GPP TR 26.926: “Traffic Models and Quality Evaluation Methods for Media and XR Services in 5G Systems”
- [49] Oscar Falmer: “AR Headsets Landscape”, 10th September 2021, <https://docs.google.com/spreadsheets/d/1aUO8nuXWCnL1xYnpe9tvSgCphifhMRLrFj1enayv0X8/edit#gid=0>
- [50] Oscar Falmer: “Mobile AR Features Landscape”, 20th September 2021, https://docs.google.com/spreadsheets/d/1S1qEyDRCqH_UkcSS4xVQLgcMSEpIu_mPtHjjsN02GNw/edit#gid=0
- [51] Microsoft, Coordinate Systems, <https://docs.microsoft.com/en-us/windows/mixed-reality/design/coordinate-systems>
- [52] Google WebRTC project update & Stadia review, https://www.youtube.com/watch?v=avtlQeaxd_I&t=438s

- [53] Joint MPEG/Khronos/3GPP Workshop on "Streamed Media in Immersive Scene Descriptions", September 29/30, 2021, <http://mpeg-sd.org/workshop.html>
- [54] MPEG Systems Output WG3 N21042 "Report of Joint Workshop on Streamed Media in Immersive Scene Descriptions", MPEG#136, October 2021, https://www.mpegstandards.org/wp-content/uploads/mpeg_meetings/136_OnLine/w21042.zip
- [55] 3GPP TS 23.501: "System architecture for the 5G System (5GS)"
- [56] 3GPP TS 26.260: "Objective test methodologies for the evaluation of immersive audio systems"
- [57] 3GPP TS 26.261: "Terminal audio quality performance requirements for immersive audio services"
- [58] Younes, Georges, et al. "Keyframe-based monocular SLAM: design, survey, and future directions." Robotics and Autonomous Systems 98 (2017): 67-88. <https://arxiv.org/abs/1607.00470>
- [59] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints". <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>
- [60] Herber Bay et. al., "SURF: Speeded Up Robust Features". <https://people.ee.ethz.ch/~surf/eccv06.pdf>
- [61] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," 2011 International Conference on Computer Vision, 2011, pp. 2564-2571, doi: 10.1109/ICCV.2011.6126544.
- [62] ETSI GS ISG ARF 004 v1.1.1 (2020-08): "Augmented Reality Framework (ARF) Interoperability Requirements for AR components, systems and services"
- [63] Microsoft Flight Simulator, <https://www.xboxachievements.com/news/news-34054-microsoft-flight-simulators-world-is-two-million-gigabytes-in-size.html>

3 Definitions, symbols and abbreviations

3.1 Definitions

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

5G AR/MR media service enabler: A 5G AR/MR media service enabler is supporting an AR/MR application to provide AR/MR experience using at least partially 5G System tools.

5G System (Uu): Modem and system functionalities to support 5G-based delivery using the Uu radio interface.

AR/MR Application: a software application that integrates audio-visual content into the user's real-world environment.

AR/MR content: AR/MR content consists of a scene with typically one or more AR objects and is agnostic to a specific service.

AR Data: Data generated by the AR Runtime that is accessible through API by an AR/MR application such as pose information, sensors outputs, and camera outputs.

AR Media Delivery Pipeline: pipeline for accessing AR scenes and related media over the network.

AR/MR object: An AR/MR object provides a component of an AR scene agnostic to a renderer capability.

AR Runtime: a set of functions that interface with a platform to perform commonly required operations such as accessing controller/peripheral state, getting current and/or predicted tracking positions, general spatial computing, and submitting rendered frames to the display processing unit.

Lightweight Scene Manager: A scene manager that is capable to handle a limited set of 3D media and typically requires some form of pre-rendering in a network element such as the edge or cloud.

Media Access Function: A set of functions that enables access to media and other AR related data that is needed in the scene manager or AR Runtime in order to provide an AR experience.

NOTE: In the context of this report, the Media Access function typically uses 5G system functionalities to access media.

Peripherals: The collection of sensors, cameras, displays and other functionalities on the device that provide a physical connection to the environment.

Scene Manager: a set of functions that supports the application in arranging the logical and spatial representation of a multisensorial scene with support of the AR Runtime.

Simplified Entry Point: An entry point that is generated by 5G cloud/edge processes to support offloading processing workloads from UE by lowering the complexity of the AR/MR content.

Spatial Computing: AR functions which process sensor data to generate information about the world 3D space surrounding the AR user.

XR Spatial Description: a data structure describing the spatial organisation of the real world using anchors, trackables, camera parameters and visual features.

XR Spatial Compute Pipeline: pipeline that uses sensor data to provide an understanding of the physical space surrounding the device and uses XR Spatial Description information from the network.

XR Spatial Compute server: an edge or cloud server that provides spatial computing AR functions.

XR Spatial Description server: a cloud server for storing, updating and retrieving XR Spatial Description.

3.2 Symbols

Void

3.3 Abbreviations

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

5GMS	5G Media Streaming
AAC	Advanced Audio Coding
AF	Application Function
AGW	Access GateWay
API	Application Programming Interface
AR	Augmented Reality
ARF	Augmented Reality Framework
AS	Application Server
ATIAS	Terminal Audio quality performance and Test methods for Immersive Audio Services
ATW	Asynchronous Time Warp
AVC	Advanced Video Coding
BLE	Bluetooth Low Energy
BMFF	Based Media File Format
BoW	Bag-Of-visual-Words
CAD	Computer Aided Design
CGI	Computer Generated Imagery
CMAF	Common Media Application Format
CoM	Centre of Mass
CPU	Central Processing Unit
CSCF	Call Session Control Function

DASH	Dynamic Adaptive Streaming over HTTP
DC	Data Channel
DCMTSI	Data Channel Multimedia Telephony Service over IMS
DIS	Draft International Standard
DoF	Degree of Freedom
DRX	Discontinuous Reception
DTLS	Datagram Transport Layer Security
EAS	Edge Application Server
EDGAR	EDGE-Dependent AR
EEL	End-to-End Latency
eMMTEL	Evolution of IMS Multimedia Telephony Service
EMSA	Streaming Architecture extensions for Edge processing
ERP	Equirectangular Projection
EVC	Essential Video Coding
FDIS	Final Draft International Standard
FFS	For Future Study
FLUS	Framework for Live Uplink Streaming
FoV	Field of View
FPS	Frame Per Second
G-PCC	Geometry-based Point Cloud Compression
GBR	Guaranteed Bit Rate
glTF	Graphics Library Transmission Format
GPS	Global Positioning System
GPU	Graphics Processing Unit
HDCA	High Density Camera Array
HEVC	High Efficiency Video Coding
HLS	HTTP Live Streaming
HMD	Head-Mounted Display
HOA	Higher-Order Ambisonics
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
ICE	Interactive Connectivity Establishment
IMS	IP Multimedia Subsystem
IMU	Inertial Measurement Unit
ISG	Industry Specification Group
ISOBMFF	ISO Based Media File Format
ITT4RT	Immersive Teleconferencing and Telepresence for Remote Terminals
IVAS	Immersive Voice and Audio Services
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LI	Lawful Interception
LIDAR	Light Detection And Ranging
LSR	Late Stage Reprojection
MAF	Media Access Function
MBS	Multicast and Broadcast Services
MIoT	Mobile Internet of Things
MMT	MPEG Media Transport
MPD	Media Presentation Description
MPR	Maximum Power Reduction
MR	Mixed Reality
MRF	Multimedia Resource Function
MSH	Media Session Handler
MTSI	Multimedia Telephony Service over IMS
NAT	Network Address Translation
NBMP	Network Based Media Processing
OHMD	Optical Head Mount Display
OMAF	Omnidirectional Media Format
OpenGL	Open Graphics Library
OTT	Over-The-Top
PBR	Physically-Based Rendering
PCC	Point Cloud Compression

PCF	Policy Control Function
PDU	Protocol Data Unit
PLY	PoLYgon file format
PRACK	Provisional Response Acknowledgement
RAN	Radio Access Network
RFC	Request For Comments
RP	Reference Point
RTC	Real-Time Communication
RTP	Real-time Transport Protocol
SDK	Software Development Kit
SDP	Session Description Protocol
SIFT	Scale Invariant Feature Transform
SID	Study Item Description
SIP	Session Initiation Protocol
SLAM	Simultaneous Localization And Mapping
SRTCP	Secure Real-time Transport Control Protocol
SRTP	Secure Real-time Transport Protocol
SSE	Server-Sent Events
STAR	STandalone AR
STUN	Session Traversal Utilities for NAT
SURF	Speeded Up Robust Features
TCP	Transmission Control Protocol
TLS	Transport Layer Security
ToF	Time of Flight
TPU	Tensor Processing Unit
TURN	Traversal Using Relays around NAT
UE	User Equipment
USB	Universal Serial Bus
V3C	Visual Volumetric Video-based Coding
V-PCC	Video-based Point Cloud Compression
VDE	Video Decoding Engine
VDI	Video Decoding Interface
VPU	Vision Processing Unit
VVC	Versatile Video Coding
WebRTC	Web Real-Time Communication
WLAR	WireLess tethered AR
WTAR	Wired Tethered AR
XHR	XML HTTP Request
XMPP	eXtensible Messaging and Presence Protocol
XR	eXtended Reality

4 Introduction to Glass-type AR/MR Devices

4.1 General

After a series of feasibility studies and normative work on Virtual Reality (VR), the feasibility study on eXtended Reality (XR) in 5G (FS_XR5G), documented in TR 26.928 [2], analysed the frameworks for eXtended Reality (XR) as a conceptual umbrella for representing the virtual, augmented, and mixed realities in a consistent fashion. This study, 5G glass-type Augmented Reality (AR) / Mixed Reality (MR) classified the XR devices into different form factors, including AR glasses, also referred to as optical head-mounted displays (OHMDs) and pointed out their power and tethering issues with the sidelink. A key aspect of this study is to identify the details of AR glasses including the capabilities for communication, processing, media handling, and offloading of power consumption.

Augmented reality composites virtual objects with reality. The compositing is a combination of light from real world and light presented on display to make them visible together to a user's eyes. Challenges in AR include the prediction of the real-world image visible to the human eye, and the accurate positioning and presentation of the virtual image on the display of AR glasses. To display a virtual image in front of a user's eyes, an optical see-through display is installed between the real world and the user's eyes, typically and most conveniently done with AR glasses.

In order to track the real-world space in which to place the virtual objects, sensors and in particular cameras are required to capture a live-action image visible to the human eye. Typically, multiple sensors and cameras are needed to construct a three-dimensional geometry around the user, called the user's geometry. The perception of geometry and the mapping of AR glass in geometry is referred to Simultaneous Localization and Mapping (SLAM), also introduced in some details in TR 26.928.

When AR objects are placed in the user's geometry, these objects are anchored to a part of the real-world geometry. With users moving, maintaining augmentation and consistency between reality and the user's geometry is challenging and requires continuous data flows and processing. In order to support devices with low power consumption, split processing such as split rendering or split perception are technologies to offload processing to powerful network servers. Such split processing approaches are considered beneficial or even essential for satisfying AR experiences, but add new necessary processes and challenges. This encoding, transmission, decoding, correction of rendering data and sensor/camera data over 5G networks, altogether pose challenges on bitrates, reliability requirements and latencies.

Based on the findings in clause 8 of TR 26.928, this clause follows up on some parts of the conclusions and proposed short term actions:

- Develop a flexible XR centric device reference architecture as well as a collection of device requirements and recommendations for XR device classes based on the considerations in clause 7.2 of TR 26.928.
- Study detailed functionalities and requirements for glass-type AR/MR UEs with standalone capabilities according to clause 7.6 of TR 26.928 and address exchange formats for AR centric media, taking into account different processing capabilities of AR devices.

Three different types of device reference architectures are identified in this clause. One major distinction among these types is the device capabilities of whether stand-alone processing of required AR media processing (in clause 4.2.2.2) or having dependencies on an entity in charge of offloading of power consuming processes, which the entity may be a cloud/edge service (in clause 4.2.2.3) and 5G wireless connectivity (in clause 4.2.2.4).

For the detailed functionalities for the device reference architecture of AR glasses, AR runtime (in clause 4.2.3) is identified for AR/MR system capability discovery, AR/MR session management, tracking of surrounding area, and rendering of AR/MR content in scene graph. Scene manager (in clause 4.2.4) is able to process a scene graph and render the corresponding 3D scene. 5G Media Access Function (in clause 4.2.4) is identified to support AR UE and the scene manager to access and stream components of AR content (in clause 4.4).

AR content consists of one or more AR objects in terms of primitives (in clause 4.4.4) and their spatial and temporal compositions described by a scene description (in clause 4.4.2). Processing of AR/MR functions may require additional metadata (in clause 4.4.3) to properly recognize user's pose and surrounding area.

Key performance indicators and metrics for AR/MR based on TR 26.928 are provided (in clause 4.5) and related works (in clause 4.6) on AR/MR in 3GPP, MPEG, and ETSI are identified for the considerations on collaborative work on device function architecture and AR content formats and codecs.

4.2 Device Functional Architecture

4.2.1 Device Functions

AR glasses contain various functions that are used to support a variety of different AR services as highlighted by the different use cases in clause 5. AR devices share some common functionalities to create AR/XR experiences. Figure 4.2.1-1 provides a basic overview of the relevant functions of an AR device.

The primary defined functions are

- **AR/MR Application:** a software application that integrates audio-visual content into the user's real-world environment.
- **AR Runtime:** a set of functions that interface with a platform to perform commonly required operations such as accessing controller/peripheral state, getting current and/or predicted tracking positions, general spatial computing, and submitting rendered frames to the display processing unit.
- **Media Access Function:** A set of functions that enables access to media and other AR related data that is needed in the scene manager or AR Runtime in order to provide an AR experience. In the context of this report, the Media Access function typically uses 5G system functionalities to access media.

- **Peripherals:** The collection of sensors, cameras, displays and other functionalities on the device that provide a physical connection to the environment.
- **Scene Manager:** a set of functions that supports the application in arranging the logical and spatial representation of a multisensorial scene based on support from the AR Runtime.

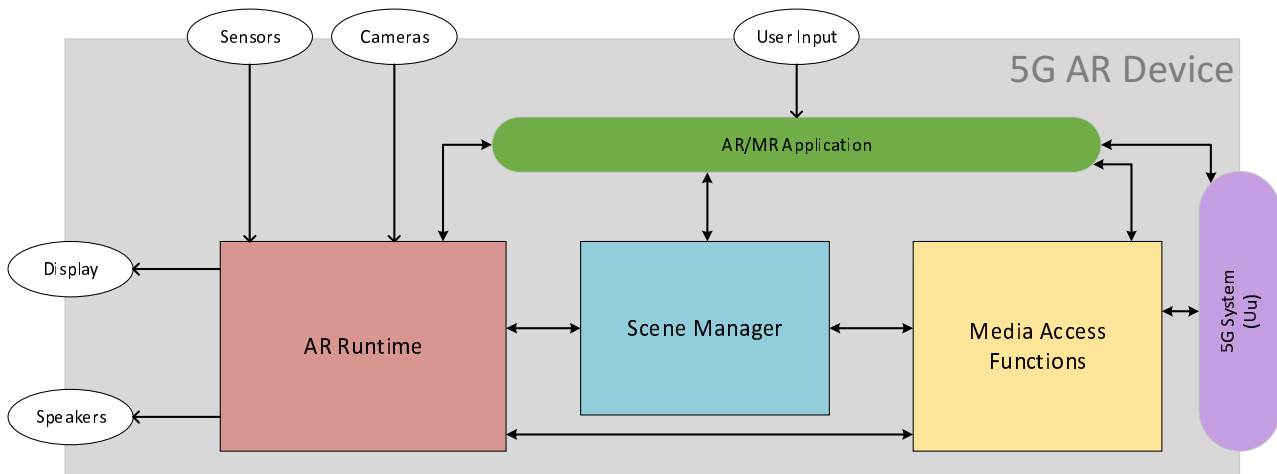


Figure 4.2.1-1: 5G AR Device Functions

The various functions that are essential for enabling AR glass-related services within an AR device functional structure include:

- Tracking and sensing (assigned to the AR Runtime)
 - Inside-out tracking for 6DoF user position
 - Eye Tracking
 - Hand Tracking
 - Sensors
- Capturing (assigned to the peripherals)
 - Vision camera: capturing (in addition to tracking and sensing) of the user's surroundings for vision related functions
 - Media camera: capturing of scenes or objects for media data generation where required

NOTE: Vision and media camera logical functions may be mapped to the same physical camera, or to separate cameras. Camera devices may also be attached to other device hardware (AR glasses or smartphone), or exist as a separate external device.

- Microphones: capturing of audio sources including environmental audio sources as well as users' voice.
- AR Runtime functions
 - XR Spatial Compute: AR functions which process sensor data to generate information about the world 3D space surrounding the AR user. It includes functions such as SLAM for spatial mapping (creating a map of the surrounding area) and localization (establishing the position of users and objects within that space), 3D reconstruction and semantic perception.
 - Pose corrector: function for pose correction that helps stabilise AR media when the user moves. Typically, this is done by asynchronous time warping (ATW) or late stage reprojection (LSR).
 - Semantic perception: process of converting signals captured on the AR glass into semantical concepts. Typically uses some sort of Artificial Intelligence (AI) and/or Machine Learning (ML). Examples include object recognition, object classification, etc.

- Scene Manager

- Scene graph handler: a function that supports the management of a scene graph that represents an object-based hierarchy of the geometry of a scene and permits interaction with the scene.
- Compositor: compositing layers of images at different levels of depth for presentation
- Immersive media renderer: the generation of one (monoscopic displays) or two (stereoscopic displays) eye buffers from the visual content, typically using GPUs. Rendering operations may be different depending on the rendering pipeline of the media, and may include 2D or 3D visual/audio rendering, as well as pose correction functionalities. Also includes rendering of other senses such as audio or haptics.

e) Media Access Function includes

- Tethering and network interfaces for AR/MR immersive content delivery
 - > The AR glasses may be tethered through non-5G connectivity (wired, WiFi)
 - > The AR glasses may be tethered through 5G connectivity
 - > The AR glasses may be tethered through different flavours of 5G connectivity
- Content Delivery: Connectivity and protocol framework to deliver the content and provide functionalities such as synchronization, encapsulation, loss and jitter management, bandwidth management, etc.
- Digital representation and delivery of scene graphs and XR Spatial Descriptions
- Codecs to compress the media provided in the scene.
 - > 2D media codecs
 - > Immersive media decoders: media decoders to decode compressed immersive media as inputs to the immersive media renderer. Immersive media decoders include both 2D and 3D visual/audio media decoder functionalities.
 - > Immersive media encoders: encoders providing compressed versions of visual/audio immersive media data.
- Media Session Handler: A service on the device that connects to 5G System Network functions, typically AFs, in order to support the delivery and QoS requirements for the media delivery. This may include prioritization, QoS requests, edge capability discovery, etc.
- Other media-delivery related functions such as security, encryption, etc.

f) Physical Rendering (assigned to the peripherals)

- Display: Optical see-through displays allow the user to see the real world “directly” (through a set of optical elements though). AR displays add virtual content by adding additional light on top of the light coming in from the real-world.
- Speakers: Speakers that allow to render the audio content to provide an immersive experience. A typical physical implementation are headphones.

g) AR/MR Application with additional unassigned functions

- An application that makes use of the AR and MR functionalities on the device and the network to provide an AR/MR user experience.

4.2.2 Generic reference device functional structure device types

4.2.2.1 Overview

In TR 26.928, clause 4.8, different AR and VR device types had been introduced. This clause provides an update and refinement in particular for AR devices. The focus in this clause mostly on functional components and not on physical implementation of the glass/HMD. Also, in this context the device is viewed as a UE, i.e. which functions are included in the UE.

A summary of the different device types is provided in Table 4.2.2.1-1. The table also covers:

- how the devices are connected to get access to information,
- where the 5G Uu modem is expected to be placed,
- where the AR Runtime (as specified in 4.2.1) is placed,
- where the Scene Manager (as specified in 4.2.1) is placed,
- where the AR/MR application is running,
- where the power supply/battery is placed.

Examples for current AR devices with assigned properties are for example provided in [49]. For all glass device types, it is assumed that sensors, cameras and microphones are on the device.

The definition for Split AR/MR in Table 4.2.2.1-1 is as follows:

- Split: the tethered device (phone/puck) or external entity (cloud/edge) does some power-intense processing (e.g., a pre-rendering of the viewport based on sensor and pose information), and the AR/MR device and/or tethered device performs post-processing considering the latest sensor information (e.g. warping to apply pose correction). Different degrees of split workflow exist, between different devices and entities. Similarly, vision engine functionalities and other AR/MR functions (such as AR/MR media reconstruction, encoding and decoding) may be subject to split computation.

Table 4.2.2.1-1: 5G Augmented Reality device types

Device Type Name	Reference	Tethering	5G Uu Modem	AR Runtime	Scene Manager	AR/MR Application	Power Supply
5G Standalone AR UE	1: STAR	N/A	Glass	Glass	Glass/Split ¹⁾	Glass	Glass
5G EDGe-Dependent AR UE	2: EDGAR	N/A	Glass	Glass/Split ¹⁾	Split ¹⁾	Split	Glass
5G WireLess Tethered AR UE	3: WLAR	802.11ad, 5G sidelink, etc.	Tethered device	Glass	Split ²⁾	Tethered device	Glass
5G Wired Tethered AR UE ³⁾	4: WTAR	USB-C	Tethered device	Tethered device	Split ²⁾	Tethered device	Tethered device
1) Cloud/Edge 2) Phone/Puck and/or Cloud/Edge 3) Not considered in this document							

The Wired Tethered AR UE device type is for reference purposes only and not considered in this document as it is not included as part of the study item objectives.

Generally, the STAR and WLAR device according to Table 4.2.2.1-1 are expected to have similar functionalities from a 5G System perspective.

Based on this, the focus is on three main different device types in the remainder of this document following the rows 1 to 3 in Table 4.2.2.1-1.

4.2.2.2 Type 1: 5G STandalone AR (STAR) UE

Figure 4.2.2.2-1 provides a functional structure for Type 1: 5G STandalone AR (STAR) UE.

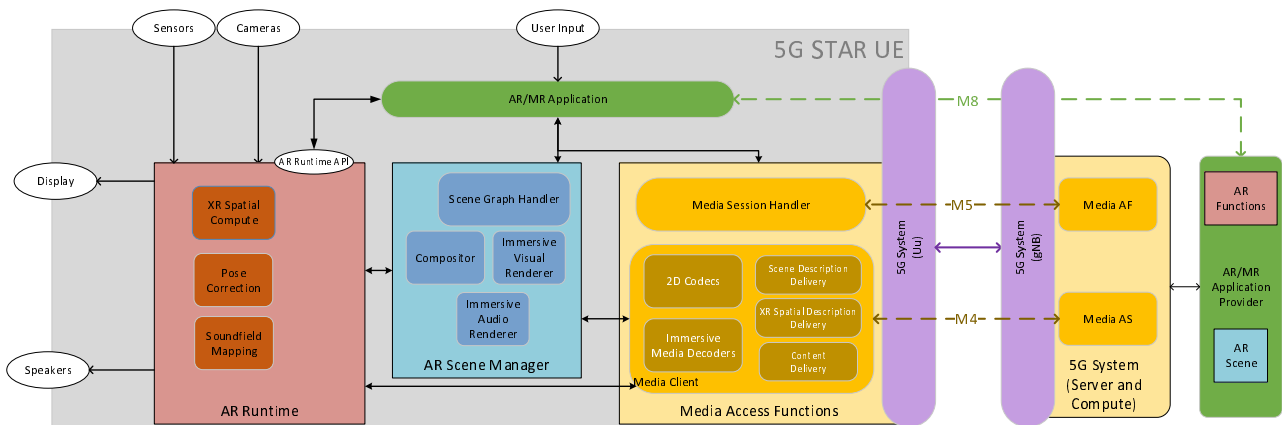


Figure 4.2.2.2-1: Functional structure for Type 1: 5G Standalone AR (STAR) UE

Main characteristics of Type 1: 5G Standalone AR (STAR) UE:

- The STAR UE is a regular 5G UE. 5G connectivity is provided through an embedded 5G modem.
- The AR Runtime is local and uses input from sensors, audio inputs or video inputs. XR Spatial Compute is primarily local, but may access or share information on the network.
- The AR Scene Manager is local and provides immersive rendering capabilities. Support of compute on the network may be provided, but scenes may typically be composed on the UE.
- The AR/MR application is resident on the device.
- An AR/MR application provider is providing a service and the service may be supported/assisted by network-based AR functions and rendering.
- Due to the amount of processing required, such devices are likely to require a higher power consumption in comparison to the other device types.
- As the device includes all UE functionalities, the application resides and pre-dominantly is executed on the device and all essential AR/MR functions are available for typical media processing use cases, the device referred to as Standalone AR (STAR) UE.
- Media Access Functions are provided that support the delivery of media content components over the 5G system. For details refer to clause 4.2.6.
- The application may also communicate through the 5G System using a dedicated interface.

4.2.2.3 Type 2: 5G EDGe-Dependent AR (EDGAR) UE

Figure 4.2.2.3-1 provides a functional structure for Type 2: 5G EDGe-Dependent AR (EDGAR) UE.

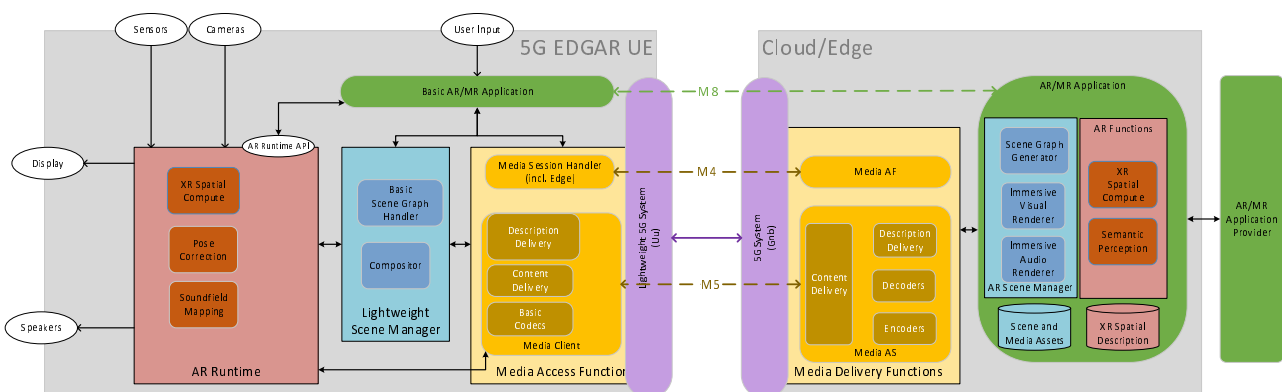


Figure 4.2.2.3-1: Functional structure for Type 2: 5G EDGe-Dependent AR (EDGAR) UE

Main characteristics of Type 2: 5G EDGE-Dependent AR (EDGAR) UE:

- The 5G EDGAR UE is a regular 5G UE. 5G connectivity is provided through an embedded 5G modem and 5G System components.
- The AR Runtime is local and uses data from sensors, audio inputs or video inputs. The AR Runtime, in particular the XR Spatial Compute, may be assisted by the cloud/edge application for example spatial localization and mapping provided by a spatial computing service.
- Media processing is local, the device needs to embed all media codecs required for decoding pre-rendered 2D view.
- A Lightweight Scene Manager is local to the AR/MR device, but the main scene management and composition is done on the cloud/edge. A scene description is generated and exchanged to establish the split work flow.
- The main AR/MR application resides on the cloud/edge, but a basic application functionality is on the UE to support regular UE functionalities and launching services and applications.
- Power consumption on such glasses must be low enough to fit the form factors. Heat dissipation is essential.
- Media Access Functions are provided that support the delivery of media content components over the 5G system, in particular cloud and split rendering supporting functions. Media Access Functions are divided in control on M5 (Media Session Handler and Media AF) and user data on M4 (Media Client and Media Application Server). Detailed requirements are for study in this report.
- While the EDGAR UE may have additional functionalities, for example those available in a STAR UE, generally for media centric or compute heavy use cases processing needs to be supported by the edge, hence referred to as EDGE-Dependent AR (EDGAR) UE.
- The application may also communicate through the 5G System using a dedicated interface.

4.2.2.4 Type 3: 5G WireLess Tethered AR UE

This clause introduces the 5G WireLess Tethered AR UE. Two sub-types are differentiated:

- Split Rendering WLAR UE. In this case the 5G phone that includes the modem also acts to support rendering of complex scenes and provides the pre-rendered data to the glass
- Relay WLAR UE: In this case, the 5G phone acts as a relay to provide IP connectivity.

Figure 4.2.2.4-1 provides a functional structure for Type 3a: 5G Split Rendering WireLess Tethered AR UE.

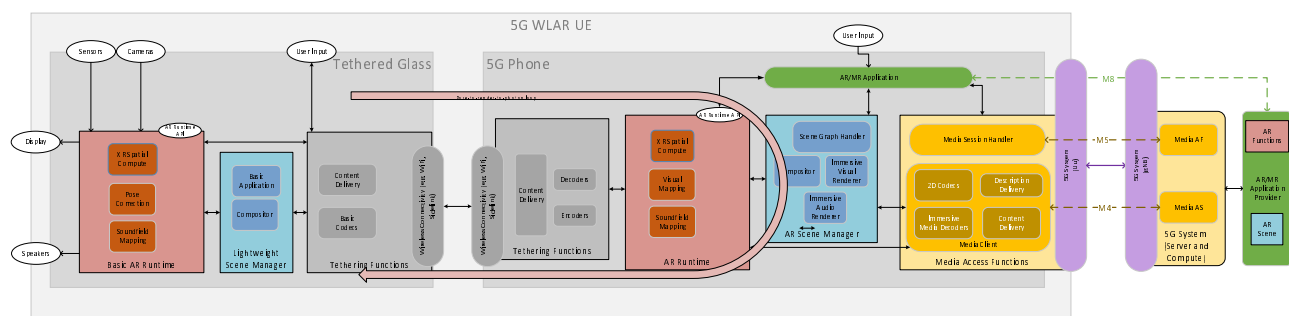


Figure 4.2.2.4-1: Functional structure for Type 3a: 5G Split Rendering WireLess Tethered AR UE

Main characteristics of Type 3a: 5G Split Rendering WireLess Tethered AR UE:

- 5G connectivity is provided through a tethered device which embeds the 5G modem. Wireless tethered connectivity is provided through WiFi or 5G sidelink. BLE (Bluetooth Low Energy) connectivity may be used for audio. The motion-to-render-to-photon loop runs from the glass to the phone. While the connectivity is outside of the 5G Uu domain, it is still expected that for proper performance when used for split rendering, a stable and constant delay link may be setup on the tethered connection.

- The AR Runtime is local and uses from sensors, audio inputs or video inputs, but may be assisted by functionalities on phone.
- While media processing (for 2D media) may be done on the AR glasses, energy intensive AR/MR media processing may be done on the AR/MR tethered device or split.
- Some devices might have limited support for immersive media decoding and rendering and may need to rely on 5G cloud/edge
- While such devices are likely to use significantly less processing than Type 1: 5G STAR devices by making use of the processing capabilities of the tethered device, they still support a lot of local media and AR/MR processing. Such devices are expected to provide 8-10h of battery life while keeping a significantly low weight.
- The tethered glass itself is not a regular 5G UE, but the combination of the glass and the phone results in a regular 5G UE.
- Media Access Functions are provided that support the delivery of media content components over the 5G system. Examples of the Media Access Functions are 5GMS functions, MTSI functions, web-connectivity or edge-related client functions. Detailed requirements are for study in this report.

Figure 4.2.2.4-2 provides a functional structure for Type 3b: 5G Relay WireLess Tethered AR UE.

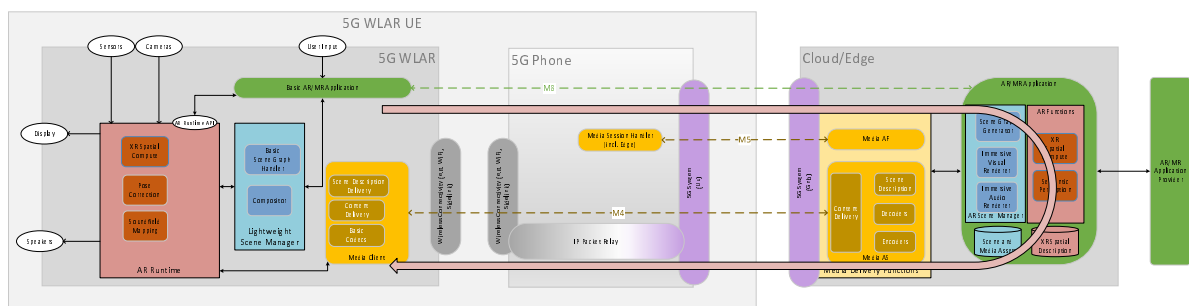


Figure 4.2.2.4-2: Functional structure for Type 3b: 5G Relay WireLess Tethered AR UE

Main characteristics of Type 3b: 5G Relay WireLess Tethered AR UE:

- 5G connectivity is provided through a tethered device which embeds the 5G modem. Wireless tethered connectivity is through WiFi or 5G sidelink. BLE (Bluetooth Low Energy) connectivity may be used for audio.
- The 5G Phone acts as a relay to forward IP packets. The 5G Phone runs a Media Session Handler including EDGE functionalities to support QoS control on the 5G System. To support proper end-to-end QoS, the media session handling needs to take into account the constraints of the tethering link to provide sufficient QoS on the 5G System link to provide adequate QoE for the end user. Details on the exact function of the relay, for example of it is on IP layer (layer 3) or on lower layer is for further study.
- Media Access functions are provided on the glass device to support the delivery of media content components over the 5G and wireless tethered link.
- The motion-to-render-to-photon loop runs from the glass to the edge and hence includes in total 4 wireless links. It is expected that for proper performance when used for split rendering, a stable and constant delay end to end link needs to be setup.
- The AR Runtime is local and uses from sensors, audio inputs or video inputs, but may be assisted by functionalities on phone.
- Media Processing is either done on the glass device or it is split with the network. In particular, relevant is that many devices have limited support for immersive media decoding and rendering and may need to rely on 5G cloud/edge.
- While such devices are likely to use significantly less processing than Type 1: 5G STAR devices by making use of the processing capabilities of the tethered device, they still support a lot of local media and AR/MR processing. Such devices are expected to provide 8-10h of battery life while keeping a significantly low weight.

- The tethered glass itself is not a regular 5G UE, but the combination of the glass and the phone results in a regular 5G UE.
- For services with low latency requirements, such as MTSI or those provided by FLUS, it may be necessary to take the status of wireless connectivity into account when configuring the services, such that the link between AR glass and 5G phone is not overly loaded. Although some work on the convergence of different access networks is defined in [3], the coordination of the operation of Uu and wireless connectivity in such services is FFS.

A key challenge for WLAR and WTAR UEs is to properly estimate the required QoS allocations for the AR sessions. The QoS allocation must take into account the wireless/wired tethering link from the glass to the UE. This applies to all QoS parameters, namely bitrate, packet loss, delay, and jitter. The following diagram depicts a breakdown of the components contributing to the end-to-end delay as an example:

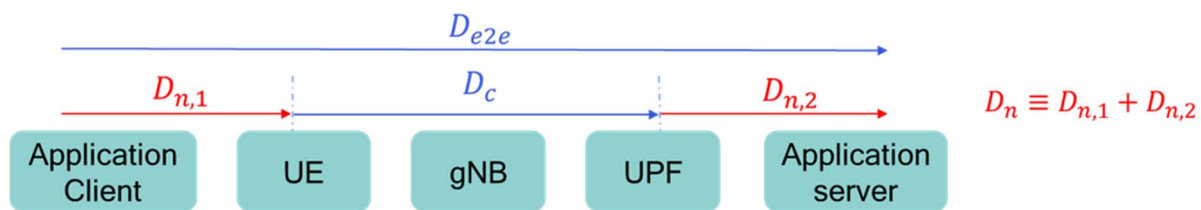


Figure 4.2.2.4-3: End-to-end delay breakdown to components

For a smooth operation of the AR session, the UE must estimate the impact of the tethering link on the overall QoS requirements. This corresponds to the $D_{n,1}$ component in the example figure. The MSH on the UE is the best entity to perform such estimates, which it may do by:

- Running some measurement tests for latency, packet loss, and bitrate
- Exchanging information with the radio and/or AF on the QoS policy

The MSH may regularly adjust its QoS allocation based on the observation of the status of the tethering link, thus targeting a consistent end-to-end QoS experience.

4.2.3 AR Runtime

The AR Runtime is a device-resident software or firmware that implements a set of APIs to provide access to the underlying AR/MR hardware. These APIs are referred to as AR Runtime APIs. An AR Runtime typically provides the following functions:

- System capability discovery: allows applications to discover capabilities of the AR glasses
- Session management: manages an AR session and its state
- Input and Haptics: receives information about user's actions, e.g. through usage of trackpads, and passes that information to the application. On request by the application, it may trigger haptics feedback using the AR glasses and associated hardware.
- Rendering: synchronizes the display and renders the composited frame onto the AR glasses displays.
- Spatial Computing: processes sensor data to generate information about the world 3D space surrounding the AR user. Spatial computing includes functions such as
 - > Tracking to estimate the movement of the AR device at a high frequency
 - > Relocalization to estimate the pose of the AR device at initialization, when tracking is lost or regularly to correct the drift of the tracking.
 - > Mapping, for reconstructing the surrounding space, for example through triangulation of identified points. This reconstruction may be sparse for localization purposes, or dense for visualization.

- > A combination of tracking, mapping and relocalization functions, for example through Simultaneous Localization and Mapping (SLAM) to build a map of the environment and establish the position of users and objects within that environment
- > Semantic perception: to process the captured information into semantical concepts, typically uses some sort of Artificial Intelligence (AI) and/or Machine Learning (ML). Examples include object or user activity segmentation, recognition, and classification.

Spatial computing functions typically include data exchange and requires network architecture. Clause 4.2.5 provides more details on XR Spatial computing.

AR runtimes are usually extensible to add support for a wide range of AR glasses and controllers that are on the market or that might be released in the future. This will allow different vendors to add custom functionality such as gaze tracking, hand control, new reference spaces, etc.

Two key representative and standardized AR runtimes are Khronos defined OpenXR [4] and W3C defined WebXR [5]. More details are provided in clause 4.6.4.

4.2.4 Scene Manager

A Scene Manager is a software component that is able to process a scene description and renders the corresponding 3D scene. The Scene Manager parses a scene description document to create a scene graph representation of the scene. For each node of the scene graph, it adds the associated media components for correct rendering of the corresponding object.

To render the scene, the Scene Manager typically uses a Graphics Engine that may be accessed by well-specified APIs such as defined by Vulkan, OpenGL, Metal, DirectX, etc. Spatial audio is also handled by the Scene Manager based on a description of the audio scene. Other media types may be added as well.

The Scene Manager is able to understand the capabilities of the underlying hardware and system, to appropriately adjust the scene complexity and rendering process. The Scene Manager may, for instance, delegate some of the rendering tasks to an edge or remote server. As an example, the Scene Manager may only be capable of rendering a flattened 3D scene that has a single node with depth and colour information. The light computation, animations, and flattening of the scene may be delegated to an edge server.

Clause 4.6.5 provides a description of glTF2.0 and the extensions defined by the MPEG-I Scene Description, which may serve as a reference for the entry point towards a Scene Manager.

4.2.5 XR Spatial Computing

XR Spatial computing summarizes functions which process sensor data to generate information about the world 3D space surrounding the AR user. It includes functions such as SLAM for spatial mapping (creating a map of the surrounding area) and localization (establishing the position of users and objects within that space), 3D reconstruction and semantic perception. This requires accurately localizing the AR device worn by the end-user in relation to a spatial coordinate system of the real-world space. Vision-based localization systems reconstruct a sparse spatial mapping of the real-world space in parallel (e.g. SLAM). Beyond the localization within a world coordinate system based on a sparse spatial map, additionally dense spatial mapping of objects is essential in order to place 3D objects on real surfaces, but also provides the ability to occlude objects behind surfaces, doing physics-based interactions based on surface properties, providing navigation functions or providing a visualization of the surface. Thirdly, for the purpose of understanding and perceiving the scene semantically, machine-learning and/or artificial intelligence may be used to provide context of the observed scene. The output of spatial computing is spatial mapping information that is organized in a data structure called the XR Spatial Description for storing and exchanging the information. Further details on XR Spatial Description formats are provided in clause 4.4.7.

XR Spatial Compute processes may be carried out entirely on the AR device. However, it may be beneficial or necessary to use cloud or edge resources to support spatial computing functions. At least two primary scenarios may be differentiated:

- 1) Spatial computing is done on the AR device, but an XR Spatial Description server is used for storage and retrieval of XR Spatial Description.
- 2) At least parts of the spatial compute functions are offloaded to a XR Spatial Compute server

Both cases are discussed further in the following.

Typically, the device stores a certain amount of XR Spatial Description locally on the device. However, in particular to create a world-experience, the AR device may not be able to store all information related to XR Spatial Description on the device, and hence, such information may continuously be updated by downloading or streaming updated information from an XR Spatial Description server as shown in Figure 4.2.5-1. In addition, the device may use personalized storage on the cloud to offload device-generated XR spatial information components. This may for example include so-called key frames, i.e. frames that are useful to provide accurate spatial mapping with relevant key points.

The architecture in Figure 4.2.5-1 relates to the case where XR Spatial computing is done standalone on the device, and hence we refer to this as STAR architecture in context of XR Spatial computing.

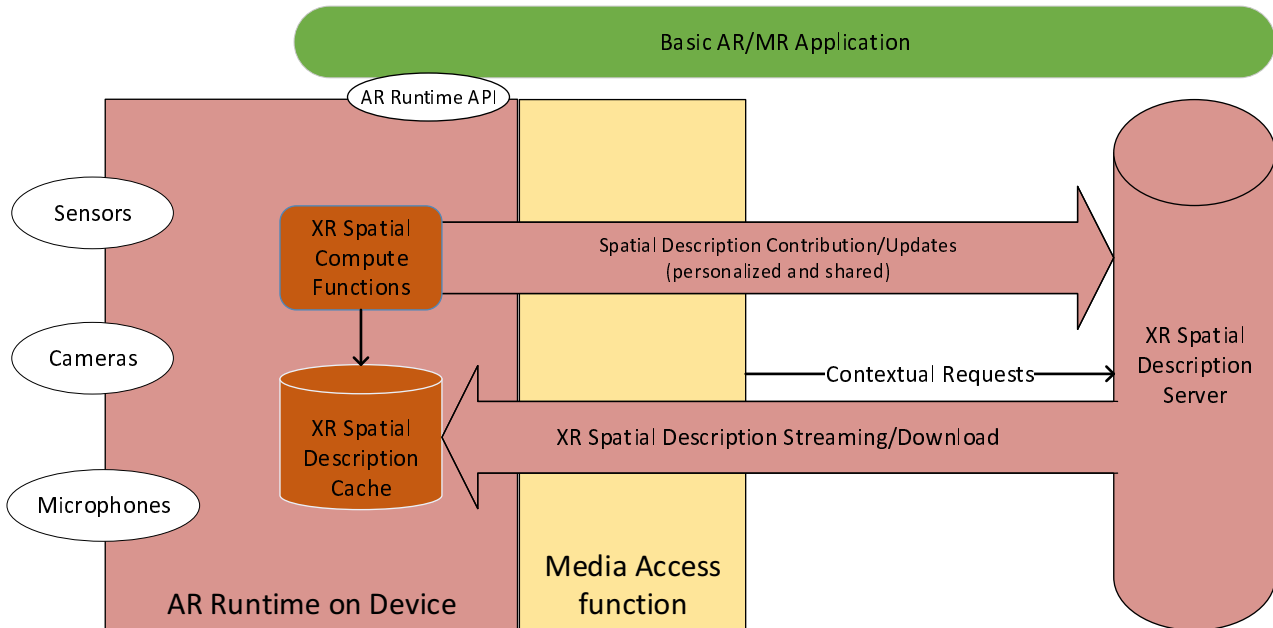


Figure 4.2.5-1 Functional diagram for XR Spatial computing with network/cloud support

If the device is limited in processing power, or if complex XR compute functionalities need to be carried out, the XR compute function on the device may be assisted by or even depend on compute resources in the network, for example on the edge or cloud. Figure 4.2.5-2 provide a basic architecture for the case where XR Spatial computing is delegated partially or completely to an XR Spatial computing edge server.

- The device sends sensor data or pre-processed sensor data (e.g. captured frames or visual features extracted from such frames) to the XR Spatial Compute server.
- The XR Spatial Compute server carries out supporting functions to extract relevant information and returns directly XR Spatial Compute-related AR Runtime data (according to the AR Runtime API), e.g. pose information, or pre-computed XR Spatial information that is used by a lightweight XR Spatial Compute function on the device to create AR Runtime data. Pre-computed XR Spatial information may, for example, be dense map of segmented objects for visualization, labels or id of recognized objects, 2D contours of recognized object to highlight them, or labels of the recognized user activity.
- The XR spatial edge compute server may further fetch the XR Spatial Description from the XR Spatial Description server and perform spatial computing based on device sensor data.

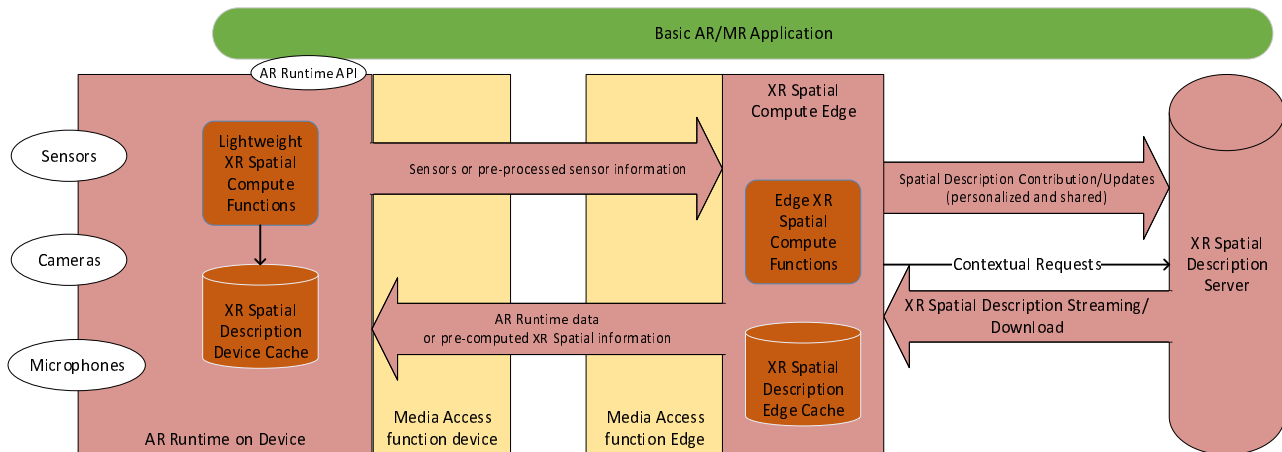


Figure 4.2.5-2 Functional diagram for spatial computing with XR Spatial Compute edge

The architecture in Figure 4.2.5-2 relates to the case for XR Spatial computing depends on an edge function, and hence we refer to this as EDGAR architecture in context of XR Spatial computing.

4.2.6 5G Media Access Function

The Media Access Function supports the AR UE to access and stream media. For this purpose, a Media Access Function as shown in Figure 4.2.6-1 includes:

- Codecs: used to compress and decompress the rich media. In several cases, not only a single instance of a codec per media type is needed, but multiple ones.
- Content Delivery Protocol: Container format and protocol to deliver media content between the UE and the network according to the requirements of the application. This includes timing, synchronization, reliability, reporting and other features.
- 5G connectivity: a modem and 5G System functionalities that allow the UE to connect to a 5G network and get access to the features and service offered by the 5G System.
- Media Session Handler: A generic function on the device to setup 5G System capabilities. This may setup edge functionalities, provide QoS support, support reporting, etc.
- Content protection and decryption: This function handles protection of content from being played on unauthorized devices.

Functions are needed in both uplink and downlink, depending on use cases and scenarios.

Example for Media Access Functions are

- 5GMSd client that includes a Media Session Handler and a Media Player as defined in TS 26.501 and TS 26.512.
- 5GMSu client that includes a Media Session Handler and a Media Streamer as defined in TS 26.501 and TS 26.512.
- A real-time communication client that includes either uplink or downlink, or both to support more latency critical communication services.
- A combination of the above based on the needs of the XR application. An XR scene may have a mix of static, streaming, and real-time media that require the usage of multiple transport channels and protocol stacks.

In all cases, the basic function of Media Session Handler and a delivery client (which includes content delivery protocols and codecs) is expected to be maintained. The Media Session Handler is a generic function to support 5G System integration.

As a subject of this report, the needs to support different types of instantiations is for codecs, delivery protocols, session handling and so is identified. Not all components are necessarily required for all scenarios and even further, not all functions may be available on all device types.

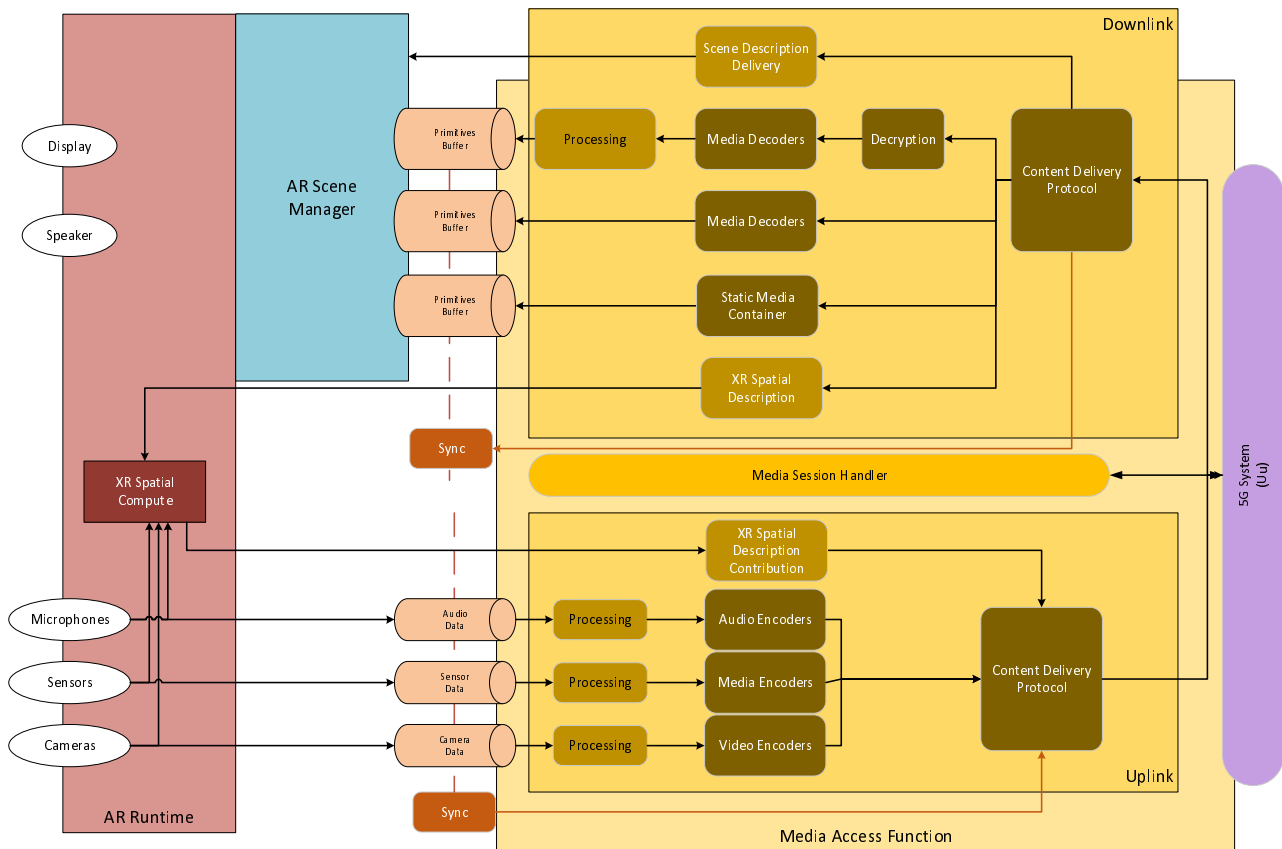


Figure 4.2.6-1 Media Access function for AR

4.3 Basic Processes for delivering an AR experience

4.3.1 Introduction

In this clause we provide a typical workflow for setting up a session between a 5G UE and network/cloud for receiving AR scenes from a scene provider. The basic workflow is provided in Figure 4.3.1-1. In this case, the following steps happen:

1. The application contacts the application provider to fetch the entry point for the content. The acquisition of the entry point may be performed in different ways and is considered out of scope. An entry point may for example be a URL to a scene description.
2. Session set up:
 - 2a. In case when the entry point is a URL of a scene description, the application initializes the Scene Manager using the acquired entry point.
 - 2b. The Scene Manager retrieves the scene description from the scene provider based on the entry point information.

NOTE: The scene provider can be mapped to a functional entity in the cloud/edge. Architecture mappings are described in clause 6.

- 2c. The Scene Manager parses the entry point and creates the immersive scene.
- 2d. The Scene Manager requests the creation of a local AR/MR session from the AR Runtime.

2e. The AR Runtime creates a local AR/MR session and performs registration with the local environment.

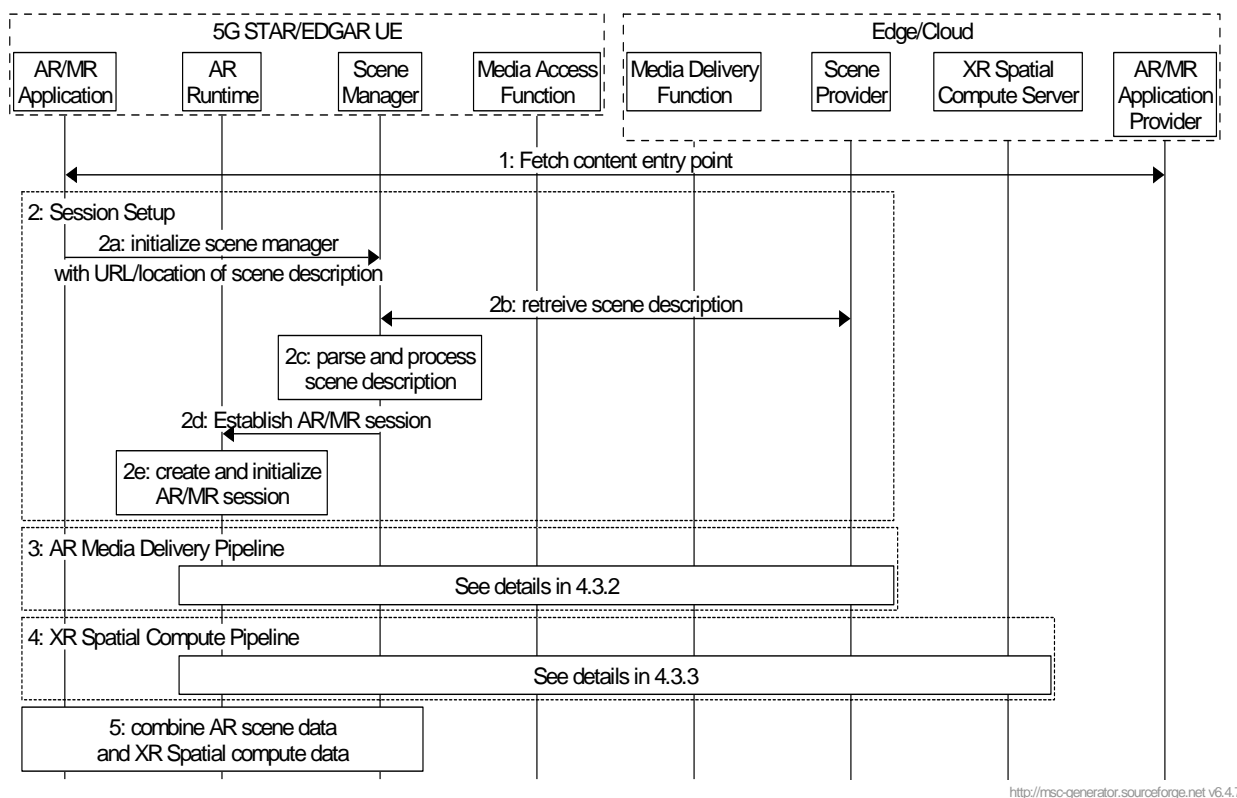
Then steps 3 and 4 run in parallel:

- 3: AR Media Delivery Pipeline: In case when entry point is a scene URL, a delivery session - for accessing scenes (new scenes or scene updates) and related media over the network is established. This can basically use the MAF as well as the scene manager and the corresponding network functions. Details are introduced in clause 4.3.2.

NOTE: The realization of AR media delivery pipeline may vary in different architectures as shown in clause 6.

- 4: XR Spatial Compute Pipeline: A pipeline that uses sensor data to provide an understanding of the physical space surrounding the device to determine the device's position and orientation and placement of AR objects in reference to the real world and uses XR Spatial Description information from the network to support this process. This uses the XR Spatial Description functions as introduced in clause 4.4.7. Details are introduced in clause 4.3.3.

- 5: Steps 3 and 4 run independently, but the results of both pipelines (e.g., media organized in a scene graph and pose of the AR device) are inputs of the AR/MR Scene Manager function. This function handles the common processing of the two asynchronous pipelines to create an AR experience.



<http://msc-generator.sourceforge.net/v6.4.7>

Figure 4.3.1-1: Basic workflow for delivering an AR experience

4.3.2 AR Media Delivery Pipeline

In this clause, we provide the detailed processes to set up AR Media Delivery Pipeline as addressed in step 3 of Figure 4.3.1-1. This generic basic process may be extended to address specific applications and use cases. The call flow as shown in Figure 4.3.2-1 aligns with the STAR/EDGAR architecture and serves as a baseline for defining use-case specific call flows.

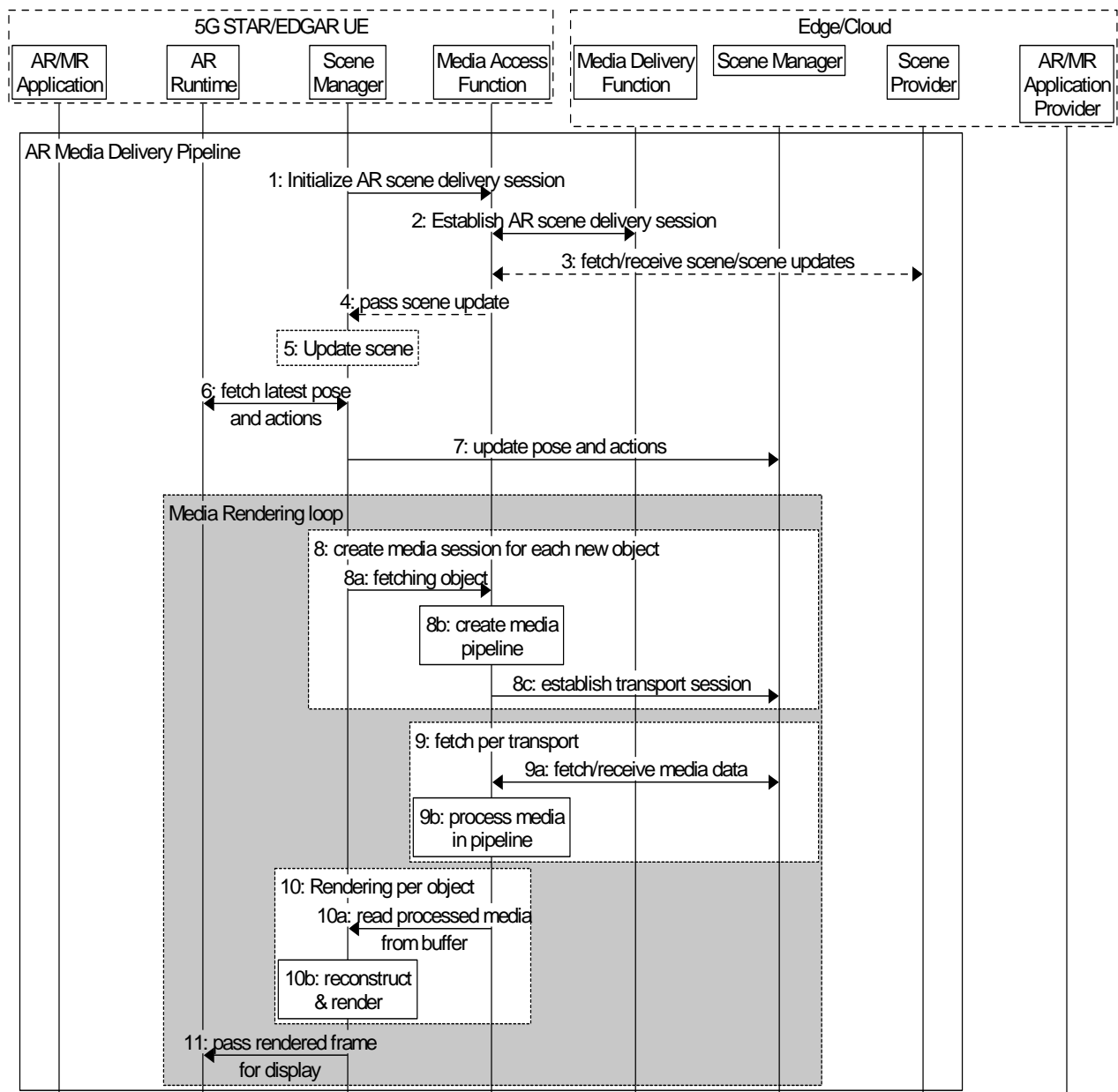


Figure 4.3.2-1: Functional diagram for AR Media Delivery Pipeline

For an AR Media Delivery Pipeline:

1. The Scene Manager initializes AR delivery session.
2. The MAF establishes AR delivery session.
3. The MAF may receive updates to the scene description from the scene provider
4. The MAF passes the scene update to the Scene Manager.
5. The Scene Manager updates the current scene.
6. The Scene Manager acquires the latest pose information and the user's actions
7. The Scene Manager in the device shares that information with the Scene Manager in edge/cloud

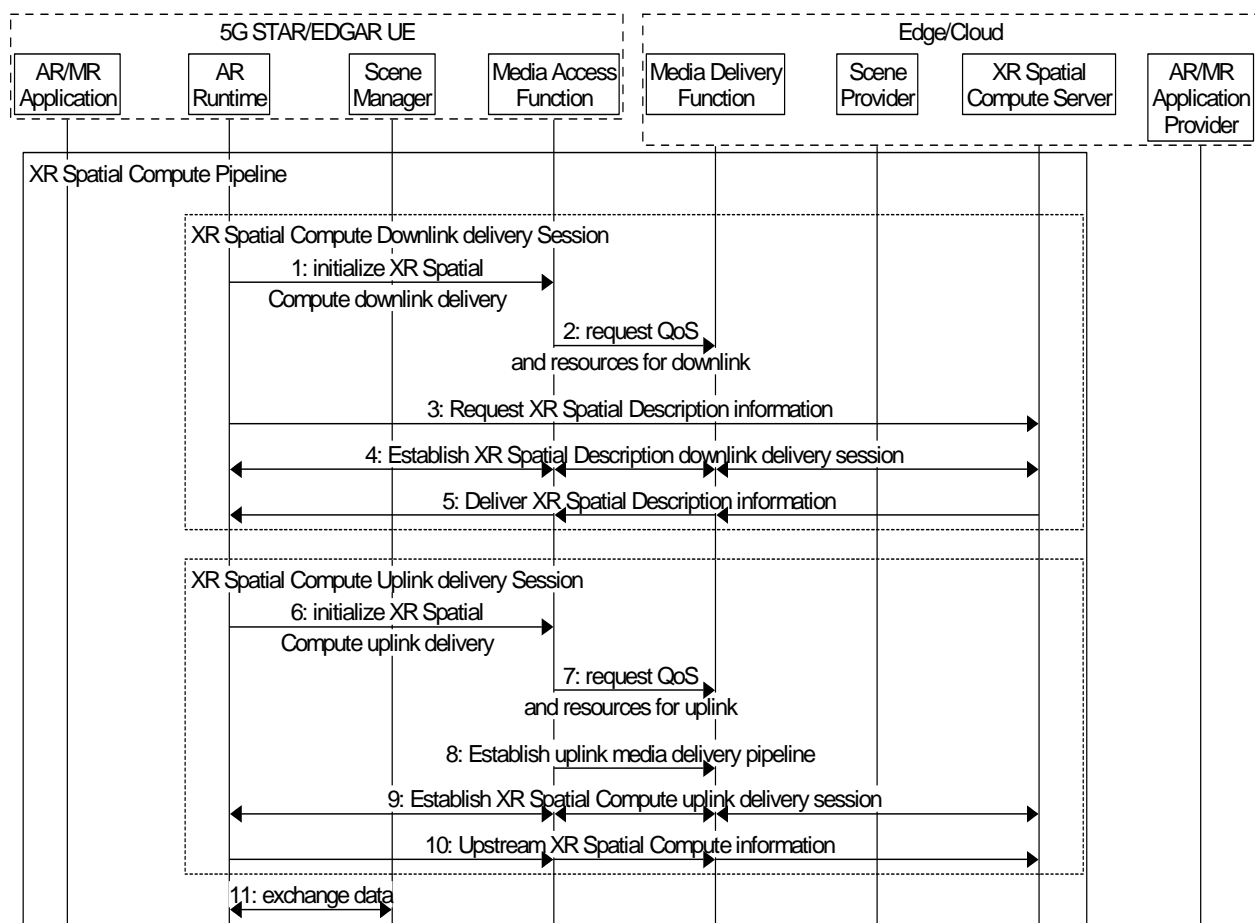
The media rendering loop consists of the following steps. Note that steps 8, 9 and 10 are running as 3 parallel loops:

8. For each new object in the scene:
 - a. The Scene Manager triggers the MAF to fetch the related media.

- b. The MAF creates a dedicated media pipeline to process the input.
 - c. The MAF establishes a transport session for each component of the media object.
9. For each transport session:
 - a. The media pipeline fetches the media data. It could be static, segmented, or real-time media streams.
 - b. The media pipeline processes the media and makes it available in buffers.
10. For each object to be rendered:
 - a. The Scene Manager gets processed media data from the media pipeline buffers
 - b. The Scene Manager reconstructs and renders the object
11. The Scene Manager passes the rendered frame to the AR/MR Runtime for display on the user's HMD.

4.3.3 XR Spatial Compute Pipeline

In this clause, we provide the detailed processes to set up an XR Spatial Compute Pipeline for spatial description related data as addressed in step 4 of Figure 4.3.1-1. This generic basic process may be extended to address specific applications and use cases. The call flow as shown in Figure 4.3.3-1 aligns with the STAR/EDGAR architecture and serves as a baseline for defining use-case specific call flows.



<http://msc-generator.sourceforge.net/v7.1>

Figure 4.3.3-1 Functional diagram for XR Spatial Compute Pipeline

For a XR Spatial Compute downlink delivery session:

1. The XR Spatial Compute function in the AR Runtime asks the MAF to establish a XR Spatial Compute downlink delivery session
2. The MAF communicates with the network to establish the proper resources and QoS

3. The XR Spatial Compute function requests access to XR Spatial Description information
4. An XR Spatial Description downlink delivery session is established across the XR Spatial Compute server, the media delivery function, the media access function and XR Spatial Compute function on the device.
5. XR Spatial Description information is delivered in this downlink delivery session

For a XR Spatial Compute uplink delivery session:

6. The XR Spatial Compute function in the AR Runtime asks the MAF to establish a XR Spatial Compute uplink delivery session
7. The MAF communicates with the network to establish the proper resources and QoS
8. The MAF established an appropriate uplink delivery pipeline
9. An XR Spatial Description uplink delivery session is established across the XR Spatial Compute function on the device, the media access function, the media delivery function and the XR Spatial Compute server.
10. Spatial compute information is upstreamed to the XR Spatial Compute server.
11. Data is continuously exchanged between the Scene Manager and the AR Runtime

4.4 AR content formats and codecs

4.4.1 Overview

A 5G AR/MR application provider offers an AR/MR experience to a 5G UE using an AR/MR application and 5G System and media functionalities. AR/MR content is typically agnostic to a delivery architecture and consists of one or more AR/MR objects, each of which usually corresponds to an immersive media type in clause 4.4.4 and may include their spatial and temporal compositions. The delivery of an immersive media adaptive to device capability and network bandwidth may be enabled by a delivery manifest in clause 4.4.5. Processing of AR/MR functions in 5GMS AS may require additional metadata in clause 4.4.3 to properly recognize user's pose and surroundings.

AR/MR functions include encoding, decoding, rendering and compositing of AR/MR object, after which localization and correction is performed based on the user's pose information.

STAR-based architecture has both basic AR functions and AR/MR functions on the device. EDGAR-based architecture has only basic AR functions on the device.

Since AR/MR functions are on-device for the STAR-based architecture, immersive media including 2D media is considered as the input media for the architecture.

Examples of immersive media are 2D/3D media such as overlay graphics and drawing of instructions (UC#16 in Annex A.1), 3D media such as furniture, a house and an animated representation of 3D modeled person (UC#17 in Annex A.2), a photorealistic volumetric video of a person (UC#18 in Annex A.3), a 3D volumetric representation of conference participants (UC#19 in Annex A.4), 2D video, and volumetric information and simple textual overlays (UC#20 in Annex A.5).

For the EDGAR-based architecture, basic AR functions are on-device therefore 2D media and additional information (such as depth map) generated from immersive media renderer are considered as the input media for basic AR functions. A rasterized and physically-based rendering (PBR) image is an example of 2D media.

A study into the existing technologies to be considered as inputs to each function and device type are identified and presented as a non-exclusive list below.

- several visual media representation formats were documented in clause 4.4.4.
- several delivery manifests were documented in clause 4.4.5.
- several scene description formats were documented in clause 4.4.2.
- metadata such as user pose information and camera information were documented in clause 4.4.3.

- management and coordination of multiple media decoders are documented in clause 4.4.6, respectively.

In order to integrate real-time media into AR scenes, a Media Access Function (MAF) provides the ability to access media and adds it to the AR scene. The MAF instantiates and manages Media Pipelines. A media pipeline typically handles content of an attribute/component of an object/mesh that is part of the scene graph. The media pipeline produces content in the format indicated by the scene description file. For real-time media, the formatted frame is then pushed into the circular buffer. Media Pipelines are typically highly optimized and customized for the type and format of media that is being fetched. Typically, for one scene, multiple media decoders of the same media type are needed to run in parallel. If the media decoders share the same hardware decoding platform on the UE, the MAF may also coordinate the different instances of media decoders to optimise the use of the hardware platform thus avoiding negative effects of resource competition or possible synchronization issues. MPEG-I Video Decoding Interface (ISO/IEC 23090-13 [6]) is an example specification that may fulfil this task of coordination. More information is available in clause 4.6.6. General considerations and challenges related to media decoder management is described in clause 4.4.6. Media Pipelines also maintain sync information (time and space) and passes that information as buffer metadata to the scene manager.

4.4.2 Scene Graph and Scene Description

A scene description may correspond to an AR/MR content. A volumetric media containing the primitives ranging from one vertex to a complex object may be described by a scene description. For the use cases listed in Table 5-1, scene description is useful to locate AR/MR objects in user's world. A scene description typically has a tree or a graph structure which of each leaf represents a component of a scene. A primitive or a group of primitives are referenced as a leaf node of the scene tree. A skeleton to allow for motion rigging or an animation of motion of the skeleton in time may present an animation of volumetric presentation.

- Formats for scene description

Khronos glTF2.0 and MPEG Scene description (ISO/IEC 23090-14) are examples of scene description technologies. They have a tree structure and internal/external resource references. There are many types of leaf of the tree. For example, a Node is one type of leaf under a Scene. A node may have a Camera as a subsidiary leaf. The node with camera represents one of the rendering frustum/viewport to be used by a scene renderer (i.e., immersive media renderer). Any translation/rotation/scaling of the node affects position and direction of its subsidiary, in this example, a camera. A node with mesh may be used as an anchor that represents AR object with its location and direction in geometric space.

MPEG Scene description is an extension of glTF2.0. It is extended to support MPEG immersive media. MPEG_media and MPEG_scene_description are the major changes to provide support of media access link including manifest, and temporal update of the scene description itself.

4.4.3 Metadata

4.4.3.1 User pose information

User's position may be represented as a geolocation with longitude and latitude. The position may also be represented as a point in a scene. The scene may be represented as a bounding box on a geometry which represents the user's real environment. When an AR/MR device reports the user position to obtain a split rendering of the immersive media from a server, the device calculating the user pose is expected to be either a geolocation, a point in a scene or a point in a user's geometry. Depending on the representation, the server is assumed to be aware of the underlying scene or the geometry. A device is expected to update whenever there is any change in the scene or the geometry through user interaction (e.g., rotating a scene by hand gesture) and/or SLAM (e.g., finer modelling of surrounding environment).

A direction may be represented with a rotation matrix, or roll, pitch, and yaw. The direction is relative to a scene/geometry and the scene/geometry has an origin and default direction of the three axes.

The device representing a user's pose moves continuously, and if the device is worn on the user's head, it is assumed that he or she frequently turns their head around. A set of position and direction information is only meaningful at a certain moment in time. Since the device reports the user pose at around a frequency of 1 KHz, any pose information would need to include a timestamp to specify when it was measured or created. A pose corrector (e.g., ATW and LSR) in a server may estimate the user's future pose, whilst a pose corrector in a device may correct the received rendered image to fit the latest user pose.

- Formats for user pose

A position in Cartesian coordinate system may be represented by either X, Y and Z or by a translation matrix. A direction may be represented by a rotation matrix or by quaternions.

OpenXR describes a possible format for user pose [4]. It consists of 4 quaternions for orientation and 3 vectors for position. Timestamp is represented by a 64 bit monotonically increasing nano-second-based integer.

4.4.3.2 Camera Parameters

Immersive media is captured by camera(s). The camera parameters such as focal length, principal points, calibration parameters and the pose of the camera all contribute in understanding the relevance between points in the volumetric scene and pixels in the captured image. Photogrammetry is the technology used to construct immersive media from a continuous capturing of images. Depth sensor-based cameras may be used to capture immersive media from one capturing of the volumetric scene

- Formats for camera information

Camera intrinsic parameters may be represented by a camera matrix. Extrinsic parameters may be represented by a transform matrix.

4.4.4 Media Formats/Primitives in AR Scenes

An AR/MR object may be represented in a form of 2D media. One camera or one view frustum in a scene may return a perspective planar projection of the volumetric scene. Such a 2D capture consists of pixels with colour attributes (e.g., RGB).

Each pixel (a) may represent a measure of the distance between the surface of an AR object, point (A) and the camera centre. Conventionally, the distance is represented by the coordinate of the point on the z-axis obtained by the orthogonal projection of the point (A) on this axis, here denoted as the point (A'). The measured distance is thus the length of the segment (CA') as depicted in Figure 4.4.4-1.

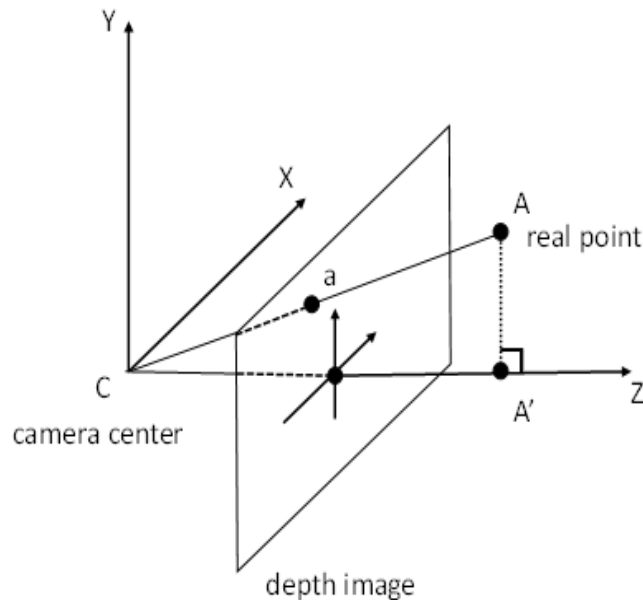


Figure 4.4.4-1: Pixel representation of depth images

This convention is used for commercially available frameworks handling depth images such as the Microsoft Azure Kinect™ SDK [7] and the Google ARCore™ [8]. According to the documentation of the Azure Kinect™ SDK, the depth sensor uses the Time-of-Flight (ToF) technique to measure the distance between the camera and a light-reflecting point in the scene. The documentation further specifies that “these measurements are processed to generate a depth map. A depth map is a set of Z-coordinate values for every pixel of the image, measured in units of millimeters”. Similarly, the Google ARCore™ documentation explains that “when working with the Depth API, it is important to understand that the depth values are not the length of the ray CA itself, but the projection of it” onto the z-axis.

Additionally, sensor API may provide the image from the viewpoint of the depth sensor which is thus not aligned with the viewpoint of RGB camera which is necessarily few millimetres away due to physical constraints. In this case, an alignment operation is necessary in order to guarantee the correspondence between a pixel of the depth image and a pixel of the RGB picture. For instance, the Azure Kinect SDK provides the `k4a_transformation_depth_image_to_color_camera()` and `k4a_transformation_color_image_to_depth_camera()` functions which generate a depth image aligned with the colour picture and a colour image aligned with the depth image, respectively. More details and illustrations are provided in [9].

A depth map thus contains pixels with the distance attribute (e.g., depth). Distance is one-dimensional information and may be represented in an absolute/relative or linear/non-linear manner. Metadata to explain the depth map may be provided.

The capturing of a volumetric scene may also be expressed as an omnidirectional image in a spherical coordinate system. Equirectangular Projection (ERP) is an example of projection methods to map a spherical coordinate system into a cylindrical coordinate system. The surface of the cylindrical coordinate system is considered as 2D media.

Capturing of a volumetric scene may be further improved/elevated with hundreds of cameras in an array; High Density Camera Array (HDCA) or lenticular are methods to capture rays of light. Each point on surface of a volumetric scene has countless rays of colours in multiple different directions. Each position of a camera captures a different colour from the same point surface of the volumetric scene. 2D images from the camera array may be packed together to form a larger plenoptic image.

From another perspective, 2D media is the output of the immersive media renderer. One view frustum that represents the user's viewport is placed in a scene, and in turn, a perspective or an orthogonal projection of the volumetric media may be produced. To minimise motion sickness, a pose corrector function performs a correction of the 2D media at the last stage of presentation. The pose corrector may require additional information such as the estimated or measured user pose that was used for the rendering of the 2D media. For the case that the latest user pose does not match with the estimated user pose, additional information that provides knowledge on the geometry, such as a depth map, may be delivered from immersive media renderer.

Immersive media may be considered as an AR/MR object and may be used to provide an immersive experience to users. The immersive experience may include a volumetric presentation of such media. The volumetric presentation does not bind to a specific display technology. For example, a mobile phone may be used to present either the whole AR media, or a part of the AR media. Users may see a volumetric presentation of a part of the AR media augmented in real space. Therefore, immersive media includes not only volumetric media formats such as omnidirectional visual format^{ERP image}, 3D meshes^{Primitives}, point clouds^{Primitives}, light fields^{Plenoptic image}, scene description, and 3D audio formats, but also 2D video^{2D image} as studied in TR 26.928.

- Formats for 2D media

Still image formats may be used for 2D media. The 2D media may have metadata for each image or for a sequence of images. For example, pose information describes the rendering parameter of one image. The frame rate or timestamp of each image are typically valid for a sequence of such images.

- Primitives

3D meshes and point clouds consists of thousands and millions of primitives such as vertex, edge, face, attribute and texture. Primitives are the very basic elements in all volumetric presentation. A vertex is a point in volumetric space, and contains position information in terms of three axes in coordinate system. In a Cartesian coordinate system, X, Y, and Z make the position information for a vertex. A vertex may have one or more attributes. Colour and reflectance are typical examples of attributes. An edge is a line between two vertices. A face is a triangle or a rectangle formed by three or four vertices. The area of a face is filled by interpolated colour of vertex attributes or from textures.

4.4.5 Compression Formats

4.4.5.1 Elementary stream

An elementary stream is an output of a media encoder. Immersive media and 2D media in clause 4.4.4 have relevant technologies to encode each media format as follows.

- 2D Video codecs

There are differences in terms of context of 2D media, such as RGB image versus depth map image, one planar perspective camera image versus ERP, or one camera image versus HDCA plenoptic image. Such differences may be considered in the proper encoder/decoder coding tools. In general, 2D video codecs may encode 2D media types listed in clause 4.4.4. AVC and HEVC are industry wide examples of 2D video codecs.

- MPEG OMAF (Omnidirectional Media Format)

OMAF consists of two parts; the first part is a pre-processing which includes a packing and projection of spherical volumetric media onto a 2D image, and the second part is an encapsulation of the compressed 2D frame packed image with metadata signalling the projection.

For the compression of the 2D images, 2D video codecs may be considered and the pre-processing operations are agnostic to specific 2D codec technology.

- MPEG V3C and V-PCC

V3C and V-PCC consists of two parts; the first part is a pre-processing which includes the decomposition of a part of the volumetric media into the planar projection, a patch, of different characteristics such as texture, geometry and occupancy. The second part is an encoding of 2D patch packing images, with metadata for signalling the decomposition.

For the encoding of the 2D images, 2D video codecs may be considered and the pre-processing operations are agnostic to specific 2D codec technology.

- MPEG G-PCC

G-PCC divides volumetric media into multiple sub-blocks. Triangle (Trisoup) or leaf (Octree) are used as the units of the divisions. A volumetric media is subdivided recursively until no more sub-blocks are left. The dimension (or level) of the tree is relatively large, such as 2^{24} . Tools including arithmetic encoding are used to encode all the tree information into the bitstream.

4.4.5.2 Storage and Delivery Formats

An encapsulation format encapsulates an elementary stream with its coding structure information and metadata information. ISO/BMFF (ISO based Media File Format, ISO/IEC 14496-12) is one of encapsulation format technology. DASH initialization/media segment and CMAF track are the extensions of ISO/BMFF for both adaptive streaming and storage purpose. They are extended to provide partial access of a media fragment on time axis.

A delivery manifest provides a description of media service consisting of multiple media components such as video and audio. Adaptation to device capability or network bandwidth is key features of a delivery manifest. In a delivery manifest, there is a group of multiple different encodings of the same media component context with the description of the encoding variations. An encapsulation format for an adaptive streaming is used to allow temporal access of media fragment to enable adaptive switching of a group of different encodings. MPD (Media Presentation Description) for DASH is one of delivery manifest for the purpose.

- File formats for Primitives

OBJ, PLY, and GPU command buffer in OpenGL-based languages (e.g., glTF Buffer) are methods of encapsulating the primitives. A sequence of primitive files – such as multiple OBJs, PLYs or a set of GPU command buffers in a time may present an animation of volumetric presentation.

4.4.6 Multiple Media Decoders management and coordination

The use of hardware video decoding platform is essential for the decoding of AR/MR content when it comes to power consumption, fast and scheduled decoding as well as battery usage. Modern hardware video decoding platform typically offer the capability to instantiate multiple decoders of the same media type at the same time and run multiple decoding instances in parallel. A typical example is the decoding of different components of the same AR/MR object, or the presentation of multiple objects in a scene. As a result, AR/MR application typically runs several decoder instances, in some cases using the same codec for different instances, in others different codecs for different streams. Note that this issue not only exists for video, but for any media type, in particular also for object-based audio. Under this high demand, there may be a resource competition and scheduled issues for the hardware decoding platform.

From an application perspective, there are different cases as well. There may exist cases for which even several applications are competing for the hardware decoding platform, for example an application renders a scene, but other applications provide overlays and notifications on top of the existing scene. A possible solution is to handle the coordination at the operating system level by setting priority to each application.

However, a single AR/MR application accessing and managing several decoding instances is a more typical and prominent case. It is thus important that the performance of the different decoder instances running is in line with the expectations and the needs of the AR/MR applications such that the AR/MR applications may optimise the usage of the hardware decoding platform when possible.

The first question from the AR/MR application point of view is to determine the number of decoder instances to instantiate. To this end, the AR/MR application may determine the number of AR/MR objects to be presented as well as the number of elementary streams contained in each AR/MR object. The hardware decoding platform is typically exposing a capability query API which lists the supported codec. This information enables the AR/MR application to calculate how many AR/MR objects may be simultaneously decoded and with which quality. In addition, there may be cases wherein different elementary streams from the same AR/MR object may be jointly decoded as part of a single elementary stream hence streamlining the rest of the pipeline by effectively decreasing the number of decoder instances and output buffers needed. When this is the case, the AR/MR application may instruct the hardware decoding platform to merge those input elementary streams.

At runtime, the AR/MR application expects the decoded frames for each AR/MR object to be ready at the same point in time so that further processing of this AR/MR object may happen without loss of frames or delay introduced due to buffering. However, the concurrent decoder instances may exhibit different performance in terms of decoding delay for each frame. Therefore, it is useful for the AR/MR application to be able to signal to the hardware video decoding platform that certain decoder instances form a group that expected to be treated collectively in terms of output synchronisation.

4.4.7 XR Spatial Description

4.4.7.1 Overview

XR Spatial Description is a data structure (typically organized in a graph) describing the spatial organisation of the real world using:

- Visual features, keyframes and spatial maps as described in more details in clause 4.4.7.3.
- Spatial anchors and trackables as described in more details in clause 4.4.7.4.
- Camera parameters as defined in clause 4.4.3.2

XR Spatial Description is derived from or needs to be processed together with camera and sensor information. Typical raw sensor data is summarized in clause 4.4.7.2.

XR Spatial Description describes the real-world including information that is used for the estimation of position and orientation (pose estimation) of AR devices for the purpose of registration, tracking and positioning, and provides a coordinate reference system in relation to the real world. Generally, it may be used for spatial computing as described in clause 4.2.5.

The XR Spatial Description may be downloaded to the AR device and reside on the device. However, to support mobility and different environments, XR Spatial Description may have to be exchanged over the network and hence a formalized representation of XR Spatial Description may be needed. In this case, XR Spatial Description data has to be downloaded and updated periodically from a XR Spatial Description server.

In addition, the AR function may send XR Spatial Description updates to a XR Spatial Description server. Such data may be derived from XR Spatial Compute functions, e.g., updated visual spatial features, keyframes attached to camera parameters, or sub-parts of an XR Spatial Description. The server may use these XR Spatial Description updates to improve the XR Spatial Description for future use of the same user or by other users.

As the data needs to be updated, exchanged as well as stored on the device and the XR Spatial Description server, an efficient and flexible representation of XR Spatial Description is desired. For example, the description needs to be serialized and fragmented to be properly accessed and downloaded over the network.

The size of the XR Spatial Description depends on several parameters, for example, size of the area covered by the XR application, number of supported viewpoints in the area, the amount of keyframes that are provided, etc. The size may for example be from 10MByte for a small room to several hundred MBytes for a building. For a global-scale spatial map, the amount of data would be massively larger. As an example, the Microsoft™ Flight Simulator is around 2 Million GByte [63]. Regular exchange of data with the network is needed, details on the frequency, the latency requirements, and the bitrate requirements typically depend on the application, but more details are for further study.

As an example, the ETSI ISG ARF 004 [62] uses the term World Graph for XR Spatial Description. It defines the relative position of trackables and world anchors by 3D transforms. In this case, the World Graph is similar to a scene graph including trackables (embedding their features), and spatial anchors representing the real world. This information may be used by the AR Runtime for spatial compute functions including activity and object detection, object recognition, and pose estimation using trackables [21]. At the end of 2021, no non-proprietary XR Spatial Description formats are known.

4.4.7.2 Camera and sensor information

In this clause we provide an overview of different sensors that may provide input data for spatial compute AR functions. All device-captured data require a common timeline and a common coordinate system in order to be meaningful for XR Spatial Compute processing. If the data is processed in the networked, such time and spatial synchronization information is expected to be maintained.

AR Glasses typically include multiple cameras (for example one device supporting 7 cameras) to build precise motion tracking and gesture recognition. Generally, these camera feeds are processed on the device, but they may be sent across the network to support spatial compute functions. Different cameras exist on a single device, namely

- Monochrome image capture cameras,
- RGB image capture cameras,
- Infrared capture cameras.

Optical 3D sensors may be used to capture and reconstruct three-dimensional depth of objects. Depending on the source of the radiation, optical 3D sensors may be divided in two sub-categories; passive and active systems. Stereoscopic systems, Shape-from-Silhouettes (SfS), and Shape-from-Texture (SfT) are examples of passive systems, which do not emit any kind of radiation themselves. The sensors collect images of the scene, eventually from different points of view or with different optical setups. Then the images are analysed in order to compute the 3D depth of points in the scene. On the contrary, active systems emit some kind of radiation and the interaction between the object and the radiation is captured by a sensor. From the analysis of the captured data, knowing the features of the emitted radiation, the coordinates of the points are obtained. Time-of-Flight (ToF), phase shift, and active triangulation are examples of active systems. The typical output of an optical 3D sensor is a depth map image as described in clause 4.4.4.

Light Detection And Ranging (LiDAR) may be another option to measure distances (ranging) by illuminating the target with a light and then measuring the reflection with an optical sensor. In practice, LiDAR cameras operate in the ultraviolet, visible or infrared spectrum. Since the laser light used is typically collimated, the LiDAR camera needs to scan the scene in order to generate an image with a usable Field-of-View. The output of a LiDAR acquisition is a point cloud which may then be enriched with other sensor data such as RGB data.

Devices may also include microphones. A typical setup is a two-channel microphone array for audio input. Multichannel microphones or even Higher-Order Ambisonics (HOA) microphone arrays may be supported as well. The resulting signals are two- or multi-channel audio signals or HOA signals.

Typical sensor and fusion data are accelerometer, gyroscope, and magnetometer samples. This information includes, for example, angular velocity from gyroscopes, accelerometer data including the effect of gravity, as well as statistical data around the measurements. Detailed representations are for further study.

4.4.7.3 Visual features, Keyframes and Spatial Maps

Visual features, keyframes, and spatial maps are used for mapping the real world, typically as part of the SLAM process.

Visual features are characteristics of real-world elements that are searched, recognized and tracked in 2D images captured by the AR device as the user moves in a real environment. These images provide a view of the same real world elements, captured from different positions (as indicated by the camera parameters attached to them) from a single

moving camera or multiple cameras. Visual features are generally extracted from points that are recognizable in multiple images.

From the captured images of the real world, keyframes that include one or multiple visual features may be stored for later use. Visual features from the captured frames may be matched by comparing those frames with keyframes available to the AR Runtime in order to support the SLAM process. Keyframes have attached camera information defined in 4.4.3.2 to triangulate 3D points correctly from multiple cameras. These 3D points, triangulated from matching visual features are called spatial features.

Finally, a spatial map may be generated from keyframes and their matched visual features. A spatial map is, thus, a digital representation of the real world surrounding users consisting of at least one spatial feature cloud, e.g., 3D points (vector of 3 floats) with their associated descriptors such as SIFT [59], SURF [60], or ORB [61]. The geometrical part of the spatial map may be represented as a sparse or dense point cloud or a mesh. The mapping process may be performed either at runtime or offline. The spatial map is then used at runtime to relocalize and thus register the AR device by matching the visual features extracted from the current captured frames with spatial features stored in the spatial map. The spatial mapping approach described herein is one of well-known keyframe-based SLAM techniques [58].

The descriptors of features, whether visual or spatial, are generally vectors of numbers (e.g., vector of 128 floats for SIFT, vector of 64 floats for SURF, vector of 32 integers for ORB). Note that other features such as 3D segments (e.g., a 3D starting point and a 3D ending point) may also be used. During the localization process, the visual features extracted from the current frame, captured by the device, are matched with the spatial features of the map, resulting in 2D-3D correspondences used to estimate the pose of the cameras. However, since the 2D-3D matching process consists of comparing the descriptors of visual features extracted from the current image and those of the spatial features of the map, the complexity may quickly increase for maps containing several hundred thousand or even millions of spatial features. To accelerate the 2D-3D matching process, a spatial map typically also includes the following metadata:

- Information required for keyframe retrieval. For example, a keyframe retrieval uses Bag-Of-visual-Words (BoW) model. In this case, the information consists of the vocabulary of the BoW model and corresponding descriptor for each keyframe (vector of occurrence counts of a vocabulary in the keyframe). Depending on the visual descriptor used, the vocabulary size is usually a 10-100 MByte, and this vocabulary may be reused.
- The visual features for each keyframes (e.g. 2D points with their associated descriptors such as SURF, SIFT, ORB represented by a vector of numbers). The number of features extracted per keyframe varies between 200 and 1000.
- A vector pair (identifier of the visual features, identifier of the spatial features) that matches the visual features of keyframes with the spatial features of the spatial feature cloud.

Using this metadata, instead of comparing all descriptors of visual features extracted from the current frame with all spatial feature descriptors (from the spatial feature cloud of the spatial map), reduces the otherwise high computational complexity.

The vision-based localization system may then accelerate the matching between visual and spatial features by:

- Matching the closest keyframe to the current frame by retrieving it with the BoW model
- Matching the visual features between the current frame and the retrieved keyframe
- Matching the visual features between the current frame and spatial feature cloud (knowing matches between visual features of the keyframes and spatial features of the spatial feature cloud)

Figure 4.4.7.3-1 illustrates the localization process of a captured 2D frame using a spatial map. The figure shows a current frame with visual features highlighted in green. The visual features from the current frame are matched with the spatial features and keyframe information stored in the spatial map to estimate the pose of the camera when it captured the frame.

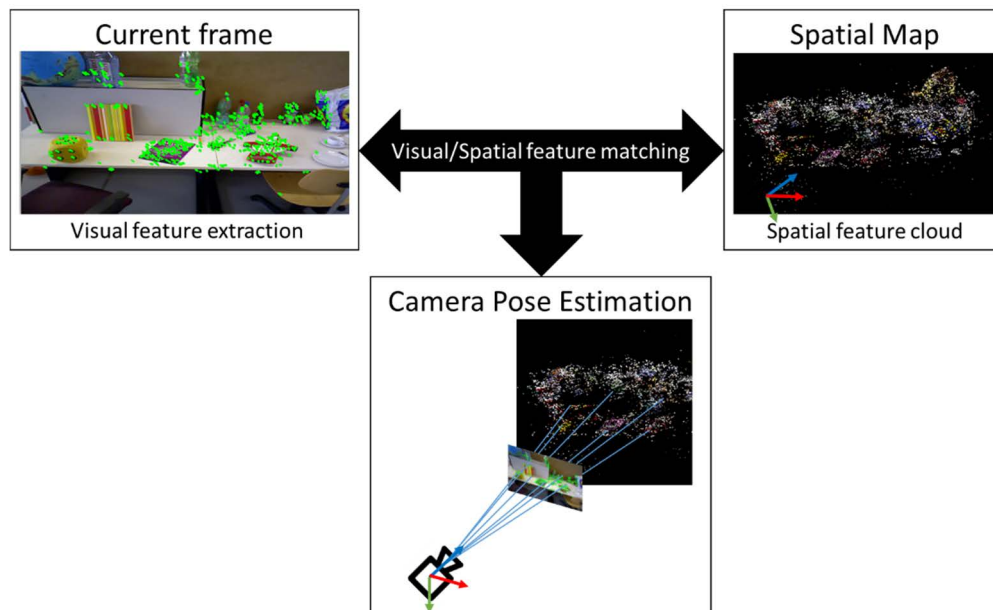


Figure 4.4.7.3-1 Camera pose estimation by features matching between a 2D captured frame and a spatial map

4.4.7.4 Spatial Anchors and Trackables

AR objects are positioned in reference to the real world (e.g., placing a vase on a table) using spatial anchors and trackables.

A spatial anchor provides a fixed position and orientation in the real world based on a common frame of reference that may be used by multiple AR devices. Spatial anchors are also used independently of other spaces in case global coordinates are available to the device. In this case, the anchors are treated as global anchors as they have global coordinates for which positions are determined.

However, in many cases an accurate global coordinate system is not available. In this case, spatial anchors refer to trackables for accurate positioning relative to the physical space. Trackables are elements of the real world for which features (visual or non-visual) are available and/or could be extracted. A trackable may for example be a spatial map that defines a full environment composed of floor walls and furniture in the real world consisting of several 3D points with visual features. However, there are other types of trackables, for example:

- A controller with LEDs that may be tracked by an AR headset's vision sensor. The feature in this case is the constellation of LEDs.
- A fiducial marker that is detected as a black and white pattern by an AR device vision sensor. The feature in this case is the black and white pattern.
- Hands visible through an AR headset's vision sensor. The feature is a learnt model for hands.

All of the above examples give a position of the trackable in reference to the position of the sensor (generally embedded in the AR headset).

4.5 Key Performance Indicators and Metrics for AR

4.5.1 Summary of TR 26.928

In TR 26.928 [2], clause 4.2 quality experience for XR is summarized. In order to provide the feeling of presence in immersive scenes, this clause provides a summary. TR 26.928 has some focus on VR and HMDs.

Table 4.5.1-1: KPIs from TR 26.928 with focus on VR and HMDs

Feature	KPI from TR 26.928
Tracking	
Freedom Tracking	6DoF
Translational Tracking Accuracy	Sub-centimeter accuracy - tracking accuracy of less than a centimeter
Rotational Tracking Accuracy	Quarter-degree-accurate rotation tracking
VR Games tracking space	roughly 2m cubes
Tracking frequency	At least 1000 Hz
Latency	
motion-to-photon latency	Less than 20 ms
pose-to-render-to-photon latency	50ms for render to photon in order to avoid wrongly rendered content
Interaction delay for games	50 to 1000ms
Video Rendering	
Persistence – Duty time	Turn pixels on and off every - 3 ms to avoid smearing / motion blur
Display refresh rate	90 Hz and beyond to eliminate visible flicker
Spatial Resolution	<ul style="list-style-type: none"> - 2K by 2K required - 4K by 4K desired
Optics	
Field of View	typically 100 - 110 degrees FOV is needed
Eye Box	<p>the minimum and maximum eye-lens distance wherein a comfortable image can be viewed through the lenses.</p> <p>at least 10mm, ideally rather 20mm</p>
Calibration	correction for distortion and chromatic aberration that exactly matches the lens characteristics
Depth Perception	Avoid vergence and accommodation conflict (VAC) for accommodation at fixed same distance (e.g. 2m)
Physics	
Maximum Available Power	<p>VR/AR HMD: 3-7 W</p> <p>AR Glass: 0.5 – 2W</p>
Maximum Weight	<p>VR HMD: several 100 grams</p> <p>AR Glass: 70g - if that weight is well distributed</p>

4.5.2 Updated KPIs for AR

In TR 26.928 [2], some high-level statements on experience KPIs for AR are provided. To achieve Presence in Augmented Reality, seamless integration of virtual content and physical environment is required. Like in VR, the virtual content has to align with user's expectations. For truly immersive AR and in particular MR, it is expected that users cannot discern virtual objects from real objects.

Also relevant for VR and AR, but in particular AR, is not only the awareness for the user for the environment. This includes, safe zone discovery, dynamic obstacle warning, geometric and semantic environment parsing, environmental lighting and world mapping.

Based on updated information, Table 4.6.2-1 provides new KPIs with focus on AR and in particular glasses. For some background and additional details refer for example to [10], [11], [49], [50], and [51].

Table 4.5.2-1 KPIs from TR 26.928 with focus on AR glasses

Feature	KPIs for AR glasses
Tracking	
Freedom Tracking	6DoF
Translational Tracking Accuracy	Sub-centimeter accuracy - tracking accuracy of less than a centimeter
Rotational Tracking Accuracy	Quarter-degree-accurate rotation tracking is desired
AR tracking space	In AR, the tracking space is theoretically unlimited. However, when moving, tracking accuracy may not be assured beyond a certain level of space or trajectory distance. SLAM based methods quickly introduce a large drift in large scale mapping. To correct the scaling issues, a loop closure technique [12] needs to be applied in order to continuously harmonize the local coordinate systems with global ones.
World-scale experience	World-scale experiences that let users wander beyond <ul style="list-style-type: none"> - orientation-only or seated-scale experiences - standing-scale or room-scale experiences To build a world-scale experience, techniques beyond those used for room-scale experiences, namely creating an absolute room-scale coordinate system that is continuously registered with the world coordinate system, typically requiring dynamic sensor-driven understanding of the world, continuously adjusting its knowledge over time of the user's surroundings.
Tracking frequency	At least 1000 Hz
Latency (for more details refer to clause 4.5.3)	
motion-to-photon latency	Less than 20 ms, and preferably even sub 10ms for AR as you may observe movement against the real world.
pose-to-render-to-photon latency	50-60ms for render to photon is desired in order to avoid wrongly rendered content with late warping applied.

Video Rendering and Display	
Persistence – Duty time	Turn pixels on and off every 2 - 4 ms to avoid smearing / motion blur
Display refresh rate	60 Hz minimum 90 Hz acceptable 120 Hz and beyond desired 240 Hz would allow always on display at 4ms
Colour	RGB colours Accurate colours independent of viewpoint.
Spatial Resolution per eye	for 30 x 20 degrees - 1.5K by 1K per eye is required - 1.8K by 1.2K per eye is desired for 40 x 40 degrees - 2K by 2K required - 2.5 K by 2.5 K desired ultimate goal for display resolution is reaching or going slightly beyond the human vision limit of roughly one arcmin ($1/60^\circ$)
Content frame rates	Preferably matching the display refresh rate for lowest latency Lower frame rates for example 60 fps or 90 fps may be used but add to overall end to end delay.
Brightness	200-500 nits for indoor Up to 2K for state-of-the-art devices in 2021 [49] 10K to 100K nits for full outdoor experiences
Optics	
Field of View	Augmentable FoV - typically, 30 by 20 degrees FoV acceptable - 40 by 40 degrees desired maximize the non-obscured field of view
Eye Relief	the minimum and maximum eye-lens distance wherein a comfortable image can be viewed through the lenses. at least 10mm, ideally rather 20mm
Calibration	correction for distortion and chromatic aberration that exactly matches the lens characteristics

Depth Perception	Avoid vergence and accommodation conflict (VAC) for accommodation being different for the real and virtual object
Physics	
Maximum Available Power	AR Glass: below 1 W, typically 500mW For less design-oriented devices, additional power may be available.
Maximum Weight	AR Glass: around 70g. However, if the weight is well distributed, several hundred grams may be acceptable.

4.5.3 Typical Latencies in networked AR Services

Building on top of the architectures introduced in clause 4.2 in this document as well as the latency considerations in TR 26.928 [2], Figure 4.5.3-1 provides a summary of different latencies involved networked AR services. Based on TR 26.928 as well as Table 4.5.2-1, two relevant latency requirements for adequate user experience matter:

- motion-to-photon latency being less 20ms, but preferably even single digit latency below 10ms.
- pose-to-render-to-photon latency: as small as 50-60ms

It is important to note that the motion-to-photon latency is primarily a function of the device implementation as it is basically covered within the AR runtime. What matters and is relevant is the time used to provide the pose information from the AR runtime to the renderer and the renderer using this pose to generate the displayed media. Final pose correction to the latest pose may always be done in the AR runtime.

Figure 4.5.3-1 provides different latency critical uplink and downlink operations, depending on where the rendering is done, locally, in the edge or in the cloud. If done in the edge or cloud, rendered data needs to be delivered in low-latency and high-quality over the network. The typical operations in this case include:

- pose detection in the UE
- sending the pose through a 5G uplink network to the edge of cloud.
- rendering the scene in the edge or cloud
- compressing and encrypting the rendered scene and delivering to the UE
- decrypting and decompressing the rendered scene
- composition of the scene
- applying the latest pose in the pose correction and display the immersive media.

Note that Figure 4.5.3-1 also adds buffers that are typically handled by the AR Run time, namely eye and depth as well as sound buffers.

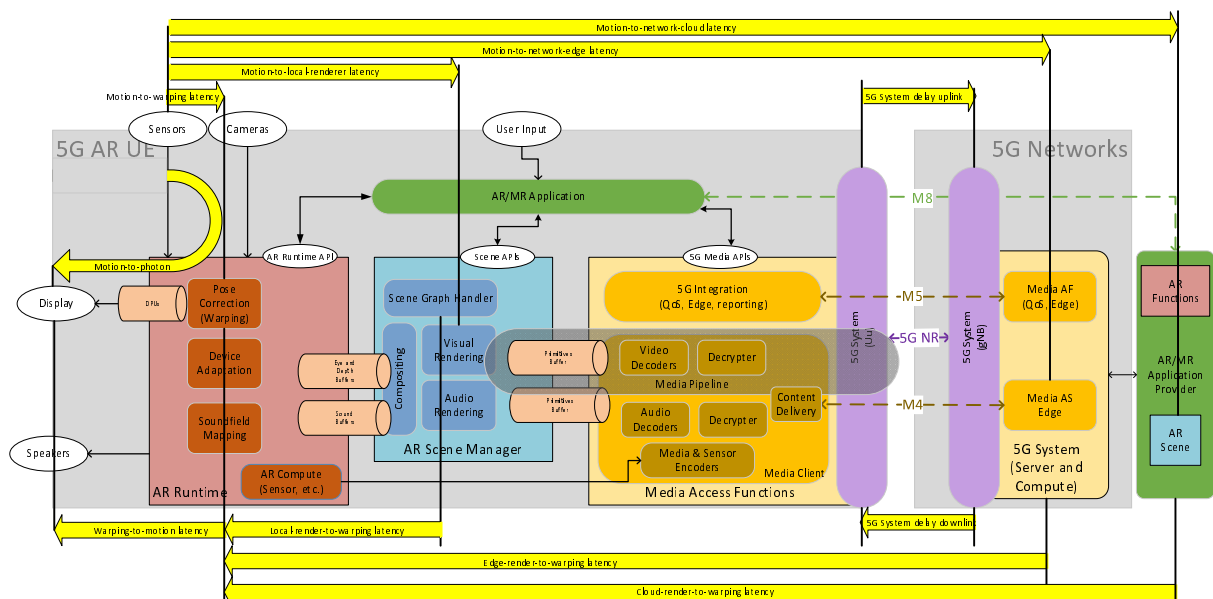


Figure 4.5.3-1: Typical Latencies in networked AR services

It is ultimately relevant that in case of networking the rendering loop, the processes in the loop are executed such that the end-to-end latency requirements for the pose-to-render-to-photon latency are ensured. Clearly the “closer” the rendering happens at the AR UE, the easier it is to meet latency requirements. However, with proper support of 5G system and media functionalities, these networked AR challenges are solved. This is subject of the remaining discussion of this report.

With reference to TR 26.928 [2], other types of latencies impact the user experience, for example when used for cloud gaming, interactive scenes or in case of real-time network-based processing of sensor data. These aspects are not specific to AR but are also relevant. Some more details are provided in clause 6 for the different scenarios.

4.6 Related Work

4.6.1 3GPP

This clause documents the 3GPP activity related to services using AR/MR device.

- 3GPP TR 26.928 [2] provides an introduction to XR including AR and a mapping to 5G media centric architectures. It also specified the core use cases for XR and device types.
- 3GPP TS 22.261 [13] identified use cases and requirements for 5G systems including AR and 3GPP TR 22.873 [14] documents new scenarios of AR communication for IMS Multimedia Telephony service.
- 3GPP SA4 is working on the documentation of 360-degree video support to MTSI in 3GPP TS 26.114 [15]. It will provide the recommendations of codec configuration and signalling mechanisms for viewport-dependent media delivery.
- 3GPP TR 26.926 [48] provides Traffic Models and Quality Evaluation Methods for Media and XR Services in 5G Systems.
- In the context of Release-17, 3GPP RAN work [16] identified traffic models for XR application and an evaluation methodology to access the XR performance.
- 3GPP SA4 is working on the development of the EVS Codec Extension for Immersive Voice and Audio Services (IVAS) codec. It targets encoding/decoding/rendering of speech, music and generic sound, with low latency operation and support of high error robustness under various transmission conditions. The IVAS codec is expected to provide support for a range of service capabilities, e.g., from mono to stereo to fully immersive audio, implementable on a wide range of UEs. The work on IVAS is expected to provide support for MTSI services and potentially streaming services through the definition of a new immersive audio media component.

NOTE: The integration of IVAS into the architectures developed in this report is for further study.

- In the context of Release-18 under the Terminal Audio quality performance and Test methods for Immersive Audio Services (ATIAS) work item, 3GPP SA4 is working on the specification of test methods in 3GPP TS 26.260 [56] and requirements in TS 26.261 [57] for immersive audio.

4.6.2 MPEG

MPEG has developed a suite of standards for immersive media with a project of MPEG-I (ISO/IEC 23090 Coded Representation of Immersive Media). It contains all the media related components, including video, audio, and system for AR/MR as well as 360-degree video.

- Part 1 – Immersive Media Architectures: Provides the structure of MPEG-I, core use cases and scenarios, and definitions of terminologies for immersive media
- Part 2 – Omnidirectional Media Format (OMAF): Defines a media format that enables omnidirectional media applications (360-degree video) with support of 3DoF, 3DoF+, and 6DoF based on ISO/BMFF. The second edition of OMAF was published in 2021 [17].
- Part 3 – Versatile Video Coding (VVC): Describes the 2D video compression standard, providing the improved compression performance and new functionalities as compared to HEVC. The first edition was published in 2021 [18].
- Part 4 – Immersive Audio Coding: It provides the compression and rendering technologies to deliver 6DoF immersive audio experience.
- Part 5 – Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC): It defines the coding technologies for point cloud media data, utilizing the legacy and future 2D video coding standards. The first edition was published in 2021 [19] and the second edition for dynamic mesh compression is developing.
- Part 6 – Immersive Media Metrics: It specifies a list of media metric and a measurement framework to evaluate the immersive media quality and experience. The first edition was published in 2021 [47].
- Part 7 – Immersive Media Metadata: It defines common immersive media metadata to be referenced to various other standards.
- Part 8 – Network-Based Media Processing (NBMP): It defines a media framework to support media processing for immersive media which may be performed in the network entities. It also specifies the composition of network-based media processing services and provides the common interfaces. The first edition was published in 2020 [20], and currently developing media processing entity capabilities and split rendering support as the second amendment.
- Part 9 – Geometry-based Point Cloud Compression (G-PCC): It defines the coding technologies for point cloud media data, using techniques that traverse directly the 3D space in order to create the predictors for compression.
- Part 10 – Carriage of Visual Volumetric Video-based Coding Data: It specifies the storage format for V3C and V-PCC coded data. It also supports flexible extraction of component streams at delivery and/or decoding time.
- Part 11 – Implementation Guidelines for Network-based Media Processing
- Part 12 – Immersive Video: It provides coding technology of multiple texture and depth views representing immersive video for 6DoF.
- Part 13 – Video Decoding Interface for Immersive Media: It provides the interface and operation of video engines to support flexible use of media decoder. MPEG Systems has also initiated the next phase of development for extending MPEG-I Scene Description including the support of additional immersive media codecs, support for haptics, AR anchoring, user representation and avatars, as well as interactivity.
- Part 14 – Scene Description for MPEG Media: It describes the spatial-temporal relationship among individual media objects to be integrated.
- Part 15 – Conformance Testing for Versatile Video Coding

- Part 16 – Reference Software for Versatile Video Coding
- Part 17 – Reference Software and Conformance for Omnidirectional Media Format
- Part 18 – Carriage of Geometry-based Point Cloud Compression Data
- Part 19 – Reference Software for V-PCC
- Part 20 – Conformance for V-PCC
- Part 21 – Reference Software for G-PCC
- Part 22 – Conformance for G-PCC
- Part 23 – Conformance and Reference Software for MPEG Immersive Video
- Part 24 – Conformance and Reference Software for Scene Description for MPEG Media
- Part 25 – Conformance and Reference Software for Carriage of Visual Volumetric Video-based Coding Data
- Part 26 – Conformance and Reference Software for Carriage of Geometry-based Point Cloud Compression Data

4.6.3 ETSI Industry Specification Group

ETSI Industry Specification Group AR Framework (ISG ARF) has developed a framework for AR components and systems [21]. It introduces the characteristics of an AR system and describes the functional building blocks of the AR reference architecture and their mutual relationships. The generic nature of the architecture is validated by mapping the workflow of several use cases to the components of this framework architecture.

The ETSI AR Framework Architecture describes a system composed of hardware and software components as well as data describing the real world and virtual content. The architecture is composed of three layers as described in clause 4 of [21] and illustrated in Figure 4.6.3-1.

- Hardware layer including:
 - > Tracking Sensors: These sensors aim to localize (position and orientation) the AR system in real-time in order to register virtual contents with the real environment. Most of AR systems such as smartphones, tablets or see-through glasses embed at least one or several vision sensors (generally monochrome or RGB cameras) as well as an inertial measurement unit and a GPSTM. However, specific and/or recent systems use complementary sensors such as dedicated vision sensors (e.g. depth sensors and event cameras), or exteroceptive sensors (e.g. Infrared/laser tracking, Li-FiTM and Wi-FiTM).
 - > Processing Units: Computer vision, machine learning-based inference as well as 3D rendering are processing operations requiring significant computing resources optimized thanks to dedicated processor architectures (e.g. GPU, VPU and TPU). These processing units may be embedded in the device, may be remote and/or distributed.
 - > Rendering Interfaces: Virtual content require interfaces to be rendered to the user so that he or she may perceive them as part of the real world. As each rendering device has its own characteristics, the signals generated by the rendering software generally need to be transformed in order to adapt them to each specific rendering hardware.
- Software layer including:
 - > Vision Engine: This software aims to mix the virtual content with the real world. It consists of localizing (position and orientation) the AR device relative to the real world reference, localizing specific real objects relatively to the AR device, reconstructing a 3D representation of the real world or analysing the real world (e.g. objects detection, segmentation, classification and tracking). This software component essentially uses vision sensors signals as input, but not only (e.g. fusion of visual information with inertial measurements or initialization with a GPS), it benefits from the hardware optimization offered by the various dedicated processors embedded in the device or remote, and will deliver to the rendering engine all information required to adapt the rendering for a consistent combination of virtual content with the real world.
 - > 3D Rendering Engine: This software maintains an up-to-date internal 3D representation of the virtual scene augmenting the real world. This internal representation is updated in real-time according to various inputs

such as user's interactions, virtual objects behaviour, the last user viewpoint estimated by the Vision Engine, an update of the World Knowledge to manage for example occlusions between real and virtual elements, etc. This internal representation of the virtual content is accessible by the renderer (e.g. video, audio or haptic) which produces thanks to dedicated hardware (e.g. Graphic Processing unit) data (e.g. 2D images, sounds or forces) ready to be played by the Rendering Interfaces (e.g. screens, headphones or a force-feedback arm).

- Data layer including:
 - > World Knowledge: This World Knowledge represents the information either generated by the Vision Engine or imported from external tools to provide information about the real world or a part of this world (CAD model, markers, etc.). This World Knowledge corresponds to the digital representation of the real space used for different usages such as localization, world analysis, 3D reconstruction, etc.
 - > Interactive Content: These Interactive Content represent the virtual content mixed to the perception of the real world. These contents may be interactive or dynamic, meaning that they include both 3D contents, their animations, their behaviour regarding input events such as user's interactions. These Interactive Contents could be extracted from external authoring tools requiring to adapt original content to AR application (e.g. 3D model simplification, fusion, and instruction guidelines conversion).

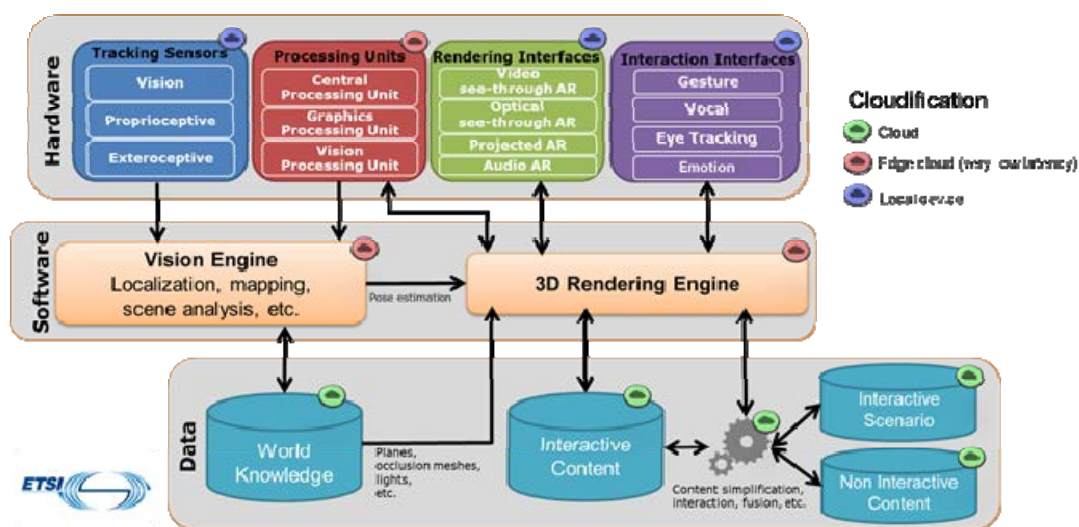


Figure 4.6.3-1: Global overview of the architecture of an AR system

In the ETSI AR functional architecture, there are eleven logical functions as illustrated in Figure 4.6.3-2. Each function is composed of two or more subfunctions.

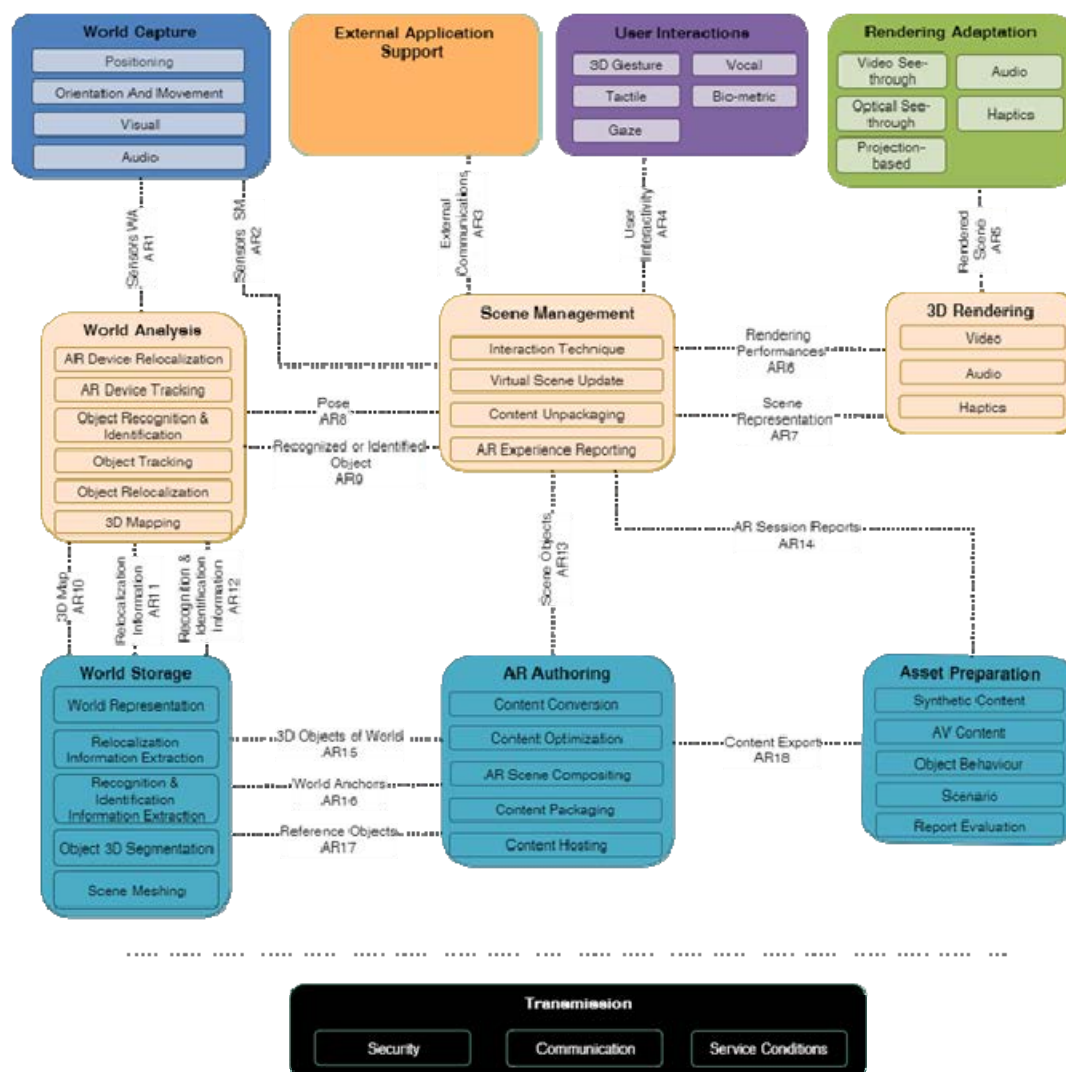


Figure 4.6.3-2: Diagram of the functional reference architecture

The ETSI ISG ARF has validated that the functional architecture covers all requirements for AR experience delivery in a variety of use cases. The logical functions are connected by Reference Point (RP). An RP in the AR functional architecture is located at the juncture of two non-overlapping functions and represents the interactions between those functions.

Details for each of the eleven functions and their subfunctions are described in clause 5 of [21] and details of each of the 18 RPs are described in clause 6 of [21].

4.6.4 Work related to AR Runtime

4.6.4.1 OpenXR

OpenXR [4] is an API that is developed by the Khronos Group for developing XR applications that address a wide range of XR devices. XR refers to a mix of real and virtual world environments that are generated by computers through interactions by humans. XR includes technologies such as virtual reality (VR), augmented reality (AR) and mixed reality (MR). OpenXR is the interface between an application and XR runtime. The runtime handles functionality such as frame composition, user-triggered actions, and tracking information.

OpenXR is designed to be a layered API, which means that a user or application may insert API layers between the application and the runtime implementation. These API layers provide additional functionality by intercepting OpenXR functions from the layer above and then performing different operations than would otherwise be performed without the layer. In the simplest cases, the layer simply calls the next layer down with the same arguments, but a more complex layer may implement API functionality that is not present in the layers or runtime below it. This mechanism is

essentially an architected "function shimming" or "intercept" feature that is designed into OpenXR and meant to replace more informal methods of "hooking" API calls.

Applications may determine the API layers that are available to them by calling the [xrEnumerateApiLayerProperties](#) function to obtain a list of available API layers. Applications then may select the desired API layers from this list and provide them to the [xrCreateInstance](#) function when creating an instance.

API layers may implement OpenXR functions that may or may not be supported by the underlying runtime. In order to expose these new features, the API layer must expose this functionality in the form of an OpenXR [extension](#). It must not expose new OpenXR functions without an associated extension.

An OpenXR instance is an object that allows an OpenXR application to communicate with an OpenXR runtime. The application accomplishes this communication by calling [xrCreateInstance](#) and receiving a handle to the resulting [XrInstance](#) object.

The [XrInstance](#) object stores and tracks OpenXR-related application state, without storing any such state in the application's global address space. This allows the application to create multiple instances as well as safely encapsulate the application's OpenXR state since this object is opaque to the application. OpenXR runtimes may limit the number of simultaneous [XrInstance](#) objects that may be created and used, but they must support the creation and usage of at least one [XrInstance](#) object per process.

Spaces are represented by [XrSpace](#) handles, which the application creates and then uses in API calls. Whenever an application calls a function that returns coordinates, it provides an [XrSpace](#) to specify the frame of reference in which those coordinates will be expressed. Similarly, when providing coordinates to a function, the application specifies which [XrSpace](#) the runtime to be used to interpret those coordinates.

OpenXR defines a set of well-known reference spaces that applications use to bootstrap their spatial reasoning. These reference spaces are: VIEW, LOCAL and STAGE. Each reference space has a well-defined meaning, which establishes where its origin is positioned and how its axes are oriented.

Runtimes whose tracking systems improve their understanding of the world over time may track spaces independently. For example, even though a LOCAL space and a STAGE space each map their origin to a static position in the world, a runtime with an inside-out tracking system may introduce slight adjustments to the origin of each space on a continuous basis to keep each origin in place.

Beyond the well-known reference spaces, runtimes expose other independently tracked spaces, such as a pose action space that tracks the pose of a motion controller over time.

Figure 4.6.4.1-1 depicts the lifecycle of an application that uses OpenXR for interaction and rendering with/to an HMD.

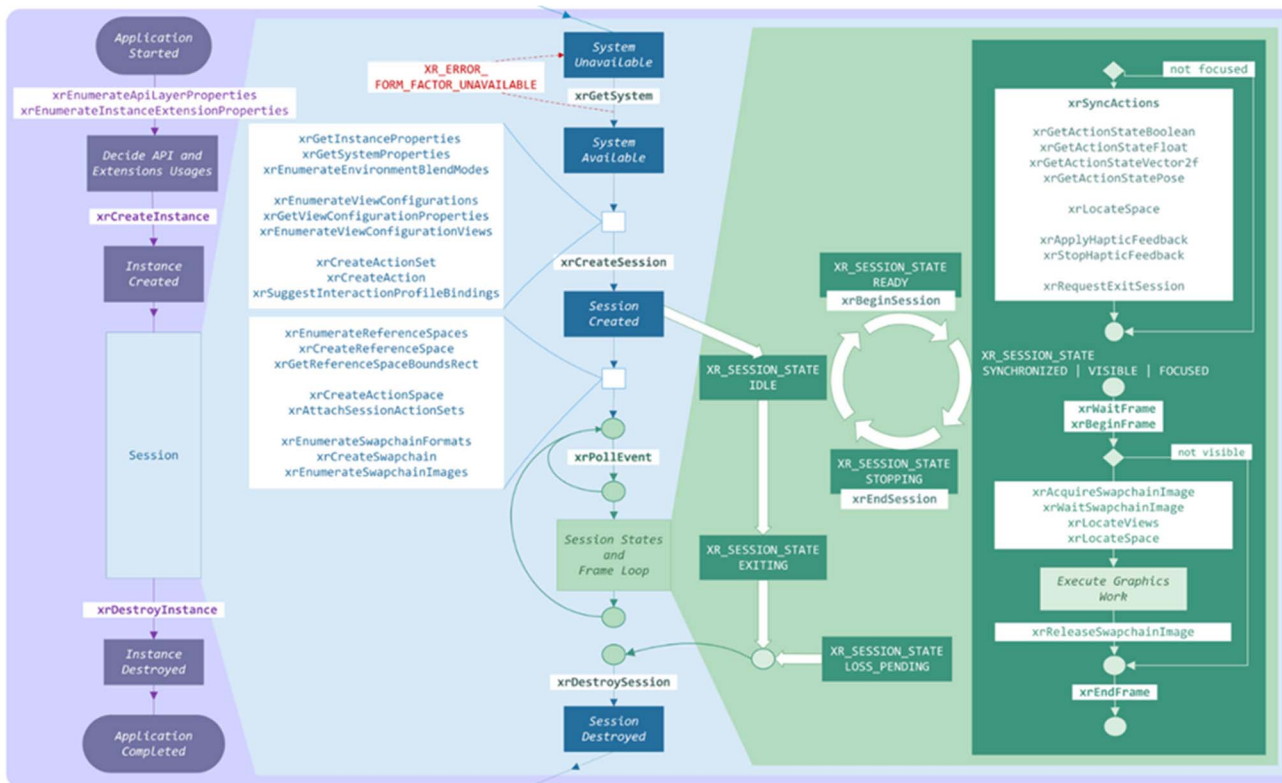


Figure 4.6.4.1-1: OpenXR application lifecycle

4.6.4.2 WebXR

WebXR [5] is a set of APIs that are developed by the W3C to provide support for augmented reality (AR) and virtual reality (VR) in web environments, hence the name WebXR for cross reality in the web. When a WebXR session is created, the mode of the session is indicated, i.e. whether it is an AR or VR session. VR sessions may be consumed in 2 ways, inline and immersive. In the inline mode, the VR content is rendered on the 2D screen as part of the web document. In the immersive mode, the content is rendered on an HMD with an immersive 3DoF experience. AR sessions are always immersive.

A typical lifecycle of a WebXR application will start by checking for availability of the WebXR API support in the current browser. When the user requests the activation of a WebXR functionality, an XRSession with the desired mode is created. The XRSession instance is then used to request a frame to render using the requestAnimationFrame call. Complex scenes may require threaded rendering, which may be achieved through the usage of Worker instances. WebGL is then ultimately used to render to the provided frame. When calling the requestAnimationFrame, the application provides a callback function that will be called when a new frame is about to be rendered. The callback function will receive a timestamp, indicating the current timestamp of the XR pose. It also receives an XRFrame, which holds information about the current XR poses for all objects that are being tracked by the session. This information is then used to perform correct rendering by the application. The XRFrame offers two main functions, the getPose and getViewerPose. The getPose functions returns the relationship between two XRSpaces, which are passed in as input to that function. The getViewerPose returns the viewer's pose in relationship to a reference XRSpace that is passed to the function call.

WebXR defines a set of reference XRSpace(s) as described in the Table 4.6.4.2-1:

Table 4.6.4.2-1: Reference XRSpace in WebXR

Reference XR Space	Description
bounded-floor	a tracking space with an origin that is located at the floor of the viewer's environment when the session was created. The XR space is bounded and movement outside that space is not supposed to happen.
local	a tracking space that corresponds to the viewer's position when the session was created. The user is not expected to move much beyond that starting position.
local-floor	a tracking space that corresponds to the viewer's floor position when the session was created, so that the viewer will be standing on that floor.
unbounded	a tracking space that allows total freedom of movement.
viewer	a tracking space that has an origin at the viewer's position and orientation. The origin tracks the viewer at all times.

4.6.5 MPEG Scene Description

A key technology in enabling immersive 3D user experiences is scene description. Scene description is used to describe the composition of a 3D scene, referencing and positioning the different 2D and 3D assets in the scene. The information provided in the scene description is then used by an application to render the 3D scene properly, using techniques such as Physically-Based Rendering (PBR) that produce realistic views.

A scene description is typically organized as a directed acyclic graph, typically a plain tree-structure, that represents an object-based hierarchy of the geometry of a scene and its attributes/properties. Nodes are organized in a parent-child hierarchy known informally as the node hierarchy. A node is called a root node when it doesn't have a parent. Any node may define a local space transformation.

Spatial transformations are represented by transformation matrices or separate transform operations such as translation, rotation, and scaling. The transformations are applied hierarchically and iteratively from the root node down to the child nodes. Scene description also support animation nodes that allow to animate properties of the corresponding objects over time.

This structure of scene description has the advantage of reduced processing complexity, e.g. while traversing the graph for rendering. An example operation that is simplified by the graph representation is the culling operation, where branches of the graph are omitted, if deemed that the parent node's space is not visible or relevant (level of detail culling) to the rendering of the current view frustum.

To address the needs of immersive applications, MPEG is finalizing the development of a scene description solution that adds extensions to glTF to support scene description. glTF 2.0 [22] provides a solid and efficient baseline for exchangeable and interoperable scene descriptions. However, glTF 2.0 has traditionally been focused on static scenes and assets, which makes it unfit to address the requirements and needs of dynamic and rich 3D scenes in immersive environments.

As part of its effort to define solutions for immersive multimedia, MPEG has identified the following gaps in glTF 2.0:

- No support for timed media
- No support for audio
- Limited support for interactions with the scene and the assets in the scene
- No support for local and real-time media, which are crucial for example for AR experiences

Based on this analysis, MPEG has an ongoing project to extend glTF 2.0 with the ability to add timed media to glTF 2.0-based scenes standardized in ISO/IEC 23090-14 [23].

Additional extensions for the support of interactivity and AR are currently being developed and will be part of the MPEG Scene Description in the next phase.

MPEG also developed an architecture to guide the work on immersive media and scene description. Figure 4.6.5-1 depicts the MPEG-I architecture and defines the key interfaces.

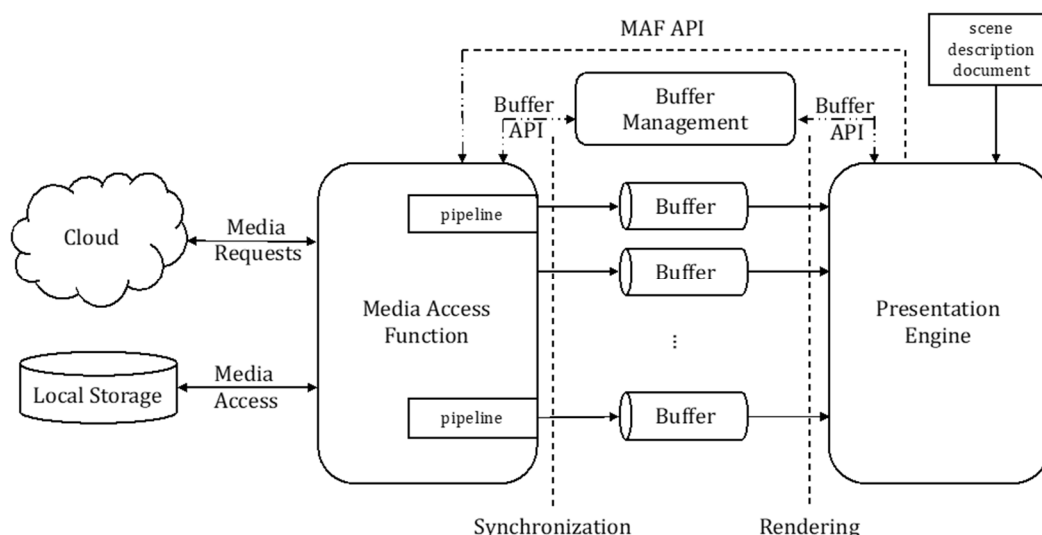


Figure 4.6.5-1: MPEG-I architecture and defines the key interfaces

The design focuses mainly on buffers as means for data exchange throughout the media access and rendering pipeline. It also defines a Media Access Function API to request media that is referenced by the scene description, which will be made accessible through buffers.

4.6.6 MPEG-I Video Decoding Interface for Immersive Media

The aim of VDI (MPEG-I part 13) is to address the challenges for media applications to handle multiple decoder instances running in parallel, especially in the case of immersive media. To this end, the scope of the VDI specification covers the interface between a media application and the Video Decoding Engine (VDE) sitting on the device.

Typically, hardware decoder is exposed via API to the application. Proprietary APIs exist but also standardised one such as Khronos® OpenMax™ and Khronos® Vulkan® Video extension. However, those APIs only allow the instantiation of video decoder instances independently from each other up to the point where the hardware decoding platform may no longer sustain the application requests, for instance due to lack of memory.

Extensions specified in MPEG-I VDI (ISO/IEC 23090-13) allow the AR/MR application to query the capacity to simultaneously decode multiple operation points (generally specified by profile, tier and levels). This allows a better predictability of what bitstreams may be decoded by the application.

Additionally, VDI also defines bitstream manipulation functions for the video codecs HEVC, VVC and EVC that enable the merging and the splitting of bitstreams. This aspect of elementary stream manipulation is covered by the so-called input formatting function in MPEG-I VDI. This way, an application may adapt the number of the decoder needed when several input bitstreams are to be decoded to the extent the merging operations has been enabled by the proper encoding constraints.

4.6.7 MPEG-I Carriage of Point Cloud Compression Data

For the encapsulation and storage of coded volumetric media, two MPEG systems standards may be considered as potential technologies: ISO/IEC 23090-10 [24] and ISO/IEC 23090-18 [25]. ISO/IEC 23090-10 and ISO/IEC 23090-18 define how to structure and carry the various components in a V3C bitstream or G-PCC bitstream, respectively, in an ISOBMFF media container to support flexible and partial access (e.g., using multiple component tracks and tile tracks) as well as adaptive streaming. Both specifications support single track encapsulation as well as multi-track encapsulation, where different components of the bitstream are carried in separate tracks in the container. In addition, these standards also define metadata tracks that may be associated with the main media tracks and carry additional timed information that signal changes in the spatial partitioning of the volumetric content and the mapping to different independently decodable tiles as well as viewport-related information.

In addition, ISO/IEC 23090-10 and ISO/IEC 23090-18 define how to signal V3C and G-PCC content in a DASH MPD file. This includes defining new DASH descriptors that signal metadata about the type of component carried by each adaptation set and using pre-selections to group the adaptation sets of the different components associated with the

volumetric media content. Other descriptors are also defined for signalling independently decoded spatial sub-divisions of the content to support partial streaming. In addition to signalling for DASH-based delivery, ISO/IEC 23090-10 and ISO/IEC 23090-18 also define descriptors for signalling volumetric media assets for delivery over MMT.

4.6.8 Web Real-Time Communication (WebRTC)

4.6.8.1 WebRTC as an OTT application

The Web Real-Time Communication (WebRTC) is an API and set of protocols that enable real-time communication. The WebRTC protocols enable any two WebRTC agents to negotiate and setup a bidirectional and secure real-time communication channel. The WebRTC API exposes a JavaScript-based API to enable the development of applications that make use of the user's existing multimedia capabilities to establish real-time communication sessions. However, access to the WebRTC set of protocols is possible through other programming languages.

The WebRTC protocols are developed and maintained by the rtcweb group in IETF. The WebRTC API is developed by W3C.

The WebRTC API is decomposed into three layers:

- API for web developers that consists mainly of the `MediaStream`, `RTCPeerConnection`, and `RTCDataChannel` objects.
- API for browser and user agent implementers and providers
- Overridable API for audio/video capture and rendering and for network input/output, which the browser implementers may hook their own implementations to.

The main WebRTC stack components are the voice engine, the video engine, and the transport component.

The transport component ensures a secure transport channel for both parties of the call to communicate. It relies on an RTP protocol stack that runs over DTLS and leverages the SRTP profile.

The following diagram depicts the WebRTC protocol stack:

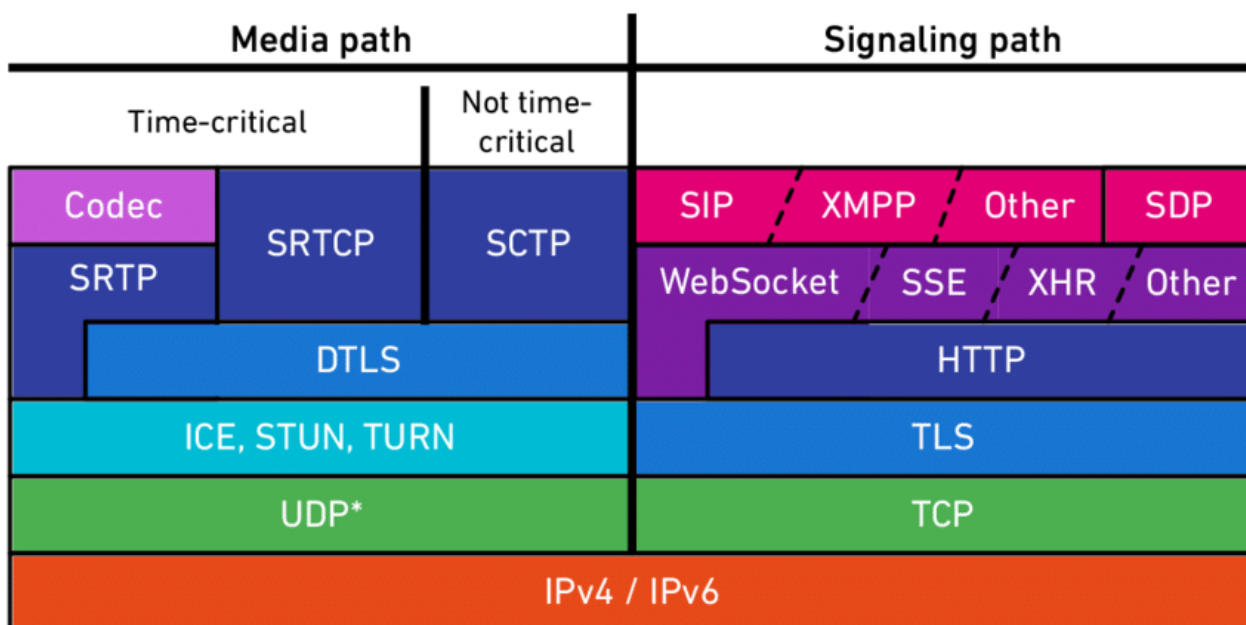


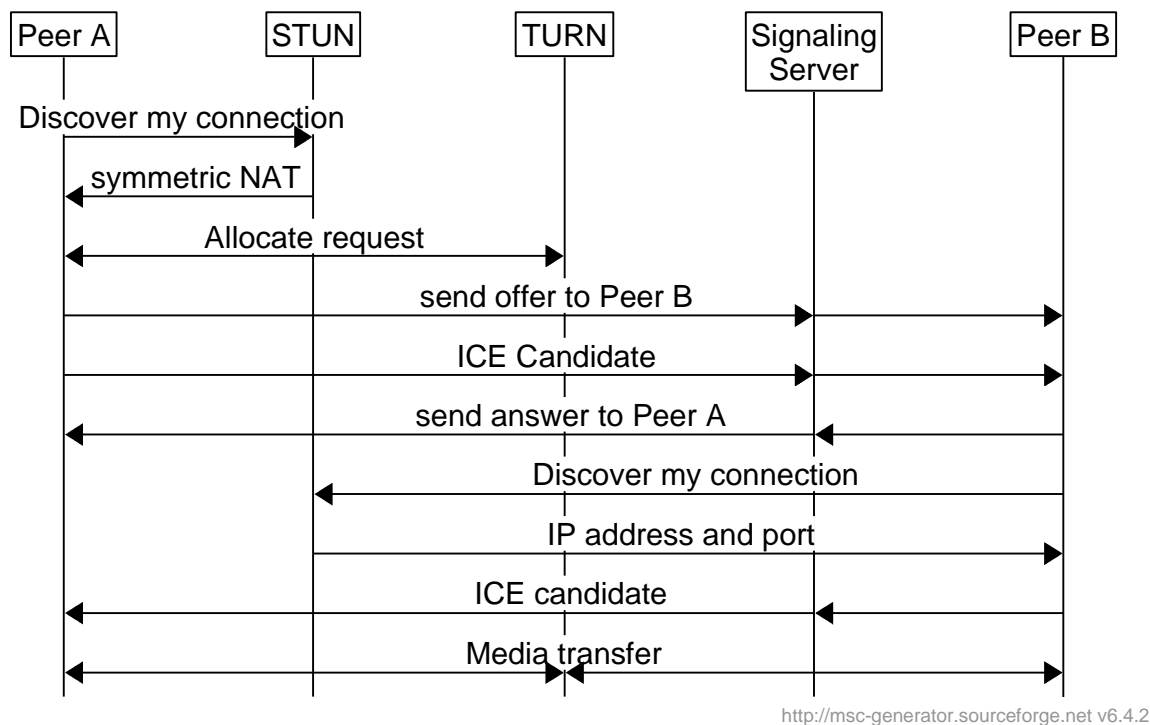
Figure 4.6.8-1: WebRTC protocol stack

WebRTC delegates the signalling exchange to the application. The signalling protocol and format may be chosen by the application freely. However, the offer and answer are generated in the SDP format. The ICE candidates may be provided as strings or in JSON format.

WebRTC needs negotiation for the following purposes:

- Negotiation of the media streams and formats: this relies on the SDP offer/answer mechanism to generate and validate media streams and parameters.
- Negotiation of the transport parameters: this relies on ICE to identify and test ICE candidates. Whenever a higher priority ICE candidate is validated, the connection will switch to it.

The following call flow shows an example of the ICE negotiation process:



<http://msc-generator.sourceforge.net> v6.4.2

Figure 4.6.8-2: WebRTC ICE negotiation process

Due to the separation of the negotiation of the transport parameters from the media parameters, appropriate QoS negotiation needs to consider consecutive and asynchronous changes to the connection parameters. In case of a relay server, such as a TURN server, is deployed, the QoS negotiation is to be extended to appropriately cover the outbound streams as well.

4.6.8.2 WebRTC framework for RTC Media Service Enablers

A subset of WebRTC, limited to a protocol stack and implementation excluding codecs and other media processing functions defined in W3C and/or IETF, is considered in clauses 6.5 and 8.3 to define an instantiation of AR conversational services.

4.6.9 Joint workshop with Khronos and MPEG on "Streamed Media in Immersive Scene Descriptions"

3GPP also participated in a joint workshop with Khronos and MPEG on the topic of "Streamed Media in Immersive Scene Descriptions" in September 2021 in order to identify common and complementary aspects on defining networked media. All presentations are provided in [53][54]. The workshop attracted more than 200 participants. A survey was conducted among the participants and there was broad positive feedback on the event with a request to a follow-up in 2022. Details are available in [54]. An initial summary of main observations is provided as follows:

- Complementary work – many touch points - collaboration seems to be beneficial
- Specific topics identified, but may be digested further
 - > gLTF and extensions by MPEG-I Scene description
 - > Tools and implementation support

- > Vulkan video and VDI
- > Extended Realities: OpenXR, MPEG-I Phase 2 including AR, Interactivity, and Haptics
- > Systems and Split Rendering: OpenXR, 3GPP connectivity, MPEG codecs
- Challenges: timelines, publication rules, IPR policies, membership
- Opportunities: complementary expertise, implementation and developer support, joint promotion, focus
- Proposed next steps:
 - > continue the discussion
 - > set up some kind of discussion platform

5 Core Use Cases

This clause documents the core use cases and scenarios for AR/MR devices, which serve to extract requirements, functional structure, related media format, and protocols for the 5G systems. Parts of the use cases are derived from XR use cases in TR26.928 [2] based on the relevance to AR/MR device type. In addition, the other use cases and scenarios are collected in Annex A of this document.

Table 5-1 provides a list of all the collected use cases.

Table 5-1: List of use cases for AR/MR services

No	Use Case	Reference
1	3D Image Messaging	Annex A.2 in [2]
2	AR Sharing	Annex A.3 in [2]
3	Real-time 3D Communication	Annex A.8 in [2]
4	AR guided assistant at remote location (industrial services)	Annex A.9 in [2]
5	Police Critical Mission with AR	Annex A.10 in [2]
6	Online shopping from a catalogue – downloading	Annex A.11 in [2]
7	Real-time communication with the shop assistant	Annex A.12 in [2]
8	360-degree conference meeting	Annex A.13 in [2]
9	XR Meeting	Annex A.16 in [2]
10	Convention / Poster Session	Annex A.17 in [2]
11	AR animated avatar calls	Annex A.18 in [2]
12	AR avatar multi-party calls	Annex A.19 in [2]
13	Front-facing camera video multi-party calls	Annex A.20 in [2]
14	AR Streaming with Localization Registry	Annex A.21 in [2]
15	5G Shared Spatial Data	Annex A.24 in [2]
16	AR remote cooperation	Annex A.1
17	AR remote advertising	Annex A.2
18	Streaming of volumetric video for glass-type MR devices	Annex A.3
19	AR Conferencing	Annex A.4
20	AR IoT	Annex A.5
21	AR gaming	Annex A.6
22	Shared AR conferencing experience	Annex A.7

The use cases may be grouped into several categories based on the similar requirements for media flow and device functional structure.

6 Mapping to 5G System Architecture

6.1 General

Based on the identified use cases in clause 5, this clause documents how AR/MR service scenarios are supported in 5G system architecture.

There already exist developed 5G system architectures relevant to deliver immersive media depending on the underlying functionalities, such as real-time communications, adaptive delivery, QoS guarantee, and a support of network node processing. An architecture of 5G Media Streaming (5GMS) for both downlink and uplink is specified in TS 26.501 [26] and it is under development to support the edge media processing. In addition, MTSI architecture extended to 5G system [15] may be applied to AR/MR conversational scenarios to guarantee the specific service QoS. In the following clauses, these relevant architectures will be analysed to identify potential standardisation areas for each scenario.

Note that only STAR UE and EDGAR UE in Table 4.2.2.1-1 are taken into account, as WLAR UE as well as WTAR UE have similar functionalities with STAR UE from a 5G system perspective. Specifically, STAR UE (and WLAR/WTAR UEs) possibly has an on-device decoding and rendering capability for immersive media and may rely on support from 5G cloud/edge for a certain condition. On the other hand, EDGAR UE always requires 5G cloud/edge for immersive media decoding and rendering, and the conventional 2D media is exchanged in Uu interface.

Table 6.1-1 provides a list of AR/MR service scenarios and the associated use cases for each. Note that some use cases may be duplicated as they address multiple features.

Table 6.1-1: List of service scenario mapping to use cases

Service Scenario	Clause	Relevant Use Case
Immersive media downlink streaming	6.2	2. AR Sharing ¹⁾ 14. AR Streaming with Localization Registry 17. AR remote advertising 18. Streaming of volumetric video for glass-type MR Devices
5G interactive immersive service	6.3	1. 3D Image Messaging 2. AR Sharing ¹⁾ 4. AR guided assistant at remote location (industrial services) ¹⁾ 5. Police Critical Mission with AR ¹⁾ 15. 5G Shared Spatial Data 16. AR remote cooperation ¹⁾ 21. AR gaming
5G cognitive immersive service	6.4	4. AR guided assistant at remote location (industrial services) ¹⁾ 5. Police Critical Mission with AR ¹⁾ 14. AR Streaming with Localization Registry ¹⁾ 16. AR remote cooperation ¹⁾ 20. AR IoT control
AR conversational service	6.5	3. Real-time 3D Communication 4. AR guided assistant at remote location (industrial services) ¹⁾ 7. Real-time communication with the shop assistant 8. 360-degree conference meeting ¹⁾ 9. XR Meeting ¹⁾ 10. Convention / Poster Session ¹⁾ 11. AR animated avatar calls 12. AR avatar multi-party calls ¹⁾ 13. Front-facing camera video multi-party calls ¹⁾ 16. AR remote cooperation ¹⁾ 19. AR Conferencing ¹⁾
Shared AR conversational experience	6.6	8. 360-degree conference meeting ¹⁾ 9. XR meeting ¹⁾ 10. Convention / Poster Session ¹⁾ 12. AR avatar multi-party calls ¹⁾ 13. Front-facing camera video multi-party calls ¹⁾ 19. AR conferencing ¹⁾ 22. Shared AR Conferencing Experience
1) may be duplicated into multiple scenarios		

6.2 Immersive media downlink streaming

6.2.1 Introduction

This clause introduces the case where immersive AR/MR media is streamed to a 5G AR UE using basic functionalities as defined in 5G Media Streaming for downlink (5GMSd).

6.2.2 Relevant use cases

The following use cases are relevant to this scenario.

- UC#2: AR sharing
- UC#14: AR Streaming with Localization Registry
- UC#17: AR remote advertising
- UC#18: Streaming of volumetric video for glass-type MR Devices

An immersive video which was pre-captured or pre-generated are stored in the server of an application provider. On a user's request, the desired immersive video is streamed to 5G AR UE throughout 5GMS architecture. The user is able to play, pause, stop, and enjoy the trick play while watching the video. In this use case, the scene description of the pre-generated video may get updated as the video progresses. However, these updates are independent to the user's interaction or change of pose.

6.2.3 Architectures

6.2.3.1 STAR-based

Figure 6.2.3.1-1 provides a basic extension of 5G Media Streaming for immersive media downlink using a STAR UE, when all essential AR/MR functions in a UE are available for typical media processing use cases. In addition to media delivery, also scene description data delivery is included.

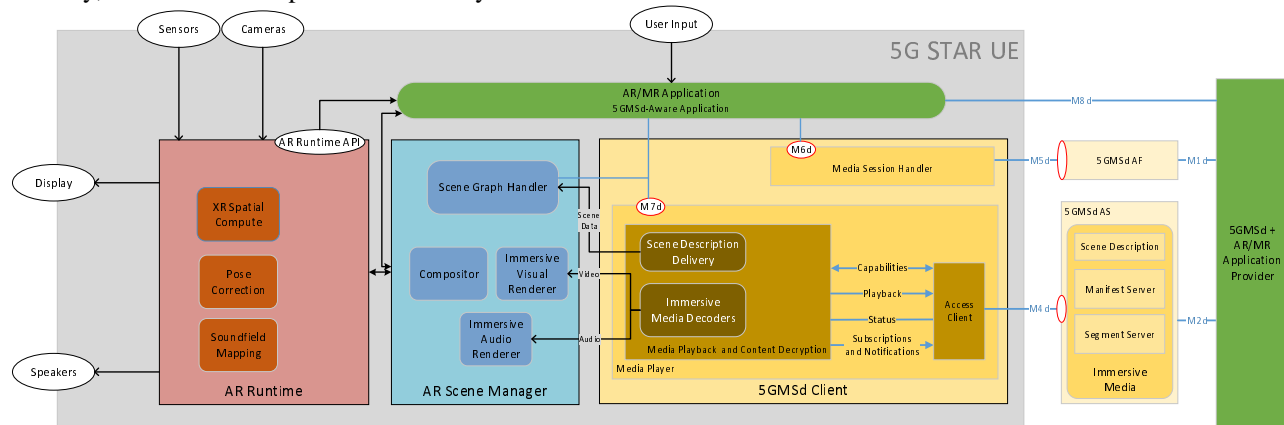


Figure 6.2.3.1-1: STAR-based 5GMS Downlink Architecture

6.2.3.2 EDGAR-based

Figure 6.2.3.2-1 provides a basic extension of 5G Media Streaming download for immersive media using an EDGAR UE. In this context, it is expected that the edge will pre-render the media based on pose and interaction information received from the 5G EDGAR UE. It is also highlighted, that the 5G EDGAR UE may consume the same media assets from an immersive media server as the STAR UE according to Figure 6.2.3.1-1, but the communication of the edge server to this immersive server is outside of the considered 5G Media Streaming architecture.

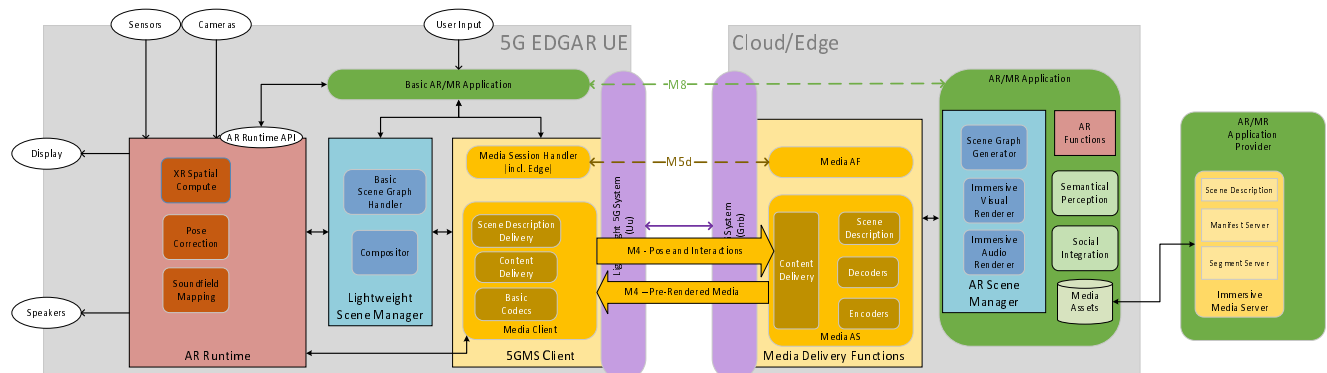
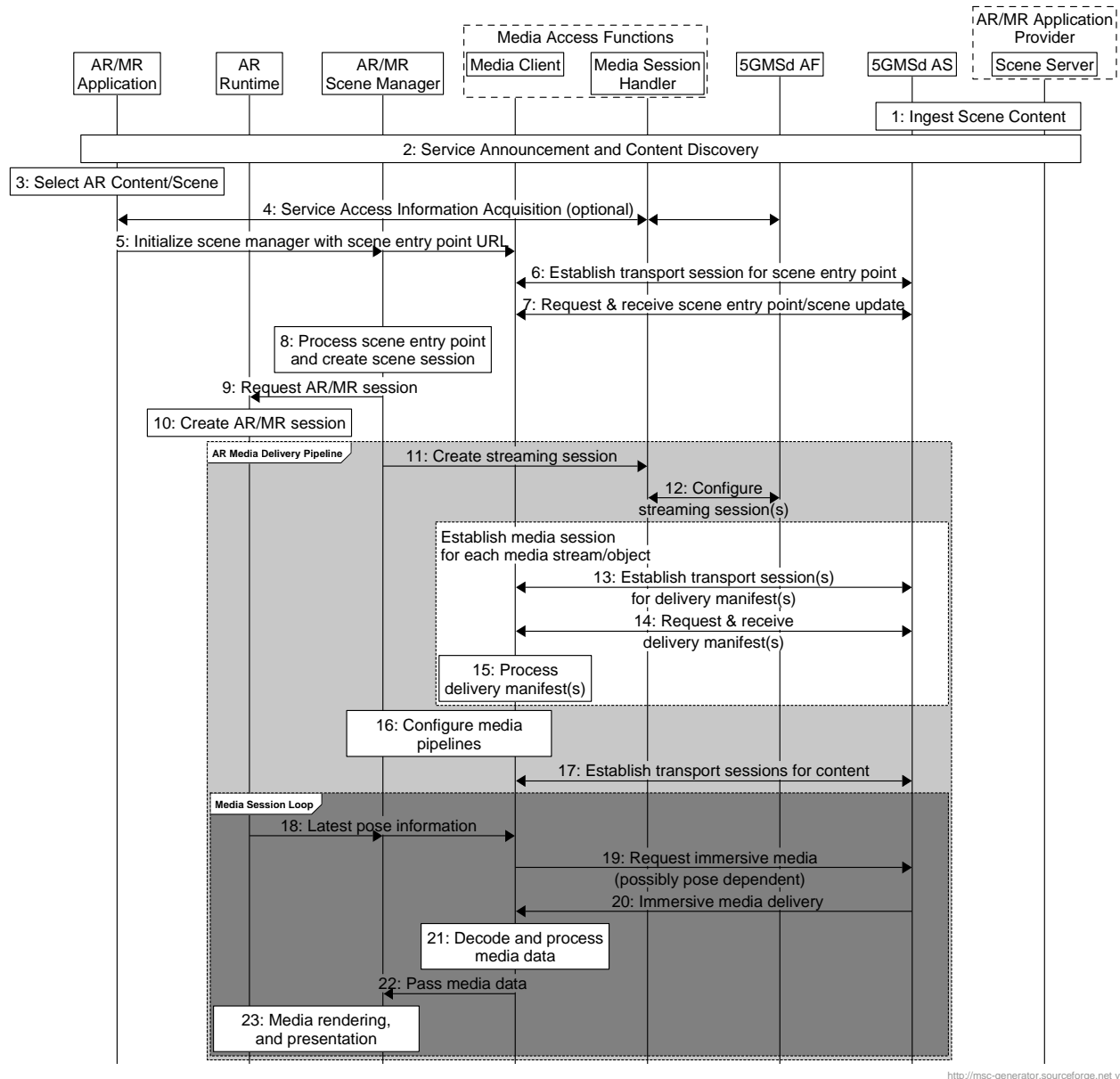


Figure 6.2.3.2-1: EDGAR-based 5GMS Downlink Architecture

6.2.4 Procedures and call flows

6.2.4.1 STAR-based media streaming

Figure 6.2.4.1-1 illustrates the procedure diagram for 5G immersive media downlink streaming using a STAR-based UE when all essential AR/MR functions in a UE are available without an assist by an edge.



<http://msc-generator.sourceforge.net v7.1>

Figure 6.2.4.1-1: STAR-based procedure for 5G downlink streaming

Prerequisites and assumptions:

- The AR/MR Scene Manager includes immersive media rendering and scene graph handling functionalities.
- The Media Player includes immersive content delivery and immersive media decoding functionalities.
- The AR/MR Application in the UE is run by the user.
- The STAR UE initialises AR registration (starts analysing the surroundings where a user/UE is located), it namely:
 - a. captures its surroundings via camera(s)

- b. analyses where the device is located
- c. registers the device into the analysed surroundings.
- AR/MR Application and AR/MR Application Provider have exchanged some information, such as device capability or content configuration, for content rendering. The exchange procedures for device capability and content configuration are FFS.
- AR/MR Application Provider has established a Provisioning Session and its detailed configurations has been exchanged.
- AR/MR Application Provider has completed to set up ingesting immersive contents.

Procedures:

1. The scene content is ingested by the 5GMSd AS.
2. Service Announcement is triggered by AR/MR Application. Service Access Information including Media Client entry or a reference to the Service Access Information is provided through the M8d interface.
3. Desired media content is selected.
4. Optionally, the Service Access information is acquired or updated.
5. The AR/MR Application initializes the Scene Manager with the entry point (full scene description) URL.
6. The Media Client establishes the transport session for receiving the entry point (scene description).
7. The Media Client requests and receives the full scene description.
8. The entry point (scene description) is processed.
9. The AR/MR Scene Manager requests the creation of a new AR/MR session from the AR Runtime.
10. The AR Runtime creates a new AR/MR session.

AR Media Delivery Pipeline, steps 11~23 requests, receives and renders scenes and scene updates:

11. The Media Client and/or AR/MR Scene Manager notifies the necessary QoS information required to the Media Session Handler.
12. The Media Session Handler shares the information with the 5GMSd AF, in some cases including desired QoS information. Based on existing provisioning by the AR/MR Application Provider, the 5GMSd AF may request QoS modifications to the PDU sessions.

Steps 13~15 establish the transport sessions, receives, and process the delivery manifests:

13. For the required media content, the Media Client establishes the transport session(s) to acquire delivery manifest(s) information.
14. The Media Client requests and receives the delivery manifest(s) from the 5GMSd AS.
15. The Media Client processes the delivery manifest(s). It determines for example the number of needed transport sessions for media acquisition. The Media Client is expected to be able to use the delivery manifest(s) information to initialize the media pipelines for each media stream.
16. The AR/MR Scene Manager and Media Client configures the rendering and delivery media pipelines.
17. The Media Client establishes the transport session(s) to acquire the media content.

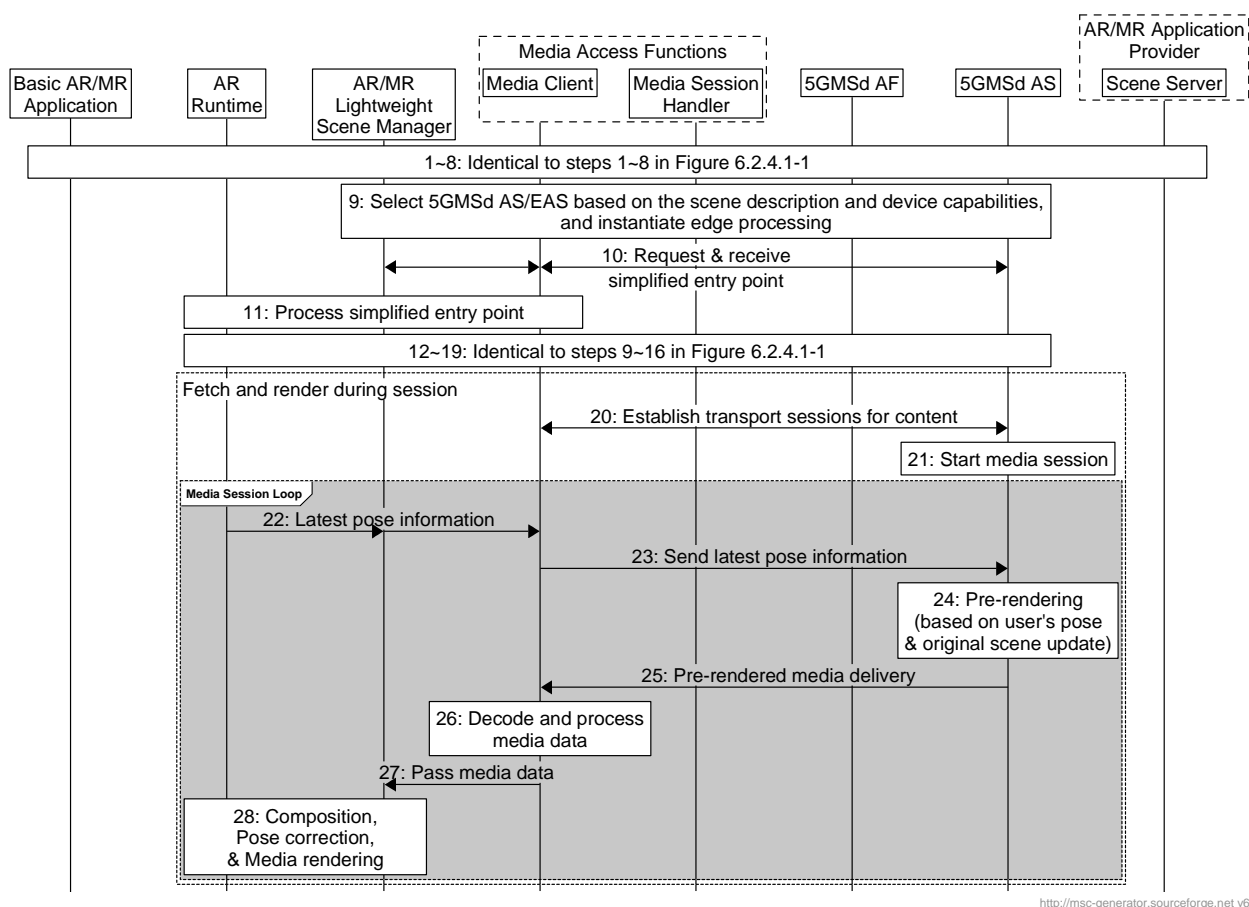
Media session loop, steps 18~23 provide the latest pose information, request, receive and render the media objects of the immersive scene:

18. The latest pose information is acquired by the AR/MR Scene Manager and shared to the Media Client.
19. The Media Client requests the immersive media data according to the delivery manifest processed, possibly taking into account pose information (e.g., viewport dependent streaming).

20. The Media Client receives the immersive media data and triggers the media rendering pipeline(s), including the registration of AR content into the real world accordingly.
21. The Media Client decodes and processes the media data. For encrypted media data, the Media Client may also perform decryption.
22. The Media Client passes the media data to the AR/MR Scene Manager.
23. The AR/MR Scene Manager renders the media, and passes the rendered media to the AR Runtime, which performs further processing such as registration of the AR content into the real world, and pose correction.

6.2.4.2 EDGAR-based media streaming

Figure 6.2.4.2-1 illustrates the procedure diagram for 5G immersive media downlink streaming using an EDGAR-based UE.



<http://msc-generator.sourceforge.net v6.4.7>

Figure 6.2.4.2-1: EDGAR-based procedure for 5G downlink streaming

Prerequisites and assumptions:

- Identical to those from the STAR UE case.

Procedures:

- 1-8. Identical to steps 1-8 from the STAR UE case (figure 6.2.4.1-1).
9. Based on the processed scene description and the device capabilities, the 5GMSd AS/EAS is selected, and edge processes are instantiated using the processes defined in 5GMS_EDGE:
 - a. The AR/MR Lightweight Scene Manager sends the scene description and the device capabilities to 5GMS AS. The 5GMS AS derives the EAS KPIs and if needed selects a new AS/EAS (through AF) based on the new KPI. Then the edge processes are started, and a new entry point URL is provided to the AR/MR Lightweight Scene Manager.

- b. The AR/MR Lightweight Scene Manager derives the EAS KPIs from the scene description and device capabilities, requests the AF to provide the list of suitable EAS. Then the AR/MR Lightweight Scene Manager selects the AS/EAS and requests to start the edge processes in the EAS. The edge processes are started, and a new entry point URL is provided to the AR/MR Lightweight Scene Manager.
10. The AR/MR Lightweight Scene Manager requests the lightweight scene description. The edge processes derive the lightweight scene description from the full scene description and provide it to AR/MR Lightweight Scene Manager.
11. The simplified entry point (lightweight scene description) is processed.
- 12~19. Identical to steps 9~16 from the STAR UE case (figure 6.2.4.1-1).
20. The Media Client establishes the transport session(s) to acquire the media content.
21. The 5GMSd AS initiates and starts a media session. This media session forms a stateful session loop specific to the UE, containing steps 22~25:

Stateful media session loop (steps 22~28):

- 22. The latest pose information is acquired by the AR/MR Lightweight Scene Manager and shared to the Media Client.
- 23. The Media Client sends the latest pose information to the 5GMSd AS.
- 24. The 5GMSd AS performs pre-rendering of the media based on the latest received pose information and possibly any original scene update. Pre-rendering may typically consist of decoding and rendering immersive media, and encoding the rendered (2D) media.
- 25. The pre-rendered media is sent by the 5GMSd AS to the Media Client.
- 26. The Media Client decodes and processes the media data. For encrypted media data, the Media Client may also perform decryption.
- 27. The Media Client passes the media data to the AR/MR Lightweight Scene Manager.
- 28. The AR/MR Lightweight Scene Manager renders the media, and passes the rendered media to the AR Runtime, which performs further processing such as registration of the AR content into the real world, composition, and pose correction.

6.2.5 Content formats and codecs

Based on the use cases, the following formats, codecs, and packaging formats are of relevance for media streaming of AR:

- General
 - > Basic scene graph and scene description
 - > 2D uncompressed video formats and video compression codecs
 - > Regular audio formats and audio compression codecs
- In addition, for STAR-based UE
 - > Richer scene graph data
 - > 3D formats such as static and dynamic point clouds or meshes
 - > Several video decoding instances
 - > Decoding tools for such formats
 - > DASH/CMAF based delivery
- In addition, for EDGAR-based UE

- > 2D compression tools for eye buffers as defined in clause 4.5.2
- > Decoding tools for such formats
- > At least two video decoding instances
- > Low-latency downlink real-time streaming of the above media
- > Uplink streaming of pose information and other relevant information, such as input actions

6.2.6 KPIs and QoS

The above scenarios relate to the following cases in TR 26.928 [2], clause 6. In particular:

- For STAR:
 - > Viewport-independent streaming based on clause 6.2.2 as defined TR 26.928 [2],
 - > Viewport-dependent streaming based on clause 6.2.3 as defined TR 26.928 [2],
- For EDGAR:
 - > Raster-based split rendering based on clause 6.2.5 as defined TR 26.928 [2].

For STAR-based devices and viewport-independent streaming, processing of updated pose information is only done locally in the XR device. Delivery latency requirements are independent of the motion-to-photon latency. Initial considerations on QoE parameters are provided in TR 26.928 [2], clause 6.2.2.5. The XR media delivery are typically built based on download or adaptive streaming such as DASH such that one may adjust quality to the available bitrate to a large extent. Compared to the viewport independent delivery, for viewport dependent streaming, updated tracking and sensor information impacts the network interactivity. Typically, due to updated pose information, HTTP/TCP level information and responses are exchanged every 100-200ms. For more details, refer to clause 6.2.3 as defined TR 26.928 [2]. Such approaches may reduce the required bitrate compared to viewport independent streaming by a factor of 2 to 4 at the same rendered quality. It is important to note that viewport-dependent streaming technologies are typically also built based on adaptive streaming allowing to adjust quality to the available bitrate. The knowledge of tracking information in the XR Delivery receiver just adds another adaptation parameter. However, generally such systems may be flexibly designed taking into account a combination/tradeoff of bitrates, latencies, complexity and quality. Suitable 5QI values for adaptive streaming over HTTP are 6, 8, or 9 as defined in TS 23.501 [55], clause 5.7.4 and also indicated in clause 4.3.3 of TR 26.928 [2].

For EDGAR-based devices, raster-based split rendering based on clause 6.2.5 as defined TR 26.928 [2] applies. With the use of pose corrections, the key latency for the network is the motion-to-render-to-photon delay as introduced in clause 4.5.2 and 4.5.3, i.e. the end-to-end latency between the user motion and the rendering is 50-60ms. The formats are defined in clause 4.5.2 as follows

- for 30 x 20 degrees, 1.5K by 1K per eye is required and 1.8K by 1.2K per eye is desired
- for 40 x 40 degrees, 2K by 2K required and 2.5 K by 2.5 K desired

Colours are typically RGB but may be converted to YUV. Framerates are typically 60fps to 90fps. The above formats result in typically in maximum 4K content at 60 fps. Modern compression tools compress this to 30 to 50 Mbit/s. Regular stereo audio signals are considered, requiring bitrates that are negligible compared to the video signals. In order to support warping and late stage reprojection, some depth information may be added. For communication a real-time capable content delivery protocol is needed, and the network needs provide reliable delivery mechanisms. 5QI values exist that may address the use case, such 5QI value number 80 with 10ms, however this is part of the non-GBR bearers (see clause). In addition, it is unclear whether the 10ms with such high bitrates and low required error rates may be too stringent and resource consuming.

Hence, for simple split rendering in the context of the requirements in this clause, suitable 5QIs 89 and 90 have been defined in Rel-17 in TS 23.501 in Rel-17 addressing the latency requirements in the range of 10-20ms and bitrate guarantees to be able to stream up to 50 Mbps consistently. Significant opportunities exist to support split rendering with advanced radio tools, see for example TR 26.926 [48] for performance evaluation.

The uplink is predominantly the pose information. Data rates are several 100 kbit/s and the latency need to be small in order to not add to the overall target latency. Suitable 5QIs 87 and 88 have been defined in Rel-17 in TS 23.501 to stream uplink pose information.

6.2.7 Standardization areas

The list of potential standardization area that has been collected is provided in the following:

- HTTP-Streaming of immersive scenes with 2D and 3D media formats and objects to STAR-based devices including
 - > Immersive media format and profile with integration into 5GMS for STAR-based devices
 - > Scene description format, functionality, and profile as an entry point of immersive media
 - > Relevant subset of media codecs for different media types and formats
 - > CMAF encapsulation of immersive media for 5G media streaming
 - > Viewport independent and viewport dependent streaming
- Split rendering delivery of immersive scenes to EDGAR-based devices
 - > Media payload format to be mapped into RTP streams
 - > Capability exchange mechanism and relevant signalling
 - > Protocol stack and content delivery protocol
 - > Cross-layer design, radio and 5G system optimizations for QoS support
 - > Uplink streaming of predicted pose information and input actions
- Required QoE metric

6.3 Interactive immersive services

6.3.1 Introduction

This clause introduces the case where interactive immersive service. In this case, pose and other interactions are sent uplink for the Interactive Immersive Server to render the scene accordingly.

6.3.2 Relevant use cases

The following use cases are relevant to this scenario.

- UC#1: 3D Image Messaging
- UC#2: AR Sharing
- UC#4: AR guided assistant at remote location (industrial services)
- UE#5: Police Critical Mission with AR
- UE#15: 5G Shared Spatial Data
- UE#16: AR remote cooperation
- UC#21: AR gaming

In this scenario, a user interaction is sent from a UE to a server, so that the server handles the user's request to the immersive media scene (e.g., changing the context such as translation, rotation, and scaling). The processed scene is sent back to a UE in a similar manner of immersive media streaming case.

6.3.3 Architectures

6.3.3.1 STAR-based

Figure 6.3.3.1-1 provides an architecture for immersive interactive media distribution using a STAR UE.

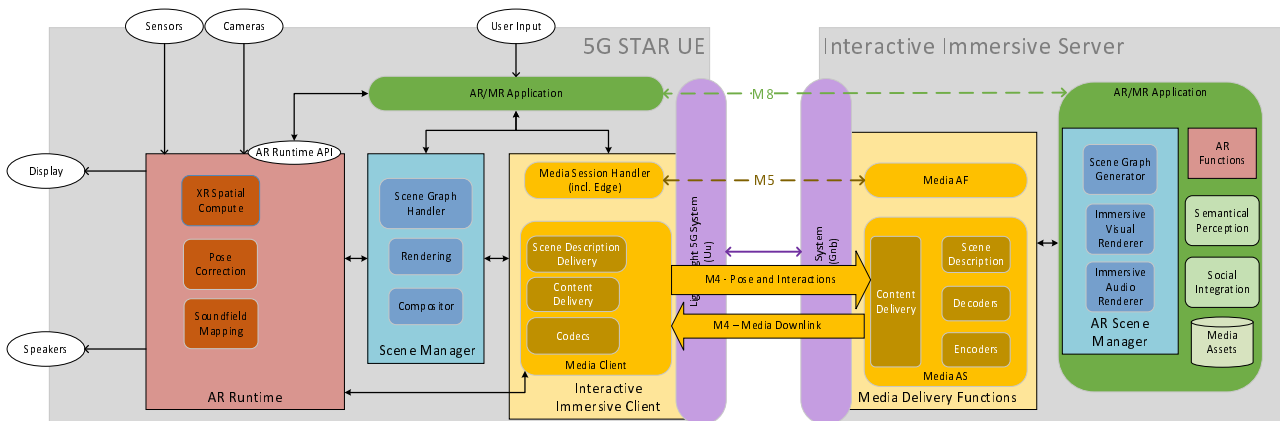


Figure 6.3.3.1-1: STAR-based 5G interactive immersive service architecture

6.3.3.2 EDGAR-based

Figure 6.3.3.2-1 provides an architecture for Interactive Immersive Media distribution using a EDGAR UE. In this case, similar as before, most of the rendering needs to be accomplished on the server.

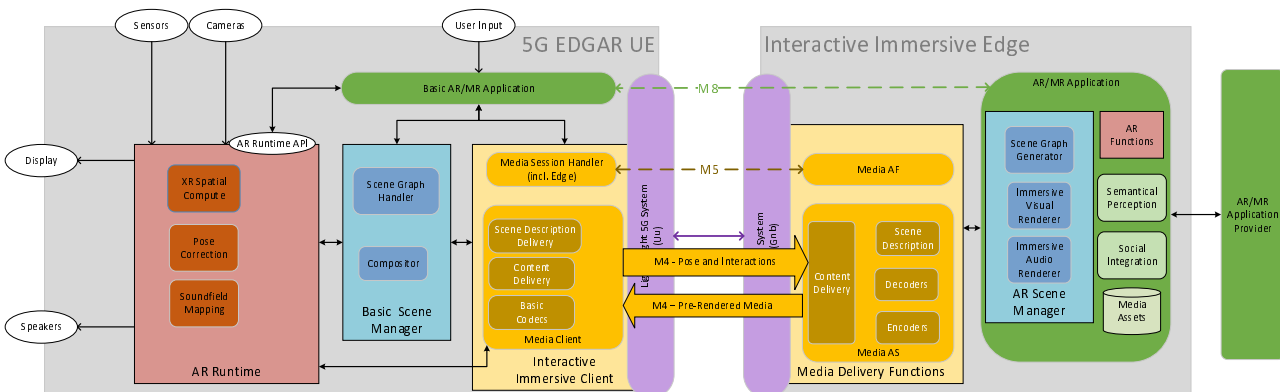


Figure 6.3.3.2-1: EDGAR-based 5G interactive immersive service architecture

6.3.4 Procedures and call flows

6.3.4.1 STAR-based interactive immersive service

Figure 6.3.4.1-1 illustrates the procedure diagram for interactive immersive services using a STAR-based UE when all essential AR/MR functions in a UE are available without an assist by an edge.

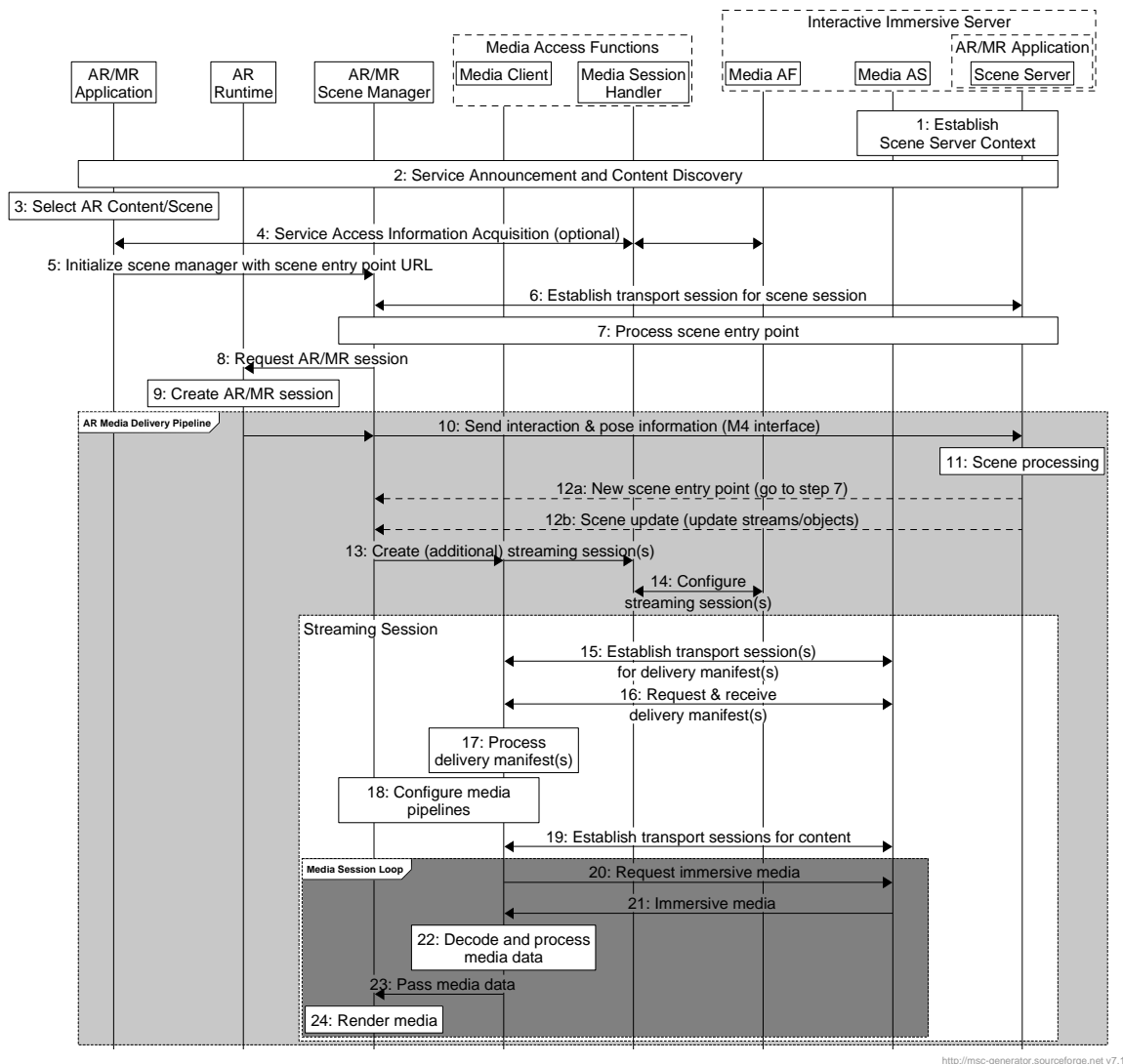


Figure 6.3.4.1-1: STAR-based procedure for interactive immersive service

Prerequisites and assumptions:

- The AR/MR Scene Manager includes immersive media rendering and scene graph handling functionalities.
- The Media Player includes immersive content delivery and immersive media decoding functionalities.
- The AR/MR Application in the UE is run by the user.
- The STAR UE initialises AR registration (starts analysing the surroundings where a user/UE is located), it namely:
 - a. captures its surroundings via camera(s)
 - b. analyses where the device is located
 - c. registers the device into the analysed surroundings.
- AR/MR Application and AR/MR Application Provider have exchanged some information, such as device capability or content configuration, for content rendering. The exchange procedures for device capability and content configuration are FFS.
- AR/MR Application Provider has established a Provisioning Session and its detailed configurations has been exchanged.
- AR/MR Application Provider has completed to set up ingesting immersive contents.

Procedures:

1. The Scene Server context is established, and scene content is ingested by the Media AS.
2. Service Announcement is triggered by AR/MR Application. Service Access Information including Media Client entry or a reference to the Service Access Information is provided through the M8d interface.
3. Desired media content is selected.
4. Optionally, the Service Access information is acquired or updated.
5. The AR/MR Application initializes the Scene Manager with the entry point (full scene description) URL.
6. The Media Client establishes the transport session for the scene session between the Scene Manager in the UE and the Scene Server.
7. The Media Client requests and receives the full scene description. The entry point (scene description) is processed by the AR/MR Scene Manager, and a scene session is created.
8. The AR/MR Scene Manager requests the creation of a new AR/MR session from the AR Runtime.
9. The AR Runtime creates a new AR/MR session.

AR Media Delivery Pipeline, steps 10~24, send the interaction and pose information and receives and renders the updated scenes accordingly:

10. The latest interaction and pose information are acquired by the AR/MR Scene Manager and shared to the Media Client. The Media Client sends this information to the Media AS and Scene Server.
 11. The Scene Server processes the scene according to the interaction and pose information from the UE. Depending on the level of processing, the current scene may be updated or replaced.
 12. When needed, one of the following steps:
 - 12a. The Scene Server sends a new scene entry points to the AR/MR Scene Manager through the Media AS and Media Client (go to step 7), or
 - 12b. The Scene Server sends a scene update (updating streams/objects) to the AR/MR Scene Manager through the Media AS and Media Client.
 13. The AR/MR Scene requests to create additional streaming sessions if needed for new media objects in the scene.
- NOTE: The number of the additional streaming sessions depends on the delivery mechanism. One or more media objects may be delivered through a single manifest and/or use of the same connection. Therefore, introduction of every new media object may not need an additional streaming session.
14. The Media Session Handle establishes the additional streaming sessions based on the received request.

Streaming session, steps 15~18 establish the transport sessions for media objects and configure the media pipelines

15. For the required media content, the Media Client establishes the transport session(s) to acquire delivery manifest(s) information.
16. The Media Client requests and receives the delivery manifest(s) from the Media AS.
17. The Media Client processes the delivery manifest(s). It determines for example the number of needed transport sessions for media acquisition. The Media Client is expected to be able to use the delivery manifest(s) information to initialize the media pipelines for each media stream.
18. The AR/MR Scene Manager and Media Client configures the rendering and delivery media pipelines.
19. The Media Client establishes the transport session(s) to acquire the media content.

Media session loop includes steps 20~24 which are for streaming, decoding and rendering media components:

20. The Media Client requests the immersive media data according to the delivery manifest processed, possibly taking into account pose information (e.g., viewport dependent streaming).

21. The Media Client receives the immersive media data and triggers the media rendering pipeline(s), including the registration of AR content into the real world accordingly.
22. The Media Client decodes and processes the media data. For encrypted media data, the Media Client may also perform decryption.
23. The Media Client passes the media data to the AR/MR Scene Manager.
24. The AR/MR Scene Manager renders the media, and passes the rendered media to the AR Runtime, which performs further processing such as registration of the AR content into the real world, and pose correction.

6.3.4.2 EDGAR-based interactive immersive service

Figure 6.3.4.2-1 illustrates the procedure diagram for interactive immersive services using an EDGAR-based UE.

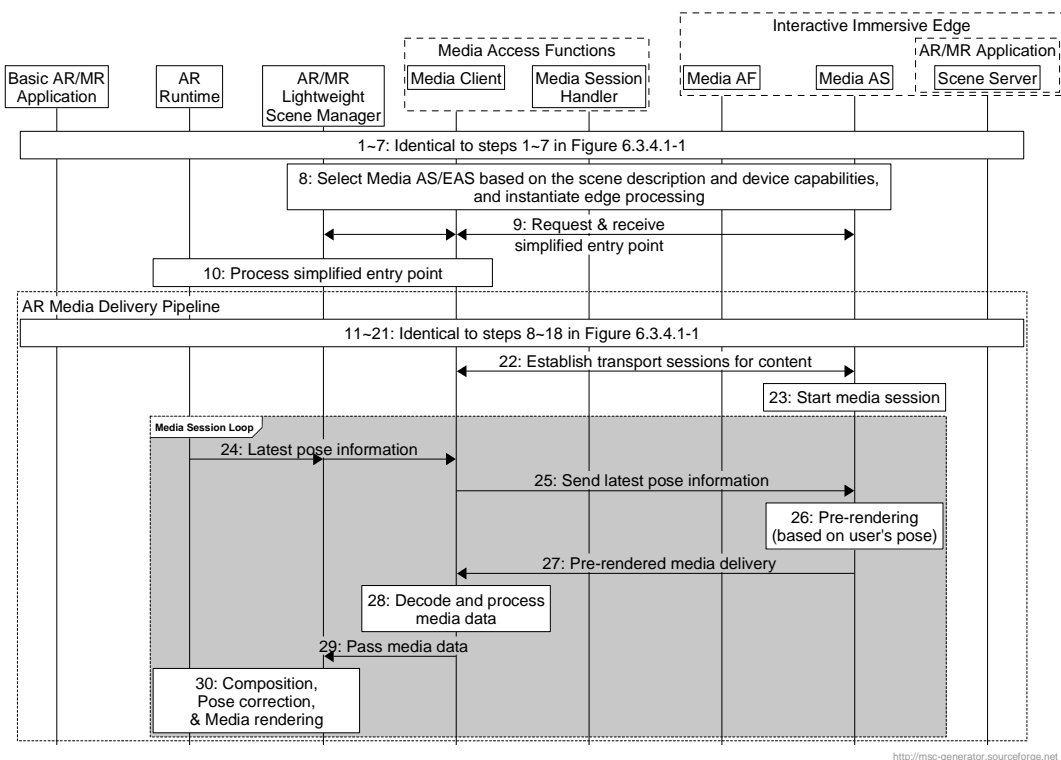


Figure 6.3.4.2-1: EDGAR-based procedure for interactive immersive service

Prerequisites and assumptions:

- Identical to those from the STAR UE case.

Procedures:

- 1~7. Identical to steps 1~7 from the STAR UE case (figure 6.3.4.1-1).
8. Based on the processed scene description and the device capabilities, the Media AS/EAS is selected, and edge processes are instantiated using the processes defined in EDGE:
 - a. The AR/MR Lightweight Scene Manager sends the scene description and the device capabilities to Media AS. The Media AS derives the EAS KPIs and if needed selects a new AS/EAS (through AF) based on the new KPI. Then the edge processes are started, and a new entry point URL is provided to the AR/MR Lightweight Scene Manager.
 - b. The AR/MR Lightweight Scene Manager derives the EAS KPIs from the scene description and device capabilities, requests the AF to provide the list of suitable EAS. Then the AR/MR Lightweight Scene Manager selects the AS/EAS and requests to start the edge processes in the AS. The edge processes are started, and a new entry point URL is provided to the AR/MR Lightweight Scene Manager.

9. The AR/MR Lightweight Scene Manager requests the lightweight scene description. The edge processes derive the lightweight scene description from the full scene description and provide it to AR/MR Scene Manager.

10. The simplified entry point (lightweight scene description) is processed.

11~21. Identical to steps 8~18 from the STAR UE case (figure 6.3.4.1-1).

NOTE: After step 15a (12a in Figure 6.3.4.1-1): go to step 10

22. The Media Client establishes the transport session(s) to acquire the media content.

23. The Media AS initiates and starts a media session. This media session forms a stateful session loop specific to the UE, containing steps 25~28:

Stateful media session loop (steps 24~30):

24. The latest pose information is acquired by the AR/MR Lightweight Scene Manager and shared to the Media Client.

25. The Media Client sends the latest pose information to the Media AS.

26. The 5GMSd AS performs pre-rendering of the media based on the latest received pose information. Pre-rendering may typically consist of decoding and rendering immersive media, and encoding the rendered (2D) media.

27. The pre-rendered media is sent by the Media AS to the Media Client.

28. The Media Client decodes and processes the media data. For encrypted media data, the Media Client may also perform decryption.

29. The Media Client passes the media data to the AR/MR Lightweight Scene Manager.

30. The AR/MR Lightweight Scene Manager renders the media, and passes the rendered media to the AR Runtime, which performs further processing such as registration of the AR content into the real world, composition, and pose correction.

6.3.5 Content formats and codecs

Based on the use cases, the following formats, codecs and packaging formats are of relevance for interactive immersive media distribution of AR:

- Scene graph and scene description
- 2D video formats
- 3D formats such as static and dynamic point clouds or meshes
- 2D video formats with depth
- Regular audio formats
- Several video decoding instances
- Decoding tools for such formats
- Low-latency downlink real-time streaming of the above media
- Uplink streaming of pose information and interaction data

6.3.6 KPIs and QoS

The above scenarios relate to the following cases in TR 26.928 [2], clause 6. In particular:

- For STAR:
 - > Viewport-dependent streaming based on clause 6.2.3 as defined TR 26.928 [2],

- > Raster-based split rendering based on clause 6.2.5 as defined TR 26.928 [2],
- > Generalized XR split rendering based on clause 6.2.6 as defined TR 26.928 [2].
- For EDGAR:
 - > Raster-based split rendering based on clause 6.2.5 as defined TR 26.928 [2].

For STAR-based delivery, a basic architecture as shown in Figure 6.3.6-1 applies. Two important latency considerations are important:

- User interaction latency, i.e. the time duration between the moment at which a user action is initiated and the time such an action is taken into account by the stage performer or content creation engine. In the context of gaming, this is the time between the moment the user interacts with the game and the moment at which the game engine processes the player's response.
- End-to-End Latency (EEL): The latency for an action that is originally presented in the scene or captured by camera until its visibility on the remote display.
- Round-trip Interaction Delay (RID): The time of an action by the user until it sees the action reflected on its screen. This delay is the sum of the user interaction delay and End-to-End Latency.

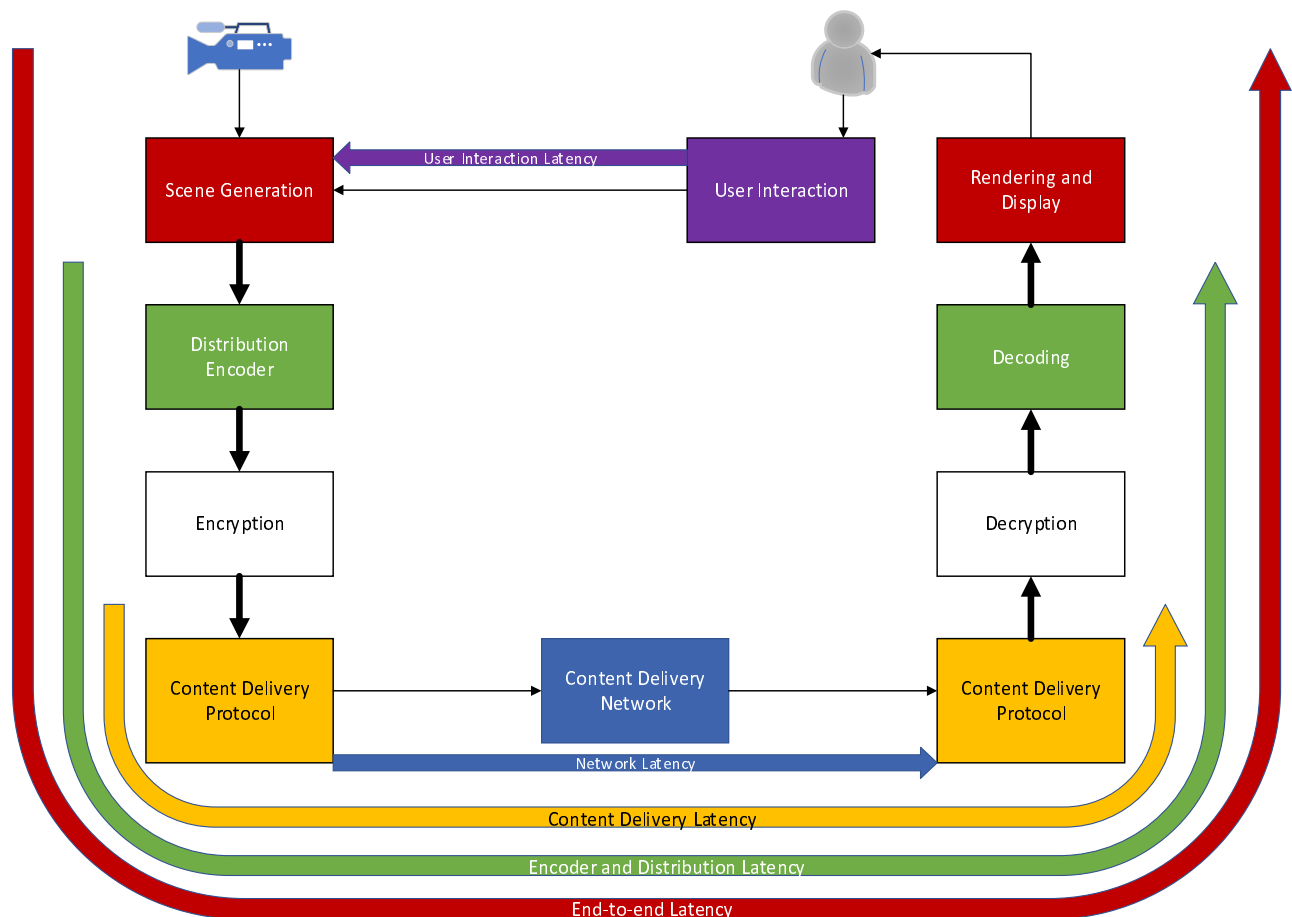


Figure 6.3.6-1: Architecture and latencies for interactive immersive service

The maximum RID depends on the type of scene. A typical example is the Stadia cloud gaming platform and an excellent introduction is provided here [52]. Some extracted high-level requirements on user experience for RID are provided between time 700 to 800 ms [52].

Similar data is collected in TR 26.928 [2], clause 4.5. Typically, systems have maximum delay requirements between 60ms and 500ms. In terms of formats and bitrates, similar considerations as for clause 6.2.6 apply. However, note that in many cases a pre-rendering is applied in the network, such that data rates and formats are more similar to the split-rendering considerations. Similar considerations as for clause 6.2.6 apply on raster-based split rendering.

For EDGAR-based devices, raster-based split rendering based on clause 6.2.5 as defined TR 26.928 [2] applies. Similar considerations as for clause 6.2.6 apply.

The uplink is predominantly the pose information and user interactions. Data rates are several 100 kbit/s and the latency need to be small in order to not add to the overall target latency.

6.3.7 Standardization areas

The list of potential standardization area that has been collected is provided in the following:

- Streaming of immersive scenes with 2D and 3D media formats and objects to STAR-based devices including
 - > Low-latency streaming protocols to support latencies in the range between 50 to 500ms, typically using RTP-based real-time streaming based on cloud rendering
 - > Scene description format, functionality, and profile as an entry point of immersive media
 - > Simplified 3D media formats and 2D media formats with integration for STAR-based devices
 - > Relevant subset of media codecs for different media types and formats
 - > RTP encapsulation of media formats
 - > 5G System and 5G Media Streaming support
- Split rendering delivery of immersive scenes to EDGAR-based devices
 - > Simple 2D media formats that match AR glass display capabilities
 - > Media payload format to be mapped into RTP streams
 - > Capability exchange mechanism and relevant signalling
 - > Protocol stack and content delivery protocol
 - > Cross-layer design, radio and 5G system optimizations for QoS support
 - > Uplink streaming of predicted pose information
- Required QoE metrics

6.4 5G cognitive immersive service

6.4.1 Introduction

This clause introduces the case of cognitive immersive service. In this case, media and other interactions are sent uplink in order for the cognitive server to create semantical perception.

6.4.2 Relevant use cases

The following use cases are relevant to this scenario.

- UC#4: AR guided assistant at remote location (industrial services)
- UE#5: Police Critical Mission with AR
- UE#14: AR Streaming with Localization Registry
- UE#16: AR remote cooperation
- UC#20: AR IoT control

In this scenario, a media captured in a UE may be sent to a cognitive server to request semantical perception. The server processes and outputs the perception results, then responds the outputs to the UE. For example, a UE regularly scans

his/her environments and sends the captured media such as video, depth-maps, sensor output, and XR Spatial Description data (if SLAM and XR Spatial Compute processing is involved) to the cognitive server. The server identifies each component in the environments and sends back to the UE the identified perception outputs so that the UE may render in textual or visual overlays. The server may also send to the UE XR Spatial Description data, such as spatial anchors and trackables, in order to facilitate rendering.

6.4.3 Architectures

6.4.3.1 STAR-based

Figure 6.4.3.1-1 provides an architecture for immersive interactive media distribution using a STAR UE.

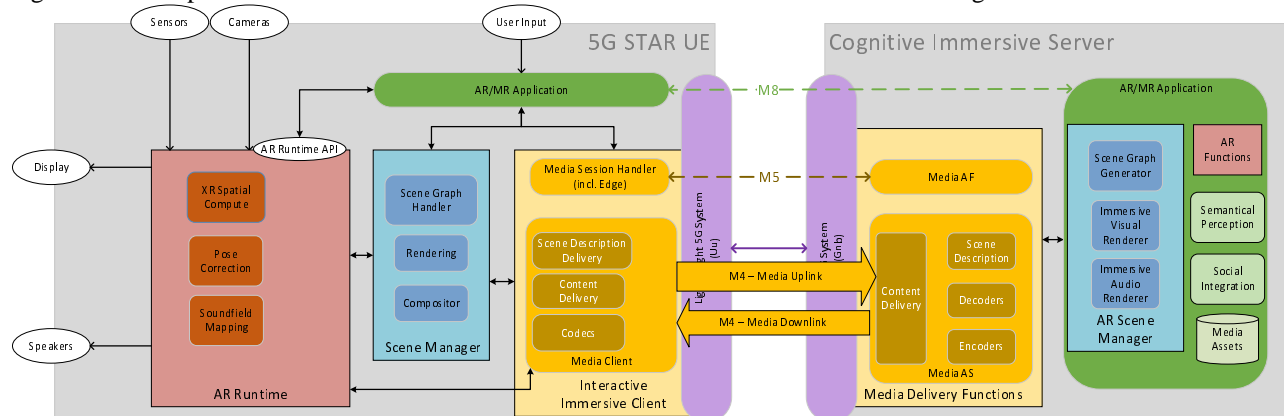


Figure 6.4.3.1-1: STAR-based 5G cognitive immersive service architecture

6.4.3.2 EDGAR-based

Figure 6.4.3.2-1 provides an architecture for Cognitive Immersive Media distribution using an EDGAR UE. In this case, similar as before, most of the rendering needs to be accomplished on the server.

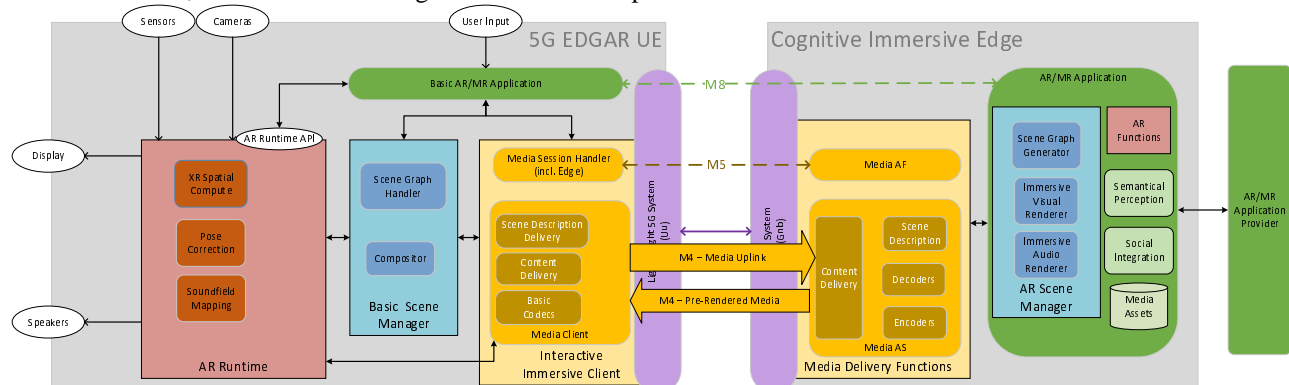


Figure 6.4.3.2-1: EDGAR-based 5G cognitive immersive service architecture

6.4.4 Procedures and call flows

Figure 6.4.4-1 illustrates the generic procedure diagram for cognitive immersive services for both STAR-based and EDGAR-based UEs.

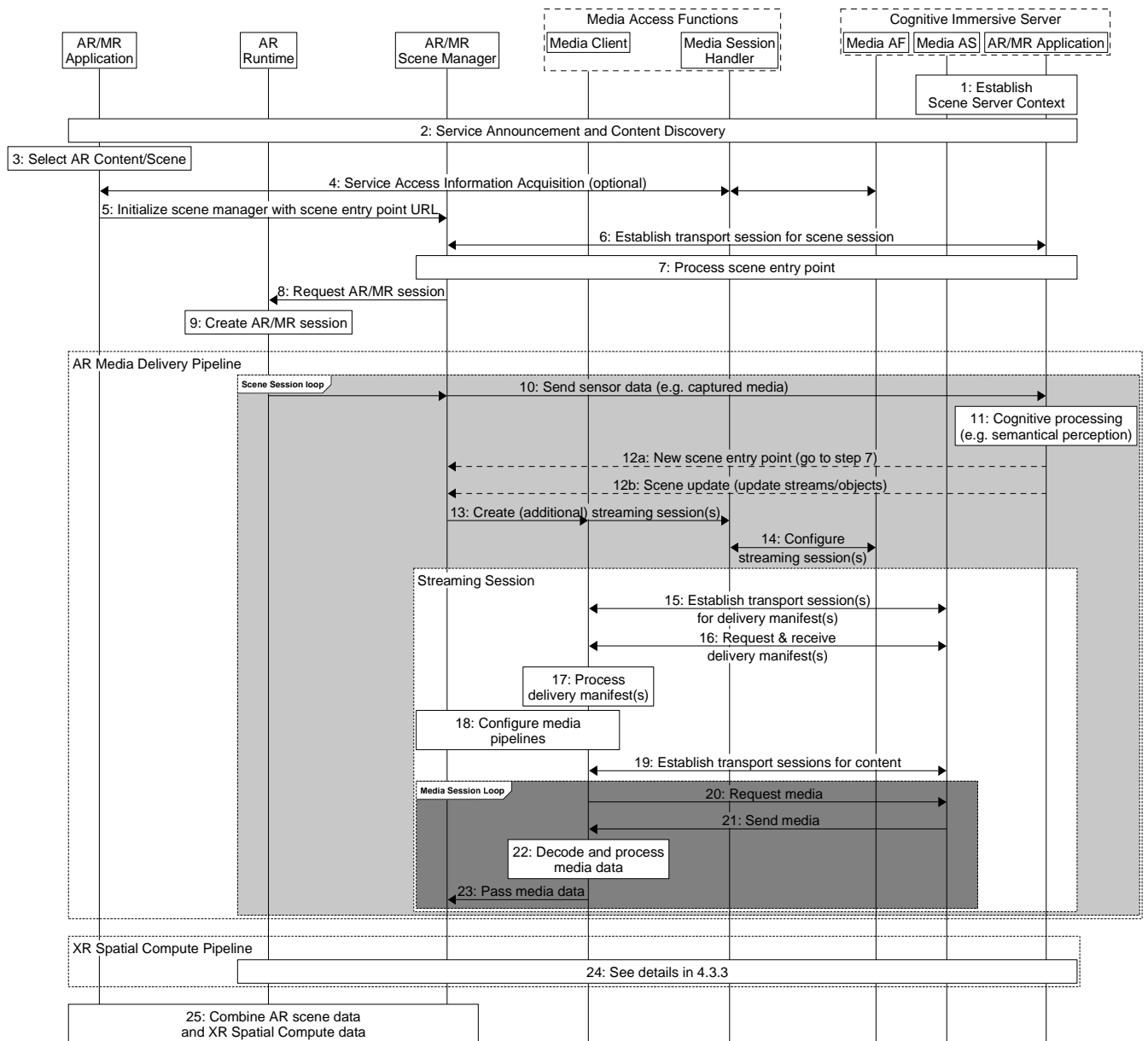

<http://msc-generator.sourceforge.net v7.1>

Figure 6.4.4-1: Generic procedure for cognitive immersive service

Prerequisites and assumptions:

- The AR/MR Scene Manager includes immersive media rendering and scene graph handling functionalities.
- The Media Player includes immersive content delivery and immersive media decoding functionalities.
- The AR/MR Application in the UE is run by the user.
- The UE initialises AR registration (starts analysing the surroundings where a user/UE is located), it namely:
 - a. captures its surroundings via camera(s)
 - b. analyses where the device is located
 - c. registers the device into the analysed surroundings.
- AR/MR Application and AR/MR Application Provider have exchanged some information, such as device capability or content configuration, for content rendering. The exchange procedures for device capability and content configuration are FFS.

- AR/MR Application Provider has established a Provisioning Session and its detailed configurations has been exchanged.
- AR/MR Application Provider has completed to set up ingesting immersive contents.

Procedures:

1. The Scene Server context is established, and scene content is ingested by the Media AS.
2. Service Announcement is triggered by AR/MR Application. Service Access Information including Media Client entry or a reference to the Service Access Information is provided through the M8d interface.
3. Desired media content is selected.
4. Optionally, the Service Access information is acquired or updated.
5. The AR/MR Application initializes the Scene Manager with the entry point (full scene description) URL.
6. The Media Client establishes the transport session for the scene session between the Scene Manager in the UE and the Scene Server.
7. The Media Client requests and receives the full scene description. The entry point (scene description) is processed by the AR/MR Scene Manager, and a scene session is created.
8. The AR/MR Scene Manager requests the creation of a new AR/MR session from the AR Runtime.
9. The AR Runtime creates a new AR/MR session.

Scene session loop, steps 10~24, send the interaction and pose information and receives and renders the updated scenes accordingly:

10. The latest sensor data (e.g. captured media) is acquired by the AR/MR Scene Manager and shared with the Media Client. The Media Client sends this information to the Media AS and AR/MR Application.
 11. The AR/MR Application performs cognitive processing according to the sensor data from the UE. Depending on the outcome, the current scene may be updated or replaced.
 12. When needed, one of the following steps:
 - 12a. The Scene Server sends a new scene entry point to the AR/MR Scene Manager through the Media AS and Media Client (go to step 7), or
 - 12b. The Scene Server sends a scene update (updating streams/objects) to the AR/MR Scene Manager through the Media AS and Media Client.
 13. The AR/MR Scene requests to create additional streaming sessions if needed for new media objects in the scene.
- NOTE: The number of the additional streaming sessions depends on the delivery mechanism. One or more media objects may be delivered through a single manifest and/or use of the same connection. Therefore, introduction of every new media object may not need an additional streaming session.
14. The Media Session Handle establishes the additional streaming sessions based on the received request.

Streaming session, steps 15~18 establish the transport sessions for media objects and configure the media pipelines

15. For the required media content, the Media Client establishes the transport session(s) to acquire delivery manifest(s) information.
16. The Media Client requests and receives the delivery manifest(s) from the Media AS.
17. The Media Client processes the delivery manifest(s). It determines for example the number of needed transport sessions for media acquisition. The Media Client is expected to be able to use the delivery manifest(s) information to initialize the media pipelines for each media stream.
18. The AR/MR Scene Manager and Media Client configures the rendering and delivery media pipelines.
19. The Media Client establishes the transport session(s) to acquire the media content.

Media session loop includes steps 20~24 which are for streaming, decoding and rendering media components:

20. The Media Client requests the media data according to the delivery manifest processed, possibly taking into account pose information (e.g., viewport dependent streaming).
21. The Media Client receives the media data and triggers the media rendering pipeline(s), including the possible registration of AR content into the real world accordingly (depending on the device type).
22. The Media Client decodes and processes the media data. For encrypted media data, the Media Client may also perform decryption.
23. The Media Client passes the media data to the AR/MR Scene Manager.
24. The XR Spatial Compute Pipeline as specified in clause 4.3.3.
25. The AR scene data and XR Spatial Compute data are combined for composition and rendering.

6.4.5 Content formats and codecs

Based on the use cases, the following formats, codecs and packaging formats are of relevance for cognitive immersive media distribution of AR:

- Scene graph and scene description
- Spatial description
- 2D video formats
- 3D formats such as static and dynamic point clouds or meshes
- 2D video formats with depth
- Regular audio formats
- Several video decoding instances
- Decoding tools for such formats
- Encoding tools for 2D formats
- Low-latency downlink and uplink real-time streaming of the above media
- Uplink streaming of pose information
- Uplink streaming of media

6.4.6 KPIs and QoS

In the downlink this scenario is equivalent to clause 6.3.6 and similar KPIs and QoS aspects apply.

For the uplink, the above scenarios relate to the following cases in TR 26.928 [2], clause 6. In particular:

- XR distributed computing based on clause 6.2.7 as defined TR 26.928 [2].

Details on uplink streaming of sensor data are for future study.

6.4.7 Standardization areas

The list of potential standardization area that has been collected is provided in the following:

- Similar functionalities as identified in clause 6.3.7 for downlink
- For the uplink, streaming of sensor information to the network

- > Low-latency streaming protocols to support latencies in the range between 50 to 500ms, typically using RTP-based real-time streaming
- > Simple 2D media formats to stream match AR sensor data
- > Payload format to be mapped into RTP streams
- > Capability exchange mechanism and relevant signalling
- > Protocol stack and content delivery protocol
- > Cross-layer design, radio and 5G system optimizations for QoS support
- Spatial description format for downlink and uplink
- Required QoE metrics

6.5 AR conversational services

6.5.1 Introduction

AR Conversational services are end-to-end use-cases that include communication between two or more parties. The following building blocks to realize AR conversational services are identified:

- a) Call setup and control: this building block covers the
 - signalling to setup a call or a conference.
 - fetching of the entry point for the AR experience. The protocol needs to support upgrading and downgrading to/from an AR experience. It also needs to support adding and removing media. This also includes the device type (Type-1, Type-2, or Type-3) as well as non-AR experience, e.g., tablet.
- b) Formats: The media and metadata types and formats for AR calls need to be identified. The format for the entry point, namely the scene description, and any extensions to support AR telephony need to be identified. Also, the format for media capturing, e.g., 2D video, depth map, 3D point clouds, colour attributes, etc. need to be identified. For AR telephony media types, the necessary QoS characteristics need to be defined, as well as format properties and codecs.
- c) Delivery: the transport protocols for the AR media need to be identified. AR telephony and conferencing applications require low latency exchange of real-time media. A protocol stack, e.g. based on RTP, will be required.
- d) 5G system integration: offering the appropriate support by the 5G system to AR telephony and conferencing applications includes:
 - signalling for QoS allocation,
 - discovery and setup of edge resources to process media for AR telephony,
 - usage of MBS and MTSI,
 - data collection and reporting.

The building blocks may have different instantiations and/or options. For example, the delivery may be mapped to a WebRTC protocol stack or to an MTSI protocol stack. Furthermore, a single session may combine several delivery methods to accommodate the different media types supported by an AR conversational service.

In addition, AR telephony and conferencing applications may support asymmetrical and symmetrical experiences. In an asymmetrical case, one party is sending AR immersive media and the backchannel from other participants may be audio only, 2D video, etc. In a symmetrical case, all involved parties are sending and receiving AR immersive media.

6.5.2 Relevant use cases

The use case relevant to this scenario may be further categorized into AR two-party call use cases and AR conferencing use cases. The AR two-party call use cases include:

- UC#3: Real-time 3D Communication
- UC#4: AR guided assistant at remote location (industrial services)
- UC#7: Real-time communication with the shop assistant
- UC#11: AR animated avatar calls
- UC#16: AR remote cooperation
- UC#19: AR conferencing

The AR conferencing use cases include:

- UC#8: 360-degree conference meeting
- UC#9: XR meeting
- UC#10: Convention / Poster Session
- UC#12: AR avatar multi-party calls
- UC#13: Front-facing camera video multi-party calls
- UC#19: AR conferencing

3GPP TR 22.873 [14] also addresses use cases relevant to this scenario, namely conference call with AR holography and AR call. Note that the first use case has similarity with the UC#19 and the second use case has similarity with the UC#4 as listed in Table 5-1.

6.5.3 Basic architecture and call flows

There are different options for mapping to 5G system:

- a) The MTSI architecture (TS 26.114 [15]) supports audio and 2D video conversational services. Extending the MTSI architecture to support AR signalling and immersive media. This includes both MTSI/RTP and MTSI/Data channel (DC) stack options.
- b) Extending the 5GMS architecture (TS 26.501 [26]) to support AR conversational services by combining live uplink and live downlink. 5GMS offers basic functionality such as QoS support, reporting, and in the future also edge, which will be beneficial for all types of applications. The typical/expected QoS parameters (especially delay) need to be clarified.
- c) An architecture based on something different than MTSI / IMS or 5GMS, for example, browser implementations such as WebRTC. WebRTC is widely deployed today for conversational services and is built on flexible ecosystem on the device side, which is important in this case since conversational AR will require significant device-side changes.

Table 6.5.3-1: Comparison of different architecture options for supporting AR conversational services

Component	MTSI: RTP and DC (TS 26.114)	5GMS/HTTP (26.501)	Other architecture (e.g. WebRTC-based)
Protocol	SIP- and RTP-based or DC-based. SDP signalling and formats for AR are missing and need to be defined. Encoding and decoding at MTSI client needs to be extended beyond ITT4RT with improved support of AR immersive media formats (e.g. meshes, point clouds).	TCP- and HTTP-based streaming, using DASH/HLS and MPEG OMAF/CMAF technology. AR & immersive media content and signalling are assumed to work with HTTP-based streaming in the other use-case mappings.	For example, using non-IMS WebRTC data channel and/or extending WebRTC audio/video for AR media such as immersive media communications. AR signalling aspects to be studied.
Connection establishment	Find and connect is solved through SIP and E.164 addressing in IMS.	Find-and-connect for the conversational, UE-to-UE, case is undefined.	WebRTC implementations offer dedicated APIs for connection establishment in various contexts such as social media platforms. Browser applications are widely available.
Performance	Technically possible, latency is likely in principle to not be a problem to achieve, building on the existing QoS and policy framework in 5GC.	Low latency and QoS DASH support in 5GMS to be studied.	WebRTC is designed with low latency in mind but has no defined relation to QoS and policy framework in 5GC and use of that need to be studied.
Deployments	Cross-operator interconnect aspects are included. Edge processing functions to be studied.	Cross-operator interconnect aspects are currently ignored. Edge processing functions to be studied (e.g. EMSA).	Cross-operator interconnect aspects are currently not applicable since WebRTC is used OTT today, but will become relevant and need study, especially if used with QoS. Edge processing functions to be studied.
Legal Intercept (only in scope of SA3, not SA4)	LI framework exist. Possible extensions to cover new AR media formats to be studied in SA3.	Not in scope since it is not a telephony service.	Not in scope if only OTT.

NOTE: There is no support of WebRTC media stack in 3GPP today, except for the WebRTC data channel stack in MTSI. WebRTC access to IMS was studied in TR 23.701 [41] and TR 23.706, [42] and OTT WebRTC client access to 3GPP core network through a gateway is specified in TS 24.371 [43].

To describe the functional architecture for AR conversational use-cases such as clause Annex A.4 and identify the content delivery protocols and performance indicators an end-to-end architecture is addressed. The end-to-end workflow for AR conferencing (one direction) is shown in Figure 6.5.3-1. Camera is capturing the participant in an AR conferencing scenario. The camera is connected to a UE (e.g. laptop) via a data network (wired/wireless). Live camera feed, sensors and audio signals are provided to a UE/Edge node (or split) which processes, encodes, and transmits immersive media content to the 5G system for distribution. The immersive media processing function in UE may include pre-processing of the captured 3D video, format conversion, and any other processing needed before compression. Immersive media content includes 3D representation, such as in form of meshes or point clouds, of participants in an AR conferencing scenario. After processing and encoding, the compressed 3D video and audio streams are transmitted over the 5G system. A 5G STAR UE decodes, processes and renders the 3D video and audio stream.

The use-case may be extended to bi-directional/symmetric case by adding a 3D camera on the receiver side and AR glasses on the sender side and applying a similar workflow. For an asymmetrical case of EDGAR UE, the immersive media is further pre-rendered by the immersive media processing function in the 5G System and transmitted to the UE. Depending on the device capability, further media processing such as main scene management, composition, and rendering partial scene for individual participant are processed in cloud/edge.

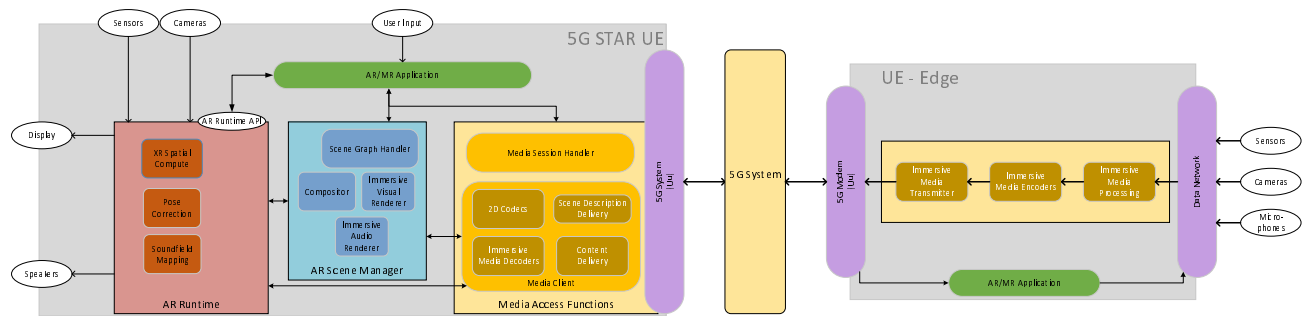


Figure 6.5.3-1: Extensions to device architecture of conversational services for STAR UE

6.5.3.1 Symmetrical case

We consider an immersive AR two party call between Alice and Bob. The end-to-end call flow is described:

1. [STAR UE Alice - STAR UE Bob]: Either one of the UEs may initiate an AR immersive call by starting an application on the phone or AR glasses.
2. [STAR UE Alice - STAR UE Bob]: Both UEs communicate with a signalling server to establish the AR call. During the session establishment, both parties agree on the format (e.g. point clouds, triangular/polygon meshes). The exact session type and configuration depends on the capabilities of STAR UE.
3. [STAR UE Alice]: Alice is captured by a depth camera embedded within the STAR UE or an external camera which generates an immersive 3D media stream (audio and video).
4. [STAR UE Alice]: The immersive 3D media is encoded and transmitted in real-time to Bob over the 5G system. Additional pre-processing may be applied before encoding such as format conversion.
5. [STAR UE Bob]: The immersive 3D media is received on Bob's STAR UE. The immersive 3D media stream is decoded and rendered on AR glasses. Additional post-processing may be applied before rendering such as format conversion, customization to match the stream to rendered environment e.g. filling holes.
6. [STAR UE Bob]: Bob is captured by a depth camera generating an immersive 3D media which is encoded and transmitted in real-time to Alice's AR glasses.
7. [STAR UE Alice]: The immersive 3D media which is received, decoded and rendered on Alice's AR glasses.
8. [STAR UE Alice - STAR UE Bob]: Both UEs terminate the service at the end of the call.

NOTE: Additional call-flows that cover other AR conferencing use-cases listed in Table 6.1-1 may be added.

6.5.3.2 Asymmetrical case

We consider an immersive AR asymmetrical call between Alice and Bob, where Bob is transmitting immersive media to be consumed on the AR glasses of Alice (STAR UE). Bob (non-STAR UE) is receiving content from Alice via other means such as audio, 2D video, etc. The end-to-end call flow is described:

1. [STAR UE Alice – non-STAR UE Bob]: Alice initiates an AR immersive call by starting an application on the phone or AR glasses.
2. [STAR UE Alice - non-STAR UE Bob]: Alice communicates with a signalling server to establish the AR call. During the session establishment, the format is identified (e.g., point clouds, triangular/polygon meshes). The exact session type and configuration depends on the capabilities of STAR UE.
3. [non-STAR UE Bob]: Bob is captured by a depth camera embedded within the STAR UE or an external camera which generates an immersive 3D media stream (audio and video).
4. [non-STAR UE Bob]: The immersive 3D media is encoded and transmitted in real-time to Alice over the 5G system. Additional pre-processing may be applied before encoding such as format conversion.

5. [STAR UE Alice]: The immersive 3D media is received on Alice's STAR UE. The immersive 3D media stream is decoded and rendered on AR glasses. Additional post-processing may be applied before rendering such as format conversion, customization to match the stream to rendered environment e.g., filling holes.
6. [STAR UE Alice]: Alice is transmitting audio, 2D video or other media content as a back channel to Bob.
7. [non-STAR UE Bob]: The 2D video or other media content which is received, decoded and rendered on Bob's device.
8. [STAR UE Alice – non-STAR UE Bob]: Alice terminates the service at the end of the call.

NOTE: Additional call-flows that cover other AR conferencing use-cases listed in Table 6.1-1 may be added.

6.5.4 Instantiation #1: MTSI-based architecture extension

This instantiation provides the detailed architecture and procedures for the case of extending the current MTSI architecture. Figure 6.5.4-1 provides an MTSI-based architecture of conversational services for STAR UE.

An MTSI client specified in TS 26.114 [15] may be extended to an AR-MTSI client which supports AR immersive media and take a role of Media Access Functions. A data channel application, an HTML web page including JavaScript(s) provided by a data channel server through a bootstrap data channel, also may be used to provide rich user experiences such as sitting side by side on a bench. Support of data channel media is optional for an MTSI client. An AR-MTSI client supporting data channel is denoted as an AR-DCMTSI client. Note that the data channel server may be implemented in IMS core or outside of it.

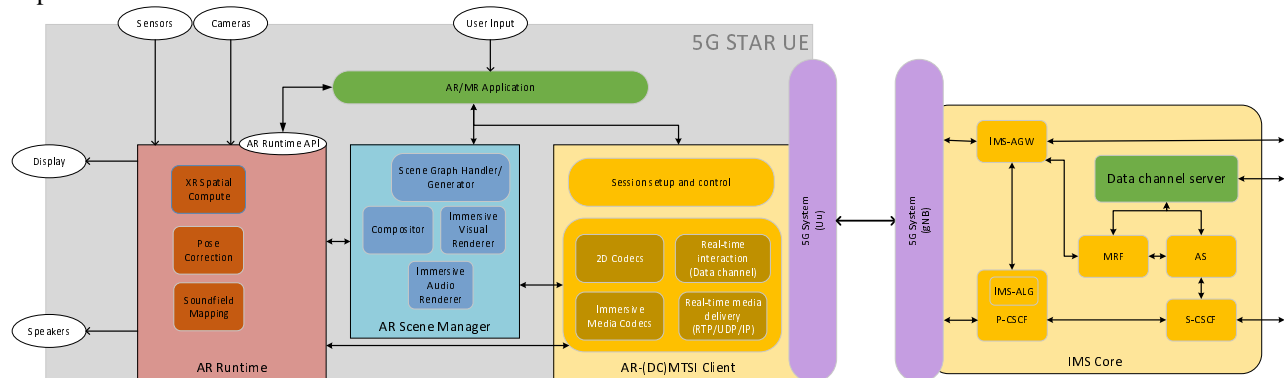


Figure 6.5.4-1: MTSI-based conversational service architecture for STAR UE

Figure 6.5.4-2 illustrates the procedure diagram for an immersive AR two party call using STAR UEs including an AR-MTSI client.

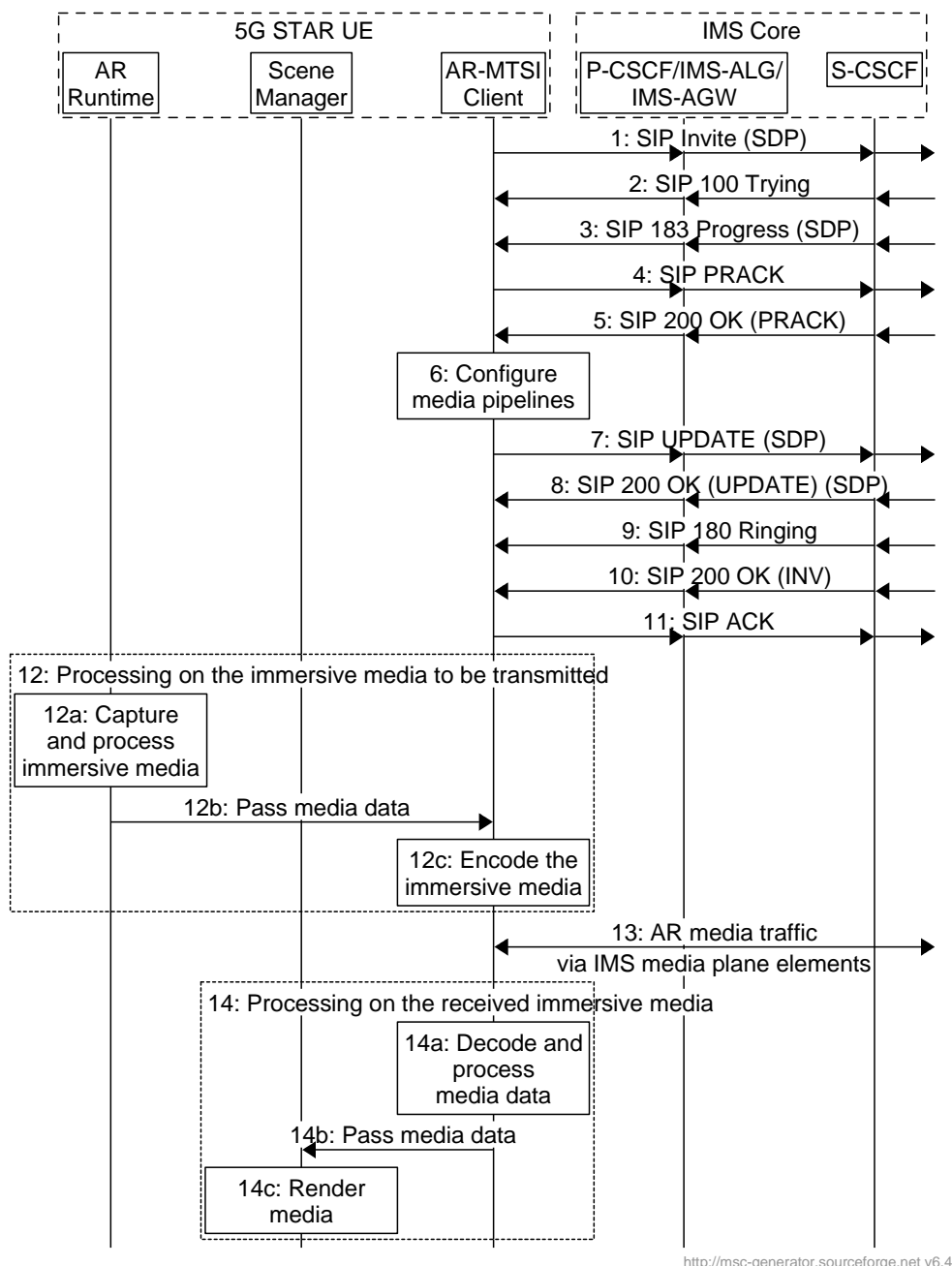


Figure 6.5.4-2: AR-MTSI client to AR-MTSI client call establishment (STAR UE)

Assumptions:

- AR immersive media is sent over RTP/UDP/IP.
- AR immersive media format (e.g. point clouds, triangular/polygon meshes) is negotiated and configured using SDP.

Procedures:

1. A STAR UE initiates a SIP INVITE request, containing the SDP offer with AR media capabilities.
2. The call propagates to the terminating STAR UE.
3. The called party's STAR UE returns an SDP answer in a SIP 183 progress message. The P-CSCF uses the SDP answer to allocate the required resources.
4. The originating STAR UE generate a PRACK which is transited to the terminating side of the call.

5. The originating STAR UE receives an associated 200 OK (PRACK).
 6. The STAR UE reserves internal resources to reflect the SDP answer and configures media pipelines.
 7. The STAR UE sends a SIP UPDATE message with a new SDP offer confirming the selected media parameters.
 8. The 200 OK (UPDATE) response is received for the terminating STAR UE containing the SDP answer.
 9. The terminating STAR UE is now alerted and sends a SIP 180 Ringing response.
 10. When the called party's STAR UE has answered the call, it sends a 200 OK to the calling party STAR UE.
 11. The STAR UE receives the 200 OK, and sends a SIP ACK message to acknowledge that the call has been established.
 12. The STAR UE processes the immersive media to be transmitted.
 - a. The AR runtime function captures and processes the immersive media to be sent.
 - b. The AR runtime function passes the immersive media data to the AR-MTSI client.
 - c. The AR-MTSI client encodes the immersive media to be sent to the called party's STAR UE.
- NOTE: The capturing may be done by an external camera. In that case, the processing and encoding may be done outside STAR UE (i.e. AR-MTSI client)
13. The STAR UE has an AR call established with AR media traffic.
 14. The STAR UE processes the received immersive media.
 - a. The AR-MTSI client decodes and process the received immersive media.
 - b. The AR-MTSI client passes the immersive media data to the Scene Manager.
 - c. The Scene Manager renders the immersive media, which includes the registration of the AR content into the real world accordingly.

Figure 6.5.4-3 illustrates the procedure diagram for an immersive AR two party call using STAR UEs including an AR-DCMTSI client.

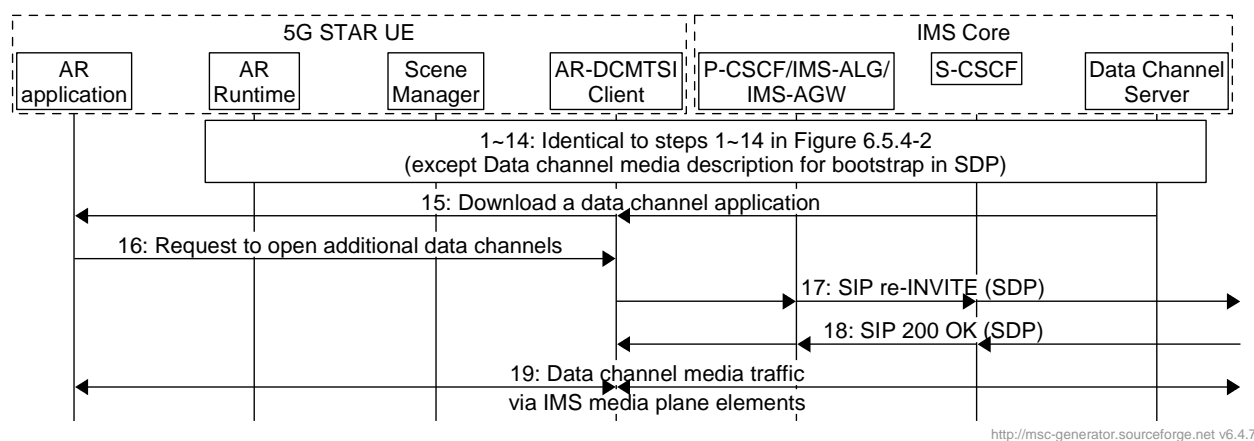


Figure 6.5.4-3: AR-DCMTSI client to AR-DCMTSI client call establishment (STAR UE)

Assumptions:

- AR immersive media is sent over RTP/UDP/IP.
- AR immersive media is negotiated and configured using SDP.
- A data channel application provides rich user experiences by utilizing both user's underlying scene and pose of objects representing users in the scene.

Procedures:

- 1-14. Same as the procedures for AR-MTSI client to AR-MTSI client call establishment except that the SDP contains a data channel media description for the bootstrap data channel.
15. The STAR UE retrieve a data channel application through the bootstrap data channel.
16. Any additional data channels created and used by the data channel application itself are requested.
17. The AR-DCMTSI client initiate SIP re-INVITE request, containing an updated SDP offer to establish those data channels.
18. The data channels for the data channel application has been established.
19. The established data channel may be used by the data channel application JavaScript(s).

6.5.5 Instantiation #2: DCMTSI-based architecture extension with immersive media processing

Compared with the instantiation for MTSI-based architecture extension, this instantiation emphasises that the IMS-AGW/MRF may support immersive media processing. It is necessary for 5G EDGAR UEs with poor media capabilities. Figure 6.5.5-1 provides an DCMTSI-based architecture of AR conversational services for EDGAR UE. A 5G EDGAR UE integrated with DCMTSI client in terminal is denoted as an EDGAR-DCMTSI client. An EDGAR-DCMTSI client may request an AR application (i.e., an entry point) via a bootstrap data channel from the data channel server. An EDGAR-DCMTSI client may also generate or retrieve some AR specific data (e.g., pose and viewpoint information) which is transmitted via additional data channels, given that non-media data is handled by using SCTP as specified in IETF RFC 8831 [44]. When an EDGAR-DCMTSI client initiates an AR call with another one, the IMS-AGW/MRF with a support of immersive media processing may perform pre-rendering with the media stream originated from the parties of this AR session if they receive the corresponding AR-specific data (i.e. the pose and viewpoint information).

EDGAR-DCMTSI clients negotiate the properties such as reliable or unreliable message transmission, in-order or out-of-order message delivery and an optional protocol for data channel using SDP as defined in IETF RFC 8864 [45]. Based on the user plane protocol stack for a basic MTSI client defined in clause 4.2 of TS 26.114 [15] and the clause 6.5 of IETF RFC 8827 [46], all data channels (e.g., both an AR application via bootstrap data channels and AR-specific data via additional data channels) are secured via DTLS.

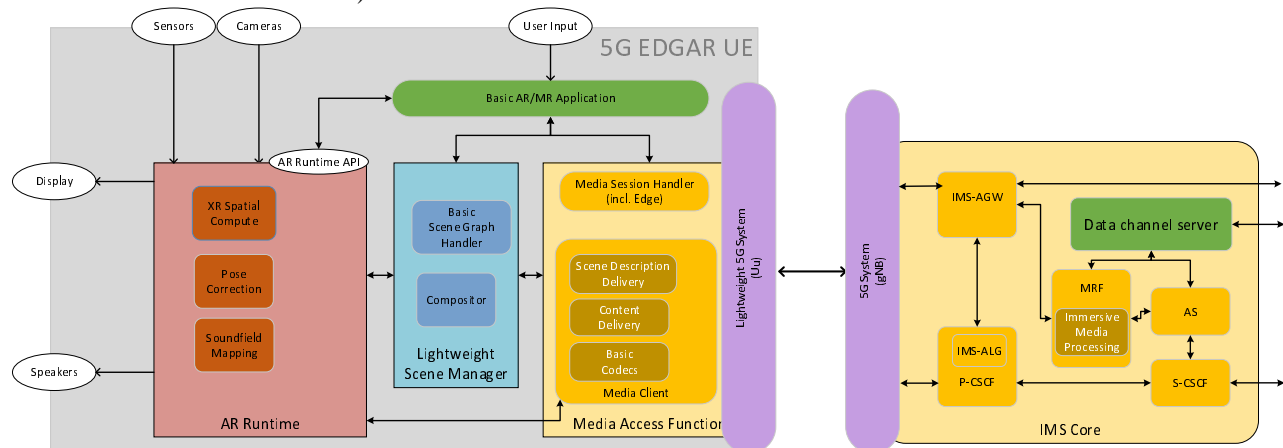


Figure 6.5.5-1: DCMTSI-based conversational service architecture for EDGAR UE

Furthermore, the IMS-AGW/MRF with a support of immersive media processing are also desirable to 5G STAR UEs due to saving power consumption. Note that the IMS-AGW/MRF with a support of immersive media processing may perform pre-rendering based on the request of the STAR UEs carried in these additional data channels. Particularly, the logical function of immersive media processing may be integrated in the MRF or other media functions.

Figure 6.5.5-2 illustrates the procedure diagram for an immersive AR conversational with two party using EDGAR UEs including an EDGAR-DCMTSI client in the context of the IMS-AGW/MRF with a support of immersive media processing. The procedure is also applicable to establish an immersive AR call where the two parties of a session are STAR UEs or one is STAR UE and the other is EDGAR UE.

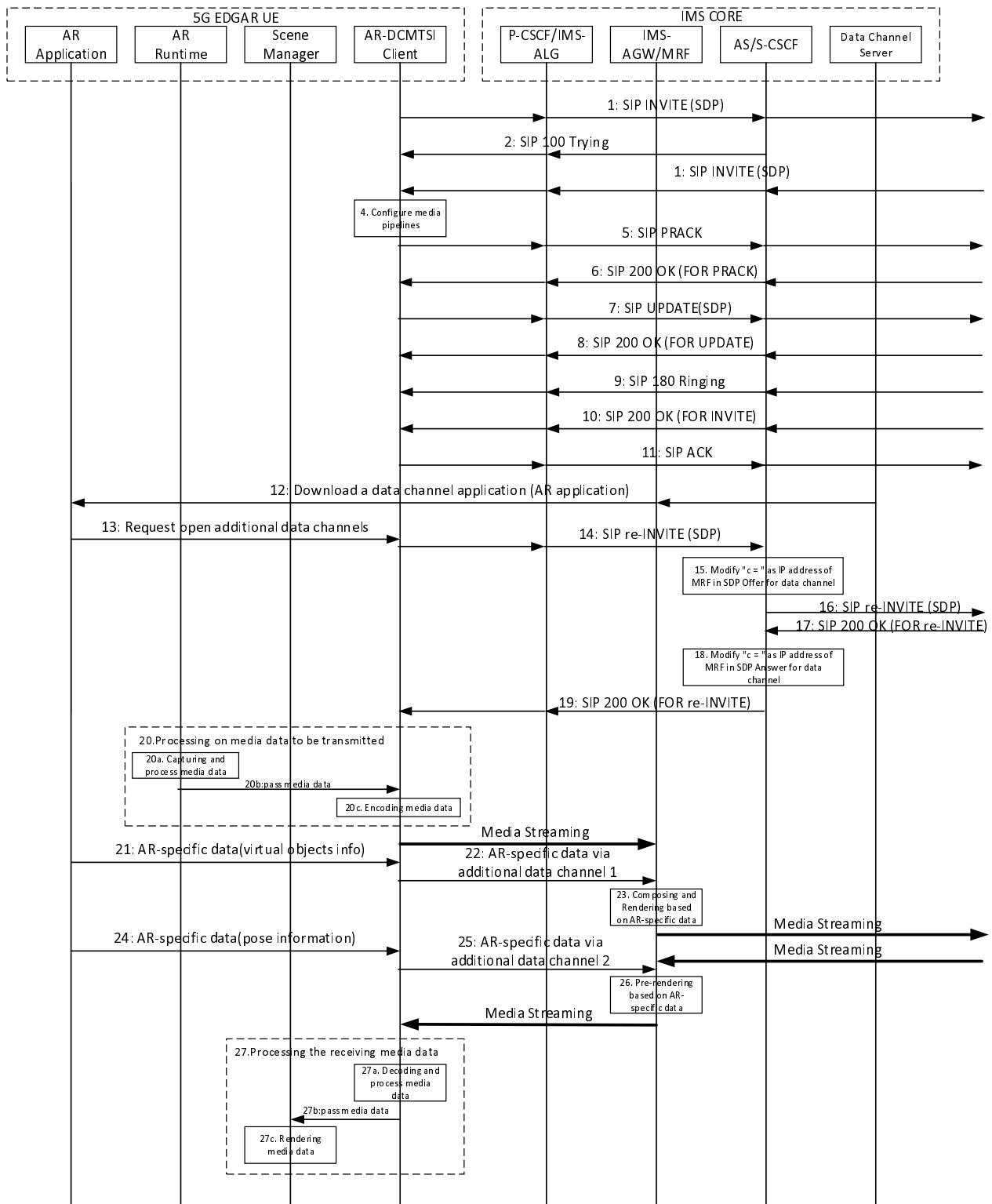


Figure 6.5.5-2: AR-DCMTSI client to AR-DCMTSI client call establishment for EDGAR UE

Assumptions:

- AR immersive media may be sent over RTP/UDP/IP and/or SCTP/UDP/IP.
- AR immersive media may be negotiated and configured using SDP.
- A data channel application may provide rich user experiences by utilizing both user's underlying scene and pose of objects representing users in the scene.

Procedures:

1. An EDGAR UE initiates a SIP INVITE request, containing the SDP offer with AR media capabilities.
2. The call propagates to the terminating EDGAR UE.
3. The terminating EDGAR UE returns an SDP answer in a SIP 183 progress message. The P-CSCF uses the SDP answer to allocate the required resources.
4. The originating EDGAR UE generate a PRACK which is transited to the terminating side of the call.
5. The originating EDGAR UE receives an associated 200 OK (PRACK).
6. The terminating EDGAR UE reserves internal resources to reflect the SDP answer and configures media pipelines.
7. The originating EDGAR UE sends a SIP UPDATE message with a new SDP offer confirming the selected media parameters.
8. The 200 OK (UPDATE) response is received for the terminating STAR UE containing the SDP answer.
9. The terminating EDGAR UE is now alerted and sends a SIP 180 Ringing response.
10. When the terminating EDGAR UE has answered the call, it sends a 200 OK to the originating EDGAR UE.
11. The terminating EDGAR UE receives the 200 OK, and sends a SIP ACK message to acknowledge that the call has been established.
12. The EDGAR UEs retrieve a data channel application for AR through the bootstrap data channel. If the EDGAR UE enables to provide native AR application, this step is not required.
13. Any additional data channels created and used by the data channel application for AR itself are requested.
14. The originating EDGAR UE initiates SIP re-INVITE request, containing an updated SDP offer to establish those additional data channels.
15. The AS/S-CSCF identify an updated SDP offer for additional data channels and modify the "c=" as the IP address of the MRF, and then send this SDP offer to the remote party.
16. The AS/S-CSCF send this updated SDP offer to the remote party.
17. The AS/S-CSCF receive an updated SDP answer from the remote party.
18. The AS/S-CSCF identify this updated SDP answer for additional data channels and modify the "c=" as the IP address of the MRF, and then send this SDP answer to the remote party.
19. The additional data channels for the data channel application has been established.
20. The EDGAR UE processes the immersive media to be transmitted.
 - a. The AR runtime function captures and processes the immersive media to be sent.
 - b. The AR runtime function passes the immersive media data to the AR-DCMTSI client.
 - c. The AR-DCMTSI client encodes the immersive media to be transmitted to the IMS-AGW/MRF supporting immersive media processing.

NOTE: The capturing process may be done by an external camera. In this case, the processing and encoding processes are done outside EDGAR UE (e.g., AR-DCMTSI client)

21. The data channel application for AR collects the AR-specific data, and decides to send them to the AR-DCMTSI client if the AR experiences requires assistance from the network side.
22. The AR-DCMTSI client sends the AR-specific data (e.g., virtual objects info) to the IMS-AGW/MRF via the designated data channel 1 based on the previous SDP negotiation.
23. The IMS-AGW/MRF composes, renders and encodes the AR immersive media based on the received media stream and the AR-specific data from the originating party, and finally send them to the terminating party.

24. The data channel application for AR collects the AR-specific data, and decides to send them to the AR-DCMTSI client if the AR experiences requires assistance from the network side.
25. The AR-DCMTSI client sends the AR-specific data (e.g. pose info and/or viewport info) to the IMS-AGW/MRF via the designated data channel 2 based on the previous SDP negotiation.
26. The IMS-AGW/MRF decodes and pre-renders media stream based on the received media stream from the terminating party and the AR-specific data from the originating party, and finally sends them to the originating party.
27. The EDGAR UE processes the received immersive media.
 - a. The AR-DCMTSI client decodes and process the received immersive media.
 - b. The AR-DCMTSI client passes the immersive media data to the Lightweight Scene Manager.

The Lightweight Scene Manager renders the immersive media, which includes the registration of the AR content into the real world accordingly.

6.5.6 Content formats and codecs

Based on the use cases, the following formats, codecs, and packaging formats are of relevance for AR conversational:

- General
 - > 2D Video Formats and video compression codecs
 - > Regular Audio Formats and audio compression codecs
- In addition, for downlink
 - > Immersive media 3D Formats such as static and dynamic point clouds or meshes
 - > Spatial Audio Formats
 - > Decoding tools for such formats
 - > Composed Scene Graph and Scene Description
- In addition, for uplink
 - > Immersive media 2D Video Formats with depth
 - > Immersive media 3D Formats such as static and dynamic point clouds or meshes
 - > Spatial Audio Formats
 - > Encoding tools for such formats
 - > Streaming of sensor information (e.g., gyroscope, accelerometer) as well as pose information

NOTE 1: Details on uplink delivery of immersive media 3D formats are for further study, to take into account the specific latency requirements of each conversational use case.

NOTE 2: It is not necessary to support all media formats listed, depending on the device type and/or application.

6.5.7 Summary of AR conversational instantiations

Table 6.5.7-1 shows the list of potential instantiations and how they may be composed from each building block described in clause 6.5.1.

Table 6.5.7-1: Summary of each instantiation for AR conversational services

Building Block	Instantiation#1: MTSI extension	Instantiation#2: DCMTSI extension
Call setup and control	Conventional MTSI	Conventional MTSI with Data Channel
Media Formats	as specified in clause 6.5.6	as specified in clause 6.5.6
Delivery	RTP/UDP/IP, SCTP/DTLS/UDP/IP	RTP/UDP/IP, SCTP/DTLS/UDP/IP
5G system integration	Need policy exchange for AR-(DC)MTSI client (P-CSCF and PCF)	Need policy exchange for EDGAR-DCMTSI client (P-CSCF and PCF)

6.5.8 Standardization areas

The list of potential standardization area that has been collected is provided in the following:

- Immersive media format and profile with integration into relevant 5G architecture
- Scene description format, functionality, and profile as an entry point of immersive media
- Scene description update mechanism
- Relevant subset of media codecs for different media types and formats
- CMAF encapsulation of immersive media for 5G media streaming
- Media payload format to be mapped into RTP streams
- Capability exchange mechanism and relevant signalling (e.g., SDP)
- Protocol stack and content delivery protocol for various architecture options as identified in Table 6.5.3-1
- Functionalities to support split rendering and network-based media processing allocation with 5G edge/MRF
- Required QoS and QoE for AR/MR conversational service

6.6 Shared AR conversational experience

6.6.1 Introduction

Shared AR Conversational experience is an end-to-end conversational service that includes communication between two or more parties through a network/cloud entity that creates a shared experience, meaning that every party in the call in its AR experience sees the same relative arrangements of the other participants relative to each other. Therefore, for instance the interaction between two parties seating next to each other in the virtual space (e.g. when these parties turn to each other when talking) is seen by all participants in the same way. Note that the AR runtime in each device customizes and updates the arrangement of the people in the virtual room. The absolute positioning of people or objects in a user's scene may vary based on the physical constraints of the user's room. This shared experience distinguishes this use case from the AR conversational experience of clause 6.5.

In addition to the building blocks listed in clause 6.5.1, an immersive media processing function is needed to create the shared virtual experience. This requirement is discussed as an abstract functionality. In various deployments, this functionally may be implemented in different ways or by different entities, in a centralized or distributed fashion, and other possible arrangements.

This experience may be deployed with a combination of AR and non-AR devices. In this context, an AR device is capable of laying over received media object on a see-through glass (e.g. AR glasses) or the display of the device while capturing live content through its camera and rendering on its display (e.g. a table or phone). A non-AR device only receives one or multiple 2D video streams each representing one of the other participants but is incapable of laying over received media object with the see-through or captured by its camera scene. In such a scenario, each AR device creates an AR scene as mentioned above. But an application running on the edge/cloud may create one or multiple 2D videos

(i.e. a VR video or multi-view videos) of a virtual room which includes all other participants and streams one or more of them to a non-AR device, based on its user's preference. The user on non-AR devices can also change its viewport to the virtual room by changing the position of its device or using navigation devices such as keyboard or mouse, but the device does not provide an AR experience.

6.6.2 Relevant use cases

The use cases relevant to this scenario may be further categorized including:

- UC#8: 360-degree conference meeting
- UC#9: XR meeting
- UC#10: Convention / Poster Session
- UC#12: AR avatar multi-party calls
- UC#13: Front-facing camera video multi-party calls
- UC#19: AR conferencing
- UC#22: shared AR conferencing experience

6.6.3 Basic architecture

To describe the functional architecture for shared AR conversational experience use case such as clause Annex A.7 and identify the content delivery protocols and performance indicators, an end-to-end architecture is addressed. The end-to-end architecture for shared AR conferencing (one direction) is shown in Figure 6.6.3-1. To simplify the architecture, only 5G STAR UE is considered in this figure.

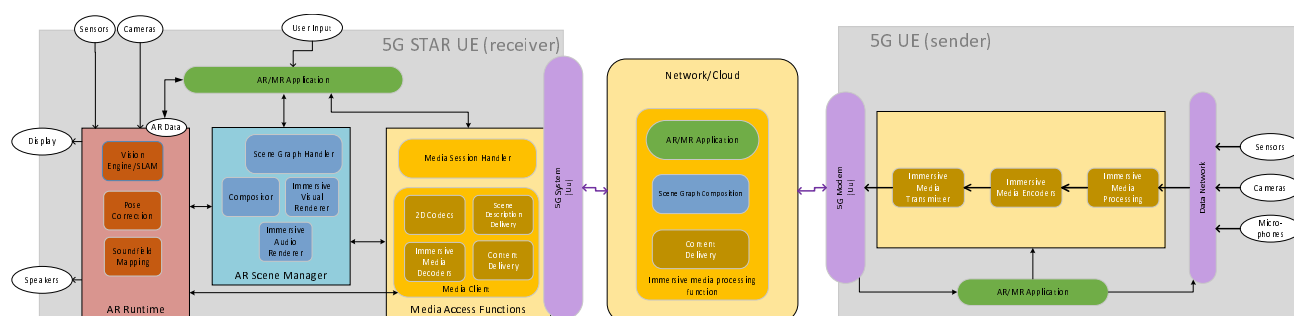


Figure 6.6.3-1: Shared AR conversational service for STAR UE.

Camera(s) are capturing the participant(s) in an AR conferencing scenario. The camera(s) for each participant are connected to a UE (e.g. laptop or mobile phone or AR glasses) via a data network (wired/wireless). Live camera feeds, sensors, and audio signals are provided to a UE which processes, encodes, and transmits immersive media content to the 5G system for distribution. In multi-party AR conversational services, the immersive media processing function on the cloud/network receives the uplink streams from various devices and composes a scene description defining the arrangement of individual participants in a single virtual conference room. The scene description as well as the encoded media streams are delivered to each receiving participant. A receiving participant's 5G STAR UE receives, decodes, and processes the 3D video and audio streams, and renders them using the received scene description and the information received from its AR Runtime, creating an AR scene of the virtual conference room with all other participants.

Also note that if the format conversion is desired, the immersive media processing function on the cloud may optionally use media services such as pre-processing of the captured 3D video, format conversion, and any other processing before compression of immersive media content including 3D representation, such as in form of meshes or point clouds, of participants in an AR conferencing scenario.

NOTE: As an example of the composite scene generation, the immersive media processing function may take the input from the participants' physical constraints, so that the generated scene is consistent with every participants' environment and can be rendered at each device consistently.

Figure 6.6.3-2 illustrates the architecture for shared AR conversational experience use case when an 5G EDGAR UE (receiver) is used. While the functionalities of the sender and the network/cloud shown in Figure 6.6.3-1 are identical in the STAR and EDGAR devices, an EDGAR device uses a split-rendering function on Cloud/Edge.

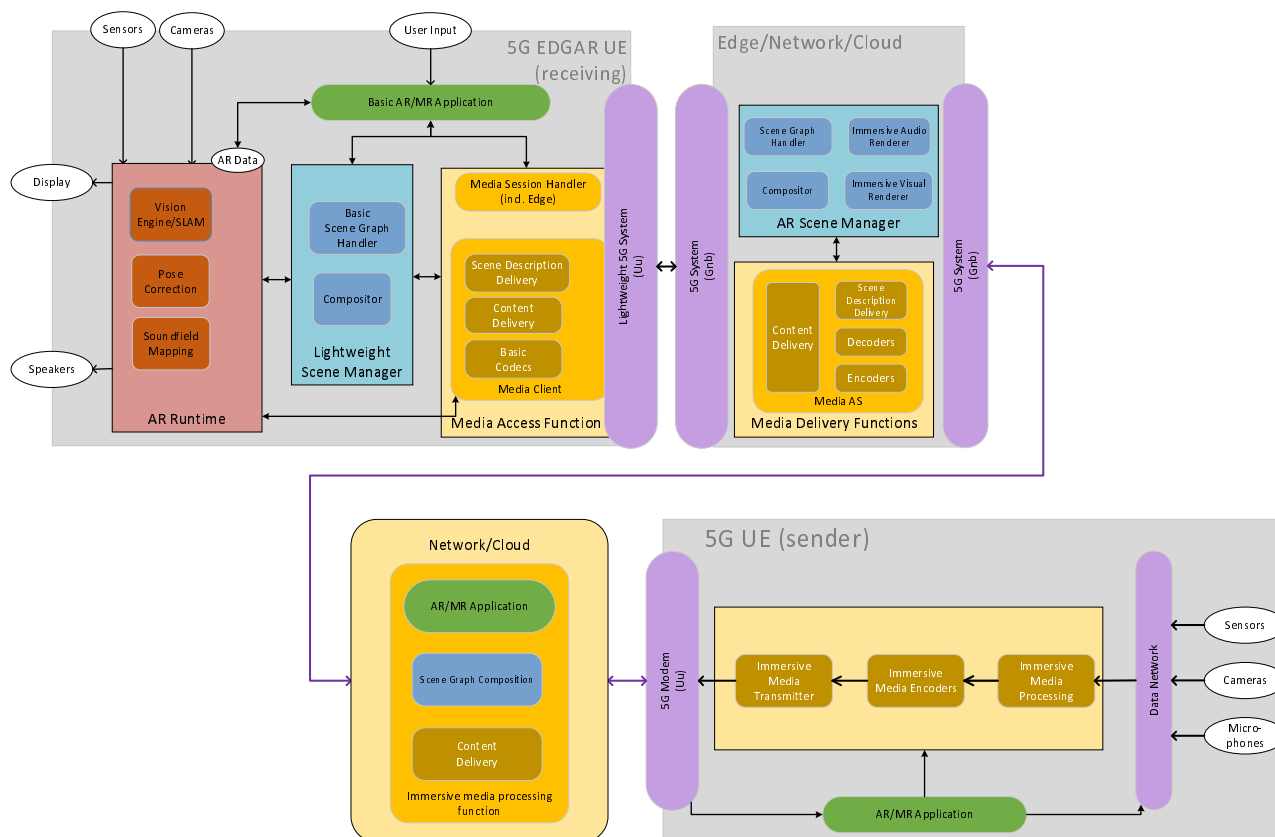
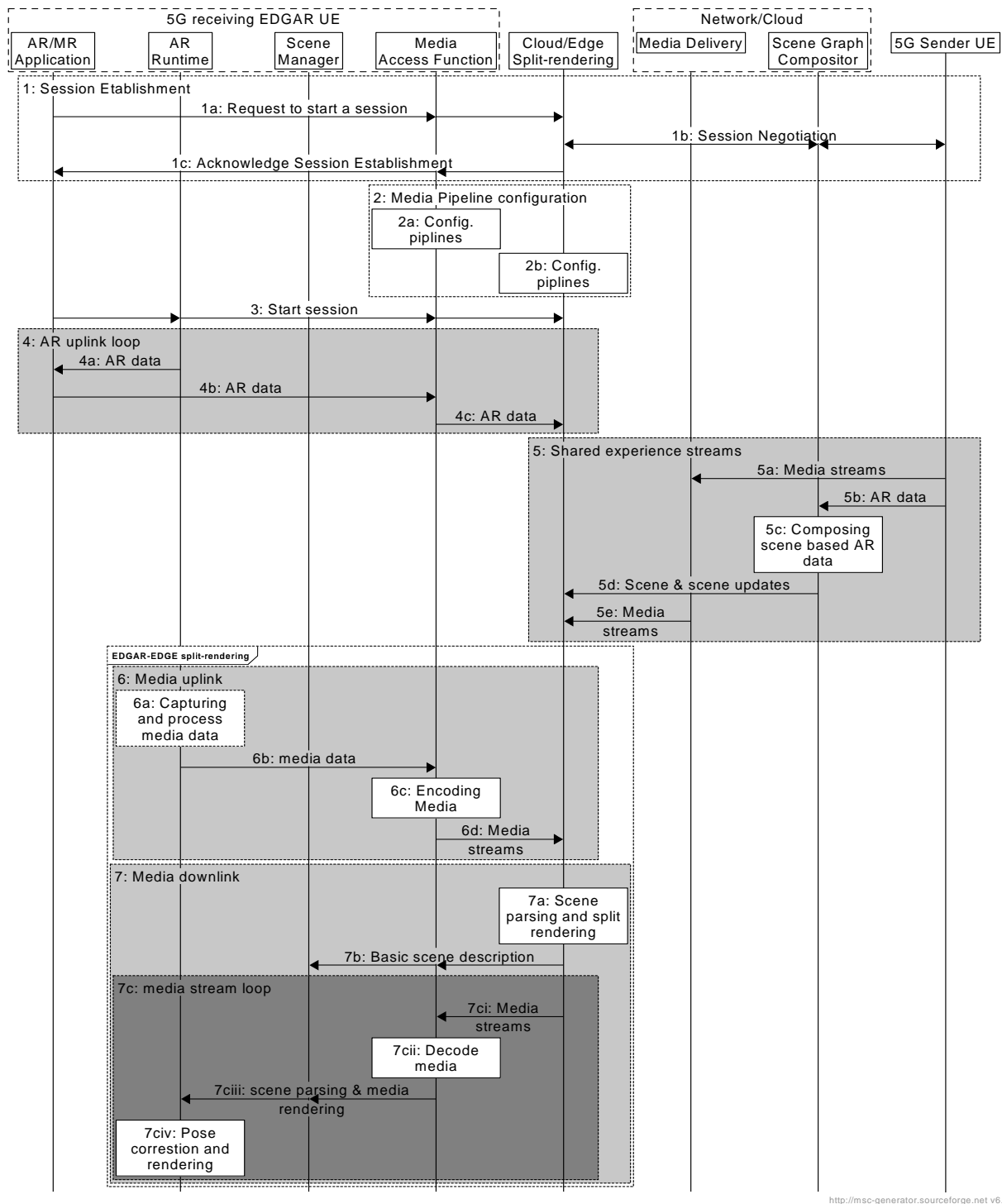


Figure 6.6.3-2: Shared AR conversational service for EDGAR UE and cloud/edge pre-rendering.

The AR session management may be done by AR/MR application on device. In this case, it is a responsibility of the device to connect and to acquire an entry point for edge/cloud during the session management. AR Scene Manager on cloud/edge generates the lightweight scene description and simple format of AR media that match AR glass display capabilities of the individual participant's 5G EDGAR device. The lightweight scene description and encoded rendered scene are delivered to the UE. The UE receives the simple format of AR media and audio streams, decodes and renders them using the received lightweight scene description and the information received from its AR Runtime, creating an AR scene of the virtual conference room with all other participants.

6.6.4 Generic Call flow

Figure 6.6.4-1 illustrates the call flow for an immersive AR conversational for a receiving EDGAR UE. Only one sender is shown in this diagram without showing its detailed call flow.



<http://msc-generator.sourceforge.net v6.4.7>

Figure 6.6.4-1: Shared AR conversational experience call flow for a receiving EDGAR UE

Procedures:

1. Session Establishment:
 - a. The AR/MR Application requests to start a session through EDGE.
 - b. The EDGE negotiates with the Scene Composite Generator (SCG) and the sender UE to establish the session.
 - c. The EDGE acknowledges the session establishment to the UE.
2. Media pipeline configuration:
 - a. MAF configures its pipelines.

- b. EDGE configures its pipelines.
- 3. The AR/MR Application requests the start of the session.

Loops 4, 5, 6, and 7 are run in parallel:

- 4. AR uplink loop:
 - a. The AR Runtime sends the AR data to the AR/MR Application.
 - b. The AR/MR Application processes the data and sends it to the MAF.
 - c. The MAF streams up the AR data to the EDGE.
- 5. Shared experience loop:
 - a. Parallel to 9, the sender UE streams its media streams up to Media Delivery (MD).
 - b. The sender UE streams its AR data up to the Scene Graph Compositor (SGC).
 - c. Using the AR data from various participants, the SCG creates the composted scene.
 - d. The composted scene is delivered to the EDGE.
 - e. The media streams are delivered to the EDGE.
- 6. Media uplink loop:
 - a. The AR Runtime captures the media components and processes them.
 - b. The AR Runtime sends the media data to the MAF.
 - c. The MAF encodes the media.
 - d. The MAF streams up the media streams to the EDGE.
- 7. Media downlink loop:
 - a. The EDGE parses the scene description and media components, partially renders the scene, and creates a simple scene description as well as the media component.
 - b. The simplified scene is delivered to the Media Client and Scene Manager.
 - c. Media stream loop:
 - i. The pre-rendered media components are streamed to the MAF.
 - ii. The MAF decodes the media streams.
 - iii. The Scene Manager parses the basic scene description and composes the scene.
 - iv. The AR manager after correcting the pose, renders the immersive scene including the registration of AR content into the real world.

6.6.5 Various instantiations

Similar to clause 6.5.3, the shared AR conversational experience may be instantiated in various 5G systems:

- a) The MTSI architecture (TS 26.114 [15]) supports audio and 2D video conversational services.
- b) Extending the 5GMS architecture (TS 26.501 [26]) to support AR conversational services by combining live uplink and live downlink.
- c) An architecture based on something different than MTSI / IMS or 5GMS, for example WebRTC.

For the comparison between different instantiations, please refer to Table 6.5.3-1.

6.6.6 Content formats and codecs

Based on the use cases, the following formats, codecs and packaging formats are of relevance for Media Streaming of AR:

- Scene Graph/Description
- 2D Video Formats
- 3D Formats such as static and dynamic point clouds or meshes
- Animated 3D meshes
- 2D Video Formats with depth
- 2D, stereo, and spatial audio formats
- Several video decoding instances
- Decoding tools for such formats
- Encoding tools for 2D formats
- Low-latency downlink and uplink real-time streaming of the above media
- Uplink streaming of pose information
- Uplink streaming of media

6.6.7 Standardization areas

The list of potential standardization area that has been collected is provided in the following:

- Immersive media format and simplified media formats with integration into relevant 5G architecture
- Scene description format, functionality, and profile as an entry point of immersive media
- Scene description update mechanism
- Relevant subset of media codecs for different media types and formats
- CMAF encapsulation of immersive media for 5G media streaming
- Media payload format to be mapped into RTP streams
- Capability exchange mechanism and relevant signalling
- Low-latency uplink streaming of captured AR data
- Functionalities and session management to support split rendering and network-based media processing allocation with 5G edge/MRF
- Required QoS and QoE for shared AR/MR conversational experience service

7 Considerations on Devices Form-factor

7.1 General

The components of AR glasses are same or similar with those of mobile phones which may launch and execute AR/MR applications. However, AR glasses have rather different requirements and limitations compared with mobile phones.

From a form factor perspective, AR glasses have several different design considerations. For example, AR glasses have two separate see-through displays for each eye. They also usually include more than two vision cameras which are

spatially separated in order to achieve better disparity for depth estimation. In addition, AR glasses are worn and closely attached to a user's face and contain IMU sensors to estimate where the user's focal point is. Most of the included components are designed and placed in order to meet requirements which differ to those for mobile phones.

From a media processing perspective, AR/MR applications consume far more energy than non-AR/MR applications [27]. Multiple, as well as different types of cameras are always turned on to track the features detected in 2D and 3D video every second. In the case when AR/MR objects are augmented into the real world, the objects need to be rendered frame by frame with different view frustum positions and directions. In the case when the AR/MR objects are rendered in a server, the AR/MR device is expected to upload the user's pose in a millisecond frequency, then download, decode, correct, and composite the pre-rendered image sequences streamed from the server.

Besides, from an ergonomics perspective, restrictions need to be considered to place the components of the AR glasses in a limited space and under the manageable range of user neck joint torque.

This clause addresses form-factor related issues from the components of AR glasses device architectures, such as battery/power consumption, camera, display, heat dissipation, and weight.

7.2 Battery/Power consumption

The run time of a typical battery is proportional to its physical size, capacity, and weight, while they are proportional to user discomfort and neck torque. A study on the characteristics of AR applications [27] measured battery consumption of commercially available applications on AR, streaming, and social networking which shows that AR applications consume around at least 46% more energy than non-AR applications. The capacity of the battery needs to be designed to support a fair amount of running time for the everyday use of AR/MR applications. The amount of running time could be from tens of minutes for shopping of products via AR remote advertising in Annex A.2, 1-2 hours for streaming of volumetric video in Annex A.3, or even several hours for AR gaming in Annex A.6. However, as capacity is typically proportional to weight, and as the AR glasses is expected to be worn and equipped under the consideration of human ergonomics such as neck strain, there are clear limitations on extending the capacity of the battery. Such limitations may be relaxed by dynamically offloading some energy-intensive workloads to 5G cloud/edge. In this case, local processing power consumption is exchanged with power consumption for 3GPP/non-3GPP connectivity and an always on connectivity as well. For connectivity Discontinuous Reception (DRX) and Reduced Capability (RedCap) may be one of examples looking for lower power consumption for the radio for AR/MR application.

The following KPI is related with battery and power consumption and listed in clause 4.5.2.

- Maximum Available Power

7.3 Camera

Augmented reality may be realized by SLAM. To understand the physical world through SLAM, various types of multiple cameras need to be continuously turned on and always need to be acquiring image sequences.

Among the various components contributing to heat, such as CPU, GPU, camera and display, it is measured that the cameras are one of major sources of heat dissipation for AR applications [27]. AR/MR applications may need to be aware of the available run time remaining, and the amount of heat dissipation felt by the user.

In addition, as multiple cameras may be equipped in AR glasses for various purposes, they need to be designed and placed optimally to process the required functions in AR Runtime. Camera related parameters, such as for calibration, pose correction, Vision Engine, SLAM etc. are expected to have a big impact on the quality of service for AR glasses. AR/MR applications may need to be aware of intrinsic and extrinsic parameters for the cameras to properly process the required functions. Such parameters may be delivered to the server whenever there is any change in camera configurations.

The following KPI is related with camera and listed in clause 4.5.2

- Maximum Available Power

7.4 Display

There is at least one display for each eye on a pair of immersive AR glasses. The AR glasses estimates the position of each eye then presents pixels of the rendered AR/MR objects on the display in order to combine the ray of light

reflected from the surface of real-world objects with each pixel. A renderer in the AR scene manager may take into consideration the shape and optical distortion characteristics of the displays, pixel arrangements, and the estimated position of each eye of the user. At least one of the view frustum models that represents either an AR glasses, each display, or each eye, with a 3D map of the surroundings may be provided to the AR scene manager in order to minimize the post processing of customizing a generic rendered image to fit to a certain pair of AR glasses.

The following KPI(s) are related with display and listed in clause 4.5.2.

- Maximum Available Power
- Persistence – Duty time
- Display refresh rate
- Spatial Resolution per eye
- Content frame rates
- Brightness
- Field of View
- Eye Relief
- Calibration
- Depth perception

7.5 Heat dissipation

It has been studied that AR applications may generate 4-5 degrees (in Celsius) higher heat than non-AR applications on the same device [27]. Another study shows that a user's heat sensation and discomfort increase with temperature. Overheated components have not only degraded performance but also power leakage through thermal throttling [28].

The following KPI may be related with heat dissipation and listed in clause 4.5.2.

- Maximum Available Power

7.6 Weight

AR glasses consists of displays, sensors, cameras, batteries and so on. The weight of AR glasses puts constant pressure on a user's skin and changes the amount of torque applied to the neck joints and muscles in a neutral posture.

A study shows that a user's posture may be changed from a neutral to a look-up posture, a look-down posture, or a body-bending posture because of the relative placement of virtual objects [29]. Those different postures increase the moment arm between the Centre of Mass (CoM) of the wearable device and the neck joint.

There are different characteristics between HMD type and glasses type devices, as the CoM of glasses type devices is biased towards the front of the device, by design. As a result, AR/MR applications need to consider the issues due to the differences in the ergonomics between the two different types of wearable devices.

The following KPI is related with weight and listed in clause 4.5.2.

- Maximum Weight

8 Potential New Work and Study Area

8.1 General

This clause documents and clusters potential new work and study areas identified in the context of this Technical Report. In particular, two areas have been identified as crucial for supporting AR type of services and applications that impact network and terminal architectures:

- 5G Generic Architecture for Real-Time Media Delivery as introduced in clause 8.2.
- Support for Media Capabilities for Augmented Reality Glasses as introduced in clause 8.5.

In order to separate the work areas of these potential work topics, Figure 8.1-1 and Figure 8.1-2 provides the high-level scope of these two work topics for STAR-based and EDGAR-based UEs, respectively.

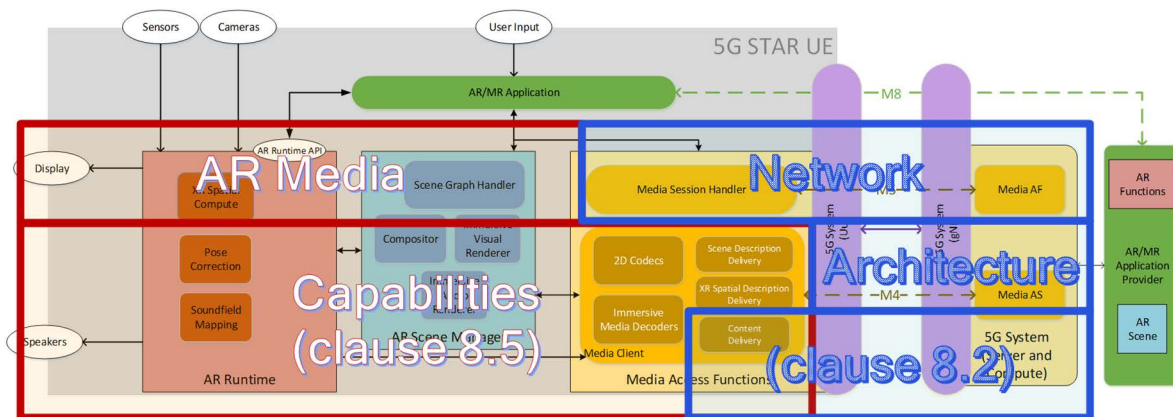


Figure 8.1-1: Work topic separation between AR media capabilities, terminal architecture and network architecture for STAR-type devices.

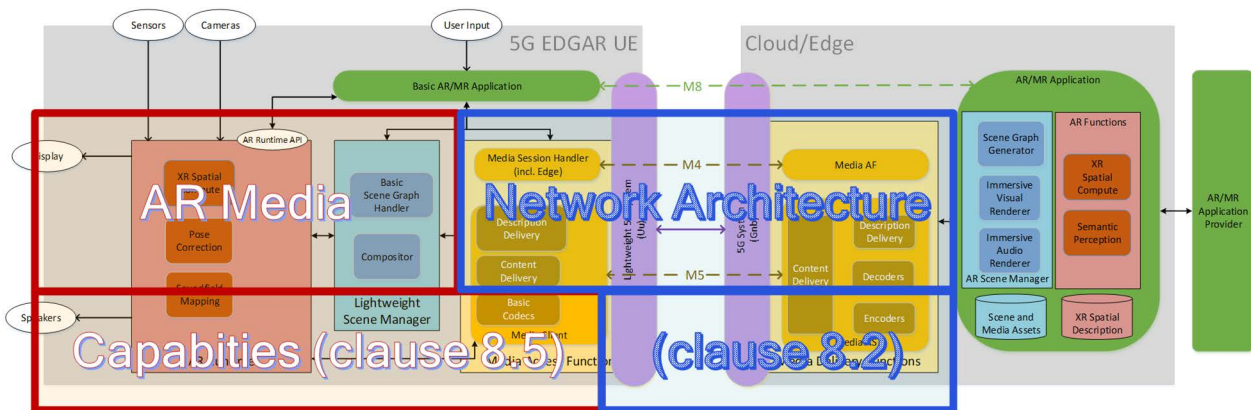


Figure 8.1-2: Work topic separation between AR media capabilities, terminal architecture and network architecture for EDGAR-type devices.

8.2 5G Generic Architectures for Real-Time Media Delivery

Based on the initial conclusions in TR 26.928 [2], clause 7, and the evaluation of architectures in clause 4 and 6 of this report, it is clear that for the integration of AR services and experiences into 5G Networks, the approach taken in 5GMS to separate the data plane and the control plane, and enable access of third-party services getting access to 5G System functionalities, is a major benefit. The basic concept is the extension of 5GMS principles to any type of service including real-time communication and split-rendering. While the work is motivated by XR and AR experiences discussed in this TR, it is neither specific nor limited to those experiences. In principle, the basic control plane is similar/identical to 5GMS, and the media plane is generic, permitting different types of operator and third-party services supported by the 5G System. The following aspects are identified:

- 5GMS-like network architectures to support any type of media services including real-time communication, split rendering and spatial computing
- Operator and third-party services need to be supported
- Separation of user and control plane functionalities

Based on the above, it is considered to specify 5G generic architectures for real-time media delivery addressing the following stage-2 work objectives:

- A generic media delivery architecture to define relevant core building blocks, reference point, and interfaces to support modern operator and third-party media services based on the 5GMS architecture
- Provide all relevant reference points and interfaces to support different collaboration models between 5G System operator and third-party media service provider, including but not limited to an AR media service provider.
- Call flows and procedures for different service types, for example real-time communication, shared communication, etc., based on the context of clause 6
- Specify support for AR relevant functionalities such split-rendering or spatial computing on top of a 5G System based on this architecture

8.3 5G-Media Service Enablers

AR applications rely on functionalities provided by devices and networks. On devices, such functionalities are typically bundled in software development kits (SDKs) in order to get access to complex hardware functionalities. SDKs typically expose APIs to simplify the communication with the underlying hardware and network functionalities.

What is clearly needed for AR and provided for example by Khronos with OpenXR, are standardized APIs to access underlying AR hardware functions. However, the standardized APIs and functions in OpenXR are restricted to local device processing. In order to enable and simplify the access to 5G network, system and media functionalities for AR, it is beneficial to provide packages and bundles for application providers. Typical assets for media service enablers are:

- Set of functions that may be used to develop applications on top of 5G Systems
- Set of robust features and functionalities which reduce the complexity of developing applications
- Functions to leverage system and radio optimizations as well as features defined in 5G System (5G Core Network and 5G NR)
- Provision and documentation of APIs to enable or at least simplify access to these functionalities
- Provision of network interfaces to connect to the 5G System
- A testable set of functions. Testing and conformance may be addressed outside 3GPP by an appropriate Marketing and Public Relations (MPR) or Industry Forum.
- Guidelines and examples to make use of the functionalities

It is proposed to use the concept of 5G-media service enablers to define relevant specifications for AR and possibly other applications. A common set of properties and functionalities for Media Service Enabler specifications is needed and hence it is proposed to provide a 3GPP internal report that:

- Define the principal properties of media service enablers
- Define minimum and typical functionalities of media service enablers
- Define a specification template for media service enablers
- Identify possibly relevant stage-2 and stage-3 work for media service enablers
- Collect a set of initially relevant media service enablers for normative work

8.4 5G Real-time Communication

As documented in clause 4.2.6 and further developed in the context of clause 6, there are several use cases that require a 5G Real-time communication. The use cases include:

- 1) EDGAR-based UEs relying on rendering on the network. In this case, the downlink requires sending pre-rendered viewports with lowest latency, typically in the range below 50ms.
- 2) Uplink streaming of camera and sensor information for cognitive/spatial computing experiences, in case the environment tracking data and sensor data is used in creating and rendering the scene.
- 3) Conversational AR services require real-time communication both in the downlink and the uplink, even independent from MTSI for app integration of the communication.

In order to provide adequate QoS as well as possible optimizations when using a 5G System for media delivery, an integration of real-time communication into the 5G System framework is essential.

As identified in clause 4.2.6 and clause 6.5, there is a need for supporting third-party applications in 5G real-time communication as well as server-based real-time streaming. From an app developer perspective, an enabler is preferable, especially to support real-time streaming, for example split-rendering.

Different options may be considered, for example re-use of parts of MTSI such as the IMS data channel and 5G Media Streaming for managed services, or re-use of WebRTC for OTT services. A 5G real-time communication is expected to be aligned with either IMS or WebRTC but provides additional functions to integrate with the 5G System.

It is proposed to define a general 5G real-time communication media service enabler that includes, among others, the following functionalities:

- A protocol stack and content delivery protocol for real-time communication based on RTP
- A common session and connection establishment framework, with instantiations based on SIP and SDP for IMS or SDP and ICE for WebRTC, including further possible investigation of control plane
- A capability exchange mechanism
- A security framework, for example based on SRTP and DTLS for WebRTC
- Uplink and downlink communication
- Suitable control protocols for end-to-end adaptation
- QoS and 5G System integration framework
- Reporting and QoE framework

8.5 Media Capabilities for Augmented Reality Glasses

In TR 26.928 [2] and this report, XR and AR device architectures have been developed and details on relevant media formats are documented, for example in, clause 4.4. In particular, it is identified that for design AR glasses, implementation and operational requirements are significantly more stringent than for smart phones (see clause 4.5.2 and clause 7). As an example, consuming media on AR glasses requires functionalities to address very low power consumption, low area size, low latency options, new formats, operation of multiple decoders in parallel, etc.

To support basic interoperability for AR applications in context of 5G System based delivery, a set of well-defined media capabilities are essential. These capabilities may be used in different services and applications and hence service-independent capabilities are relevant. The media capabilities typically address three main scenarios:

- Support of basic media services on such glasses with simple rendering functionalities
- Support of split-rendering, e.g. a pre-rendering of eye buffers is carried out in the cloud/edge
- Support of sensor and device data streaming to the network in order to support network-based processing or device sensor information

Media functions are relevant for the Media Access Function as defined in clause 4.2.6. The media capabilities are importantly driven by realistic deployment options addressing device capabilities, as documented in clause 4.5.2, as well as the relevant KPIs.

In particular, the following objectives need to be considered:

- Define a reference terminal architecture for AR devices
- Define at least one AR device category that addresses the constraints of an EDGAR-type AR glasses

NOTE: Additional device categories may be defined, but with lower priority

- For each AR device category
 - > Define media types and formats, including scene description, audio, 3D/2D graphics and video, as well as sensor information and metadata of user and environment.
 - > Define decoding capabilities, including support for multiple parallel decoders
 - > Define encoding capabilities
 - > Define security aspects related to media capabilities
- Support signalling (e.g., SDP and MPD) of AR media for generic capability exchange mechanisms
- Define capability exchange mechanisms based on complexity of AR media and capability of device to support EAS KPIs for provisioning of edge/cloud resources

NOTE: Identify a suitable existing capability framework, or if it does not exist, collaboration with broader industries (e.g., IETF, Khronos, W3C) is required.

- Define relevant KPIs and QoE Metrics for AR media
- Encapsulation into RTP and ISO/BMFF/CMAF

The media capabilities may be referenced and added to 3GPP Media service enablers and/or 3GPP service specifications such as 5G Media Streaming or MTSI.

8.6 Split Rendering Media Service Enabler with AR profile

In the context of this report, it was clearly identified that AR glasses depend on cloud or edge-based pre-rendering. However, not only AR glasses benefit from such a functionality, also for VR, XR and gaming, as identified in TR 26.928 and TR 26.926, would benefit from split rendering approaches. Hence, a basic Media Service Enabler for split rendering is paramount, in particular in combination with 5G new radio and 5G System capabilities.

Based on this discussion it is proposed to specify a generic raster-based split rendering media service enabler that includes, among others, the following functionalities:

- A content delivery protocol defined as a profile of 5G real-time communication for downlink with possible extension
- A relevant subset of codecs for different media types
- A scene description functionality to support a scene manager end point
- Relevant edge compute capabilities, for example Edge procedures, EAS profiles and KPIs for rendering, and rendering context relocation
- Relevant APIs and network communication
- Integration into 5GS and RAN, possibly with support of cross-layer optimizations
- Operational requirements and recommendations for low-latency communications
- Guidelines and examples

In addition to the generic enabler for split rendering a specific profile for AR is recommended to be defined that includes special considerations for:

- The formats to be supported on AR glasses
- The post-processing for pose correction and the integration with XR runtimes
- The power consumption challenge for AR glasses
- The metrics and KPIs for AR glasses
- The required QoS and QoE for AR type of applications as defined in clause 4.5
- Other AR specific considerations

8.7 Tethering AR Glasses

In clause 4.2.2.4, the important aspect of wireless tethering of AR glasses was introduced. The tethering technology between a UE and an AR glass may use different connectivity. Wireless tethered connectivity is provided through WiFi or 5G sidelink. BLE (Bluetooth Low Energy) connectivity may be used for audio. Two main types are identified:

- Functional structure for Type 3a: 5G Split Rendering WireLess Tethered AR UE
- Functional structure for Type 3b: 5G Relay WireLess Tethered AR UE

In the first case, the motion-to-render-to-photon loop runs from the glasses to the phone, whereas in the second case the 5G phone acts as a relay to forward IP packets. The architectures result in different QoS requirements, session handling properties, and also media handling aspects. For enhanced end-to-end QoS and/or QoE, AR glasses may need to provide functions beyond the basic tethering connectivity function, and the resulting AR glasses may be referred to as smartly tethering AR glasses. Generally, smartly tethering AR glasses is an important aspect. Based on these observations, it is proposed to further study this subject including specific topics such as:

- Defining different tethering architectures for AR Glasses including 5G sidelink and non-5G access based on existing 5G System functionalities
- Documenting end-to-end call flows for session setup and handling
- Identify media handling aspects of different tethering architectures
- Identify end-to-end QoS-handling for different tethering architectures and define supporting mechanisms to compensate for the non-5G link between the UE and the AR glasses
- Provide recommendations for suitable architectures to meet typical AR requirements such as low power consumption, low latency, high bitrates, security and reliability.
- Collaborate with relevant other 3GPP groups on this matter
- Identify potential normative work for stage-2 and stage-3

8.8 IMS-based AR conversational services

As identified in Table 6.1-1, AR conversational and shared AR conversational services have a number of related use cases. 3GPP TR 22.873 [14] also addresses use cases relevant to AR conversational services, namely conference calls with AR holography and AR calls, which have similarities with UC#19 and UC#4 in this study, respectively.

As documented in clause 6.5 and clause 6.6, AR conversational services and shared AR conversational experiences may be realized using various building blocks, including call setup and control, formats, delivery and 5G system integration, and these building blocks may have different instantiations and/or options.

In this study, the MTSI architecture is identified as one of the options to map those services to the 5G system. Furthermore, SA1's Rel-18 eMMTEL work item introduced new service requirements for 5G IMS Multimedia Telephony Service, including the support of AR media processing in TS 22.261[13] and it is expected that enhancements on the IMS architecture and/or IMS procedures to fulfil new requirements will be handled by SA2 in Rel-18.

It is proposed to define an IMS-based instantiation for a complete AR communication service, including:

- Terminal architecture(s) for various device types integrated with an MTSI client based on the work in summarized in clause 8.2, 8.4, 8.5, and 8.7
- IMS session setup, control, and capability exchange procedures for AR media in an IMS communication session
- Real-time transport of AR media, scene description, and metadata, as addressed in clause 4.4, via IMS media path including Data Channel

8.9 Audio Media Pipelines for AR Experiences

The current focus of this document is on general system aspects, especially targeting visual rendering on glasses. As such it may lack accuracy on audio media type. For example, extrapolations on architectural aspects derived for primarily visual media pipelines to audio pipelines need confirmation and further considerations. In particular, the following aspects may require further study:

- In device functional architecture, the type of audio capture and playback and the related system integration need to be defined.
- In 5G AR device types, the functional structures identified in this TR may be differentiated for immersive media types, e.g. operating immersive audio standalone while immersive video functions are split, involving tethered and/or cloud/edge entities.
- In the 5G system architecture mapping, the split of codecs and rendering assumed for video may not be appropriate for audio.

9 Conclusions

AR/MR experiences involve augmenting visual/auditory contents into the real world to improve the user's experience with better immersiveness, unlike VR, which provides an entirely virtual world. To realize these experiences, glass-type AR/MR devices may be a good candidate device, easily combining the lights from the real world and those from the display without a need of holding a device in one's hand.

In this study, the generic finding for eXtended Reality (XR) in TR 26.928 [2] have been further analysed with specific focus on Augmented Reality (AR) experiences and in particular also with a new device type, AR glasses. Different device centric functions of AR glasses are defined, and different device types are defined. Of particular relevance are 5G STandalone AR (STAR) UEs, i.e. devices that have sufficient capabilities to render rich AR experiences on the device as well as 5G EDGe-Dependent AR (EDGAR) UEs for which edge-based rendering support is a must to provide rich AR experiences. Three basic functions are introduced, the AR Runtime, the Scene Manager and the 5G Media Access Function. Basic AR processes are defined, and a comprehensive summary of AR related media formats is provided. The relevant work in external organizations is summarized.

Based on core use cases, different scenarios are mapped to the 5G System architecture, namely (i) Immersive media downlink streaming (ii) Interactive immersive services (iii) 5G cognitive/spatial computing immersive services as well (iv) AR conversational services. Potential normative work is identified and summarized in clause 8.

Based on the details in the report, the following next steps are proposed.

In the short-term:

- Document the relevant 5G generic architecture for real-time media delivery based on the 5GMS architecture as addressed in clause 8.2.
- Establish the concept of 5G media service enablers as introduced in clause 8.3 and make use of the concept to define relevant AR media service enablers.
- Define a 5G real-time communication media service enabler to support different low-latency streaming and conversational AR related services based on the considerations in clause 8.4.

- Define media capabilities for AR glasses in a service-independent manner based on the considerations in clause 8.5. The outcomes may affect the other items, especially the 5G real-time communication media service enabler and the IMS-based conversational services.
- Based on the work on above, define a split rendering media service enabler to support EDGAR devices, as addressed in clause 8.6.
- Study options for smartly tethering AR glasses based on the discussion in clause 8.7.
- Develop the extension of IMS-based AR conversational services and shared AR experiences, including an extended MTSI terminal architecture, as addressed in clause 8.8.
- Complement this TR with the relevant audio aspects in a follow-up study based on the considerations in clause 8.9.

In the mid-term:

- Add issues around semantical perception and spatial mapping to an AI/ML study, taking into account the findings in clause 4.2.3 and 4.2.5 as well as TR 22.874.

All work topics will benefit to be carried out in close coordination with other groups in 3GPP on 5G System and radio related matters, edge computing and rendering as well in communication with experts in MPEG on the MPEG-I project as well as with Khronos on their work on OpenXR, glTF and Vulkan/OpenGL. A follow-up workshop based on the information in clause 4.6.9 may be conducted in order to explore additional synergies and complementary work in different organizations in the XR/AR domain.

Annex A: Collection of Glass-type AR/MR Use Cases

A.1 Use Case 16: AR remote cooperation

Use Case Name
AR remote cooperation
Description
<p>As described in Annex A.9 of 3GPP TR26.928[2], a remote expert makes AR actions (e.g. overlaying graphics and drawing of instructions) to the received local video streams. This use case highlights that both parties can share their own video streams and overlay 2D/3D objects on top of these video streams compared with the scenario from TR 26.928.</p> <p>For example, a car technician contacts the technical support department of the car components manufacture by phone when he has some difficulty in repairing a consumers' car. The technical support department can arrange an engineer to help him remotely via real-time communication supporting AR.</p> <p>The car technician makes a video call with the remote engineer, uses his camera to capture the damaged parts of the car and shares them with the remote engineer in-call. And he marks possible points of failure by drawing instructions on the top of these video contents in order that the remote engineer can see the marks and make a detailed discussion. Also, they have respectively FOVs on their sides to check the failure. Likewise, the remote engineer can also overlay graphics and animated objects based on these shared video contents to adjust or correct the technician's operations. Furthermore, if the maintenance procedures are complex, the remote engineer can show the maintenance procedures step by step which are captured in real-time to the local technician. Therefore, the local technician can follow the operations. Finally, they find out the problems and fix them. It looks like that the remote engineer is beside the technician, discusses and solves the problems together.</p> <p>In the extension to this use case, if the remote engineer enables front-facing and back-facing cameras at the same time, the car technician can see a small video stream, which is captured by the front-facing camera of the remote engineer to achieve more attentive experiences.</p>
Categorization
<p>Type: AR, MR</p> <p>Degrees of Freedom: 3DoF+, 6DoF</p> <p>Delivery: Interactive, Conversational</p> <p>Device: XR5G-P1, XR5G-A2, XR5G-A3, XR5G-A4, XR5G-A5, others</p>
Preconditions
<p><provides conditions that are necessary to run the use case, for example support for functionalities on the end device or network></p> <p>Both parties on the device with the following features</p> <ul style="list-style-type: none"> - Support for conversational audio and video - Collect and delivery of AR actions and viewer information - Enabling of the front-facing and back-facing cameras at the same time <p>The network with the following features</p> <ul style="list-style-type: none"> - Rendering of overlying AR actions and viewer information

- Rendering of virtual and real superposition of different video contents
Requirements and QoS/QoE Considerations
<p><provides a summary on potential requirements as well as considerations on KPIs/QoE as well as QoS requirements></p> <p>QoS:</p> <ul style="list-style-type: none"> - conversational QoS requirements - sufficient bandwidth to delivery compressed 2D/3D objects <p>QoE:</p> <ul style="list-style-type: none"> - Synchronized rendering of overlay AR actions and pose information - Synchronized rendering of audio and video - Fast and accurate positioning information
Feasibility and Industry Practices
<p><How could the use case be implemented based on technologies available today or expected to be available in a foreseeable timeline, at most within 3 years?</p> <ul style="list-style-type: none"> - What are the technology challenges to make this use case happen? - Do you have any implementation information? <ul style="list-style-type: none"> - Demos - Proof of concept - Existing services - References - Could a reduced experience of the use case be implemented in an earlier timeframe or is it even available today? <p>></p> <p>Enhancements in media processing for multiple video streams both from different parties and/or the same party together with all kinds of AR actions may be performed in the network (e.g. by a media gateway) and in order to enable richer real-time experiences. Accordingly, the extensive hardware capabilities (e.g. multi-GPU) are required.</p>
Potential Standardization Status and Needs
<p><identifies potential standardization needs></p> <ul style="list-style-type: none"> - MTSI regular audio and video call between both parties - Standardized format for AR actions (e.g. static and/or dynamic 2D/3D objects) and posture information - Delivery protocols for AR actions and posture information - Rendering of more than one video stream

A.2 Use Case 17: AR remote advertising

Use Case Name
AR remote advertising
Description
<p>Compared with the use cases described in Annex A.8 and A.12 of 3GPP TR 26.928[2], this use case emphasizes that the shared video contents between two parties of a session are from a third party. Furthermore, the shared video contents may be 3D model objects, 360 degree and even free-viewpoint in order to help people have more interactive and immersive experiences.</p> <p>For example, a real estate salesman initiates an audio call to a client by his smartphone to advertise houses remotely. The real estate salesman can request some video contents which are restructured 3D objects for houses to be sold/rent in advance from the third content provider and then switch a video call. The real estate salesman and the client can receive the video contents from the third content provider simultaneously. The real estate salesman can introduce via audio while he rotates the model. At the same time, the client can hear the introduction and see the rotational model via the touch-screen of his smartphone. And vice versa, the client is able to ask what he cares via audio while he is marking in different colours on the shared model, the client can hear the questions and see the colourful marks in real-time.</p> <p>In an extension to the use case, if the video content is free-viewpoint and embed some 3D objects representing furniture, the client wearing an AR-glass is able to see layouts and furnishings of the virtual houses which can rendered following his posture. It seems that the client is just inside the advertised and virtual house, and is able to walk around different rooms (e.g., dining room and living room). Furthermore, the client can draw a 3D object for a small couch back and forth in the living room using his hand. In addition, the real estate salesman can insert his 3D animated model in the virtual house and it can move following the view scope of the client as if he is just beside the client and introduces the house to the client.</p> <p>In another extension to the use case, the client can invite his friend to see the virtual house together. They can see it from their respectively viewpoint. They can also walk around the virtual house when wearing an AR-glass and communicate with each other via audio.</p>
Categorization
<p>Type: AR, MR</p> <p>Degrees of Freedom: 6DoF</p> <p>Delivery: Interactive, Conversational, Download, Streaming</p> <p>Device: XR5G-P1, XR5G-A2, XR5G-A3, XR5G-A4, XR5G-A5, others</p>
Preconditions
<p><provides conditions that are necessary to run the use case, for example support for functionalities on the end device or network></p> <p>the devices with the following features</p> <ul style="list-style-type: none"> - Support for conversational audio and video - Support for receiving the video contents from the third party - Collecting of AR actions (e.g. rotation and mark) and posture information - Support of depth location technologies (e.g. SLAM for AR-glasses) <p>the network with the following features</p> <ul style="list-style-type: none"> - Rendering of overlying AR actions and posture information - Delivery of AR actions and posture information

- Support for establishing a connection the third party

Requirements and QoS/QoE Considerations

<provides a summary on potential requirements as well as considerations on KPIs/QoE as well as QoS requirements>

QoS:

- conversational QoS requirements
- sufficient bandwidth to delivery compressed 2D/3D objects
- Accurate user positioning information

QoE:

- Synchronized rendering of overlay AR actions and posture information
- Synchronized rendering of audio and video
- High-quality depth video captured from both parties

Feasibility and Industry Practices

<How could the use case be implemented based on technologies available today or expected to be available in a foreseeable timeline, at most within 3 years?

- What are the technology challenges to make this use case happen?
- Do you have any implementation information?
 - Demos
 - Proof of concept
 - Existing services
 - References
- Could a reduced experience of the use case be implemented in an earlier timeframe or is it even available today?

>

Enhancements in media processing for the media contents from the third party together with all kinds of AR actions and 2D/3D model objects may be performed in the network (e.g. by a media gateway) and in order to enable richer interactive and immersive experiences.

Potential Standardization Status and Needs

<identifies potential standardization needs>

- Delivery protocol of the shared media contents from the third party
- Standardized format and delivery protocols of AR actions and 2D/3D objects
- Standardized format and delivery protocols of posture information

- More than one communication channels can be setup

A.3 Use Case 18: Streaming of volumetric video for glass-type MR devices

A.3.1 Use case description

Use Case Description: Streaming volumetric video for glass-type MR devices

Bob and Patrick are gym instructors and run a gym 'VolFit'. 'VolFit' provides their clients with a mixed-reality application to choose and select different workout routines on a 5G-enabled OHMD. The workout routines are available as high-quality photorealistic volumetric videos of the different gym instructors performing the routines. Bob and Patrick book a professional capture studio for a high-quality photorealistic volumetric capture of the different workout routines for their clients. Bob and Patrick perform the workout routines in the studio capture area. The studio captures Bob and Patrick volumetrically.

Alice is a member of 'VolFit' gym. Alice owns a 5G-enabled glass-type OHMD device. The 'VolFit' MR application is installed on her OHMD. The OHMD has an untethered connection to a 5G network.

Alice wears her OHMD device. The MR application collects and maps spatial information of Alice's surrounding from the set of sensors available on the OHMD. The OHMD can further process the spatial mapping information to provide a semantic description of the Alice's surrounding.

Alice wants to learn a workout routine from her instructors, Bob and Patrick. The photorealistic volumetric videos of Alice's instructors are streamed to the MR application installed on her OHMD. The MR application allows Alice to position the volumetric representations of Bob and Patrick on real-world surfaces in her surroundings. Alice can move around with 6DoF, and view the volumetric videos from different angles. The volumetric representations are occluded by real-world objects in the XR view of Alice; when Alice move to a location where the volumetric objects are positioned behind real-world objects or vice-versa. During the workout session, Alice gets the illusion that Bob and Patrick are *physically present* in her surroundings, to teach her the workout routine effectively.

The MR application allows Alice to play, pause and rewind the volumetric videos. The functions can be triggered for example by hand-gestures, a dedicated controller connected to the OHMD, etc.

Categorization

Type: MR (XR5G-A1, XR5G-A2, XR5G-A4, XR5G-A5)

Degrees of Freedom: 6DoF

Delivery: Streaming, Split-rendering

Device: OHMD with/without a controller

Preconditions

- The application uses existing hardware capabilities on the device, including A/V decoders, rendering functionalities as well as sensors. Inside-out tracking is available.

- Spatial mapping to provide a detailed representation of real-world surfaces around the device
- Media is captured properly (refer to clause 4.6.7 of TR 26.928). The quality of the capture depends on different factors:
 1. Point-cloud based workflows
 - Studio setup i.e. camera lenses, distance of the captured object from the camera(s), stage lights
 - Filtering/Denoising algorithms
 2. Mesh-based workflows
 - Mesh reconstruction algorithms (e.g. Poisson surface reconstruction)
 - Geometric resolution of the object i.e. poly counts
 - Texture resolution e.g. 4K, 8K, etc.
- Media is accessible on a server
- Connectivity to the network is provided

Requirements and QoS/QoE Considerations

- QoS:
 - bitrates and latencies that are sufficient to stream a high-quality volumetric content within the immersive limits
 - bitrate for a single compressed volumetric video (mesh compression using tools such as Google Draco [30] and texture compression using video encoding tools such as H.264), for example, “Boxing trainer” sequence [31] further processed to generate a 3D mesh sequence with 65,000 triangles; 25fps, Texture: 2048x2048 pixels; 25fps:
 - o Data rate of 47.3Mbps, which constitutes of following:
 - Mesh sequence: 37 Mbps (using Google Draco [30])
 - Texture sequence: approximately 10 Mbps (encoding using H.264)
 - Audio: 133 kbps (AAC)
 - access link bitrate estimates in case of split-rendering delivery methods (multiple objects):
 - approximately 30% higher bitrate than typical video streaming due to ultra-low delay coding structure (e.g. IPPP)
 - left and right view (packed stereo frame)
 - bitrates of a compressed stereo video depend on rendered objects resolution
 - bitrates approximately 1Mbps (small objects)-35 Mbps (objects covering majority of the rendered viewport)
- Required QoE-related aspects:
 - volumetric video captured roughly in the range of ~1-10 million points per frame (this is dependent on capturing workflows as well as the level of details in captured object e.g. clothes’ textures)
 - high geometric resolution of the volumetric object’s geometry to achieve accurate realistic simulations of rendering equations
 - frame rate at least 30 FPS and above
 - high-quality content rendering according to the user’s viewpoint

- real-time rendering of multiple high-quality volumetric objects
- fast reaction to user's head and body movements
- fast reaction to hand-gestures, or a connected controller, etc.
- real-time content decoding
- accurate spatial mapping
- accurate tracking
- Desired QoE-related aspects:
 - accurate scene lighting [Note: PBR feasibility]

Feasibility

Volumetric content production:

- Volucap studios: <https://volucap.de/>
- Mixed Reality studio: <https://www.microsoft.com/en-us/mixed-reality/capture-studios>
- Metastage: <https://metastage.com/>

Device Features:

- Spatial mapping
- Tracking
- Scene understanding [32]
- A/V decode resources

Selected Devices/XR Platforms supporting this:

- Microsoft HoloLens™: <https://www.microsoft.com/en-us/hololens>
- Nreal Light™ glasses: <https://www.nreal.ai/>
- Magic Leap 1™: <https://www.magicleap.com/en-us/magic-leap-1>

Current solutions:

For a real-time mobile on-device mesh-based system, an acceptable-quality experience for 30 FPS can be achieved using at least 30,000-60,000 poly count for a volumetric object's geometry with at least 4K texture resolution. On-device rendering of multiple complex 3D models is limited by graphics capabilities of the device.

In addition, advanced rendering techniques for lighting, reflection and etc. are subject to complex rendering equations which may result in inconsistent frame rate and increased power consumption. Therefore, it is challenging to achieve a real-time volumetric streaming for multiple high-quality 3D models with current networks and on-device hardware resources. Some existing solutions use remote rendering for streaming volumetric video:

Azure remote rendering, <https://azure.microsoft.com/en-us/services/remote-rendering/>

, allows to render a huge and complex 3D model with millions of polygons remotely in cloud and stream in real-time to a MR device such as HoloLens 2™. An intuitive demonstration of the Azure™ remote rendering of 3D model with approximately 18 million polygons on HoloLens 2™ is publicly available at: <https://www.youtube.com/watch?v=XR1iaCcZPrU>

Mesh-based multiple high-quality volumetric video streaming using remote rendering [33]. More information is available at: <https://www.hhi.fraunhofer.de/5GXR>

Nvidia CloudXR: <https://developer.nvidia.com/nvidia-cloudxr-sdk>

Scene Lighting:

The light sources included in the scene have a significant impact on the rendering results. The light sources share some common properties such as;

- Colour: the colour of the light
- Intensity: the brightness of the light

Under Azure remote rendering, Scene lighting [34] provides the functionality to add different light types to a scene. Only the objects in the scene with PBR material type [35] are affected by light sources. Simpler material types such as Colour material [36] don't receive any kind of lighting.

Potential Standardization Status and Needs

The following aspects may require standardization work:

- Storage and access formats
- Network conditions that fulfill the QoS and QoE Requirements
- Relevant rendering APIs
- Scene composition and description
- Architecture design

A.3.2 Call flow for STAR UE

The call flow of the establishment of the AR session for STAR UE is shown in Fig. A.3.2-1:

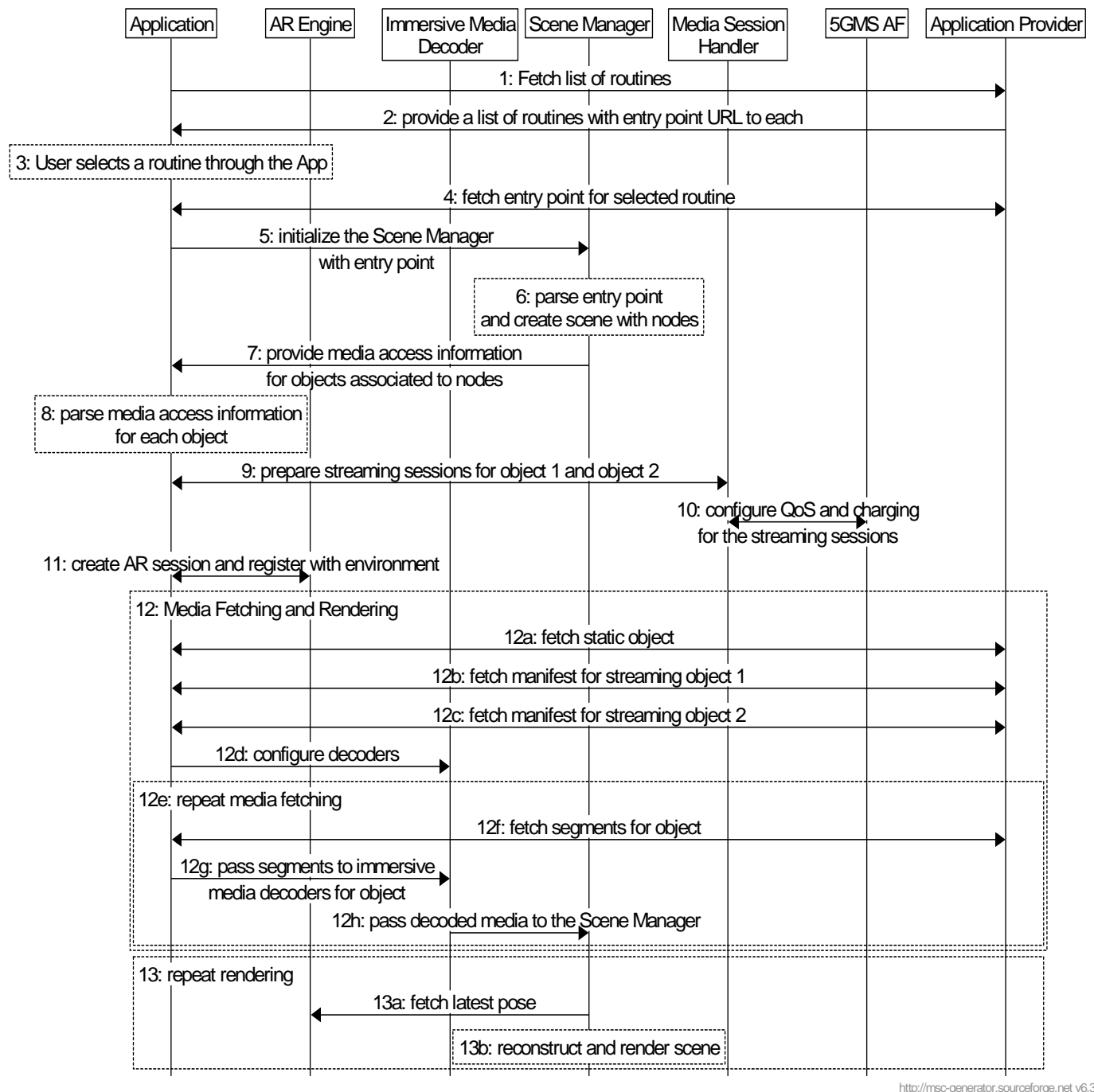

<http://msc-generator.sourceforge.net v6.3.8>

Fig. A.3.2-1: Call flow for STAR UE

A description of the steps is provided:

1. User starts the app. The app connects to the cloud to fetch a list of exercise routines for the user
2. The AP sends a list of routines to the app. Each routine is associated with an entry point for that routine. The entry point is typically a scene description that describes the objects in the scene and anchors the scene with the world space.
3. The user selects a routine in the app
4. The app fetches the scene description for the selected routine from the application provider
5. The app initializes the Scene Manager and passes the entry point to it.
6. The Scene Manager parses entry point to extract information about the required objects in the scene. In this case, the coach, the student, and a speaker are the 3 objects that will be rendered in the scene.

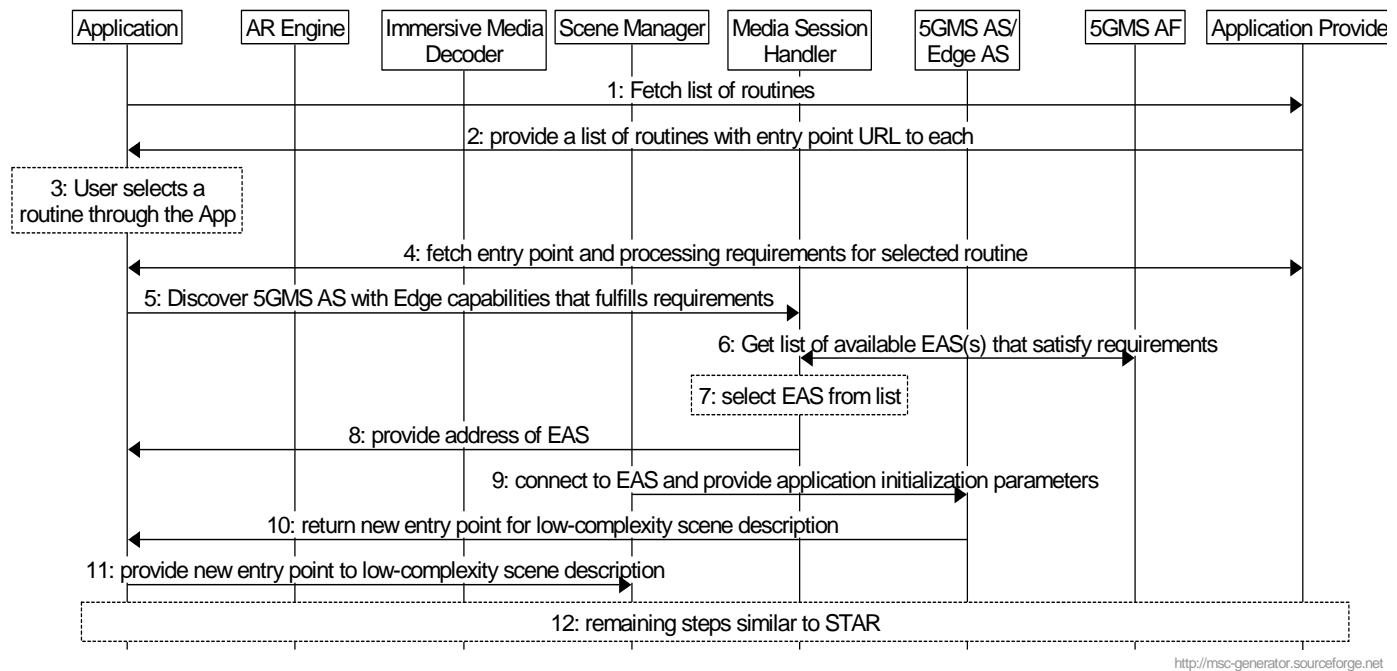
7. The Scene Manager informs the application about the required media that needs to be accessed.
8. The app parses the information about the media objects and determines how and when to access each of them.
9. The app informs the MSH that it will start 2 streaming sessions for the 2 dynamic objects.
10. The MSH shares the information with the AF and based on existing provisioning by the Application Provider, the AF may request QoS and charging modifications to the PDU sessions.
11. The App creates a new XR session and anchors the scene to a selected space in the XR session.
12. The media exchange begins:
 - a. The static object in the scene, the loudspeaker, is fetched by the App
 - b. The manifest for object 1 is retrieved
 - c. The manifest for object 2 is retrieved
 - d. The App configures the immersive video decoders based on the components of each object
 - e. Media segment for each component of each object is fetched
 - f. The segment is decoded and passed to the immersive media renderer
13. The Scene Manager periodically renders a frame by iteratively reconstructing each object and rendering it to a swapchain image. The swapchain image is passed to the AR Runtime for rendering to the HMD device.

The following media is assumed for this use case:

- An entry point: a scene description that describes the objects in the scene.
- Dynamic objects for the coach and student: these can be dynamic meshes, animated meshes, or point clouds
- Static object for the loudspeaker: this can be a static mesh
- Spatial audio: representing the vocal instructions associated with the dynamic object of the coach
- Spatial audio: representing the music for which the loudspeaker is the source

A.3.3 Call flow for EDGAR UE


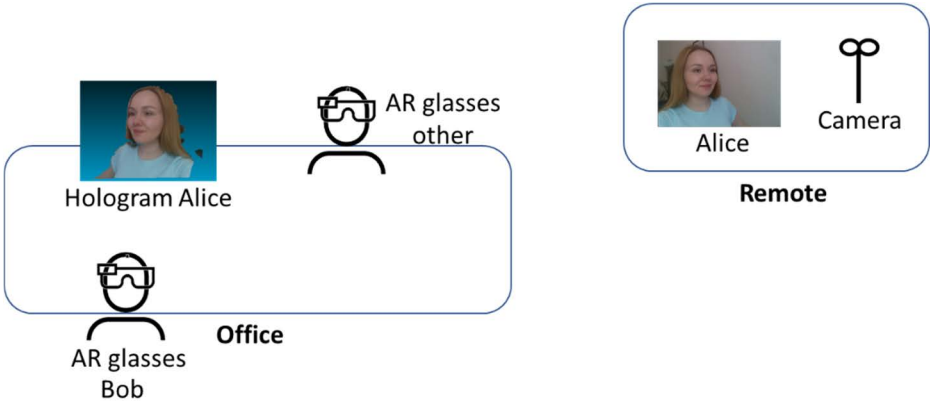
The call flow of the establishment of the AR session for EDGAR UE is shown in Fig. A.3.3-1:

**Fig. A.3.3-1: Call flow for EDGAR UE**

A description of the steps is provided:

1. User starts the app. The app connects to the cloud to fetch a list of exercise routines for the user
2. The AP sends a list of routines to the app. Each routine is associated with an entry point for that routine. The entry point is typically a scene description that describes the objects in the scene and anchors the scene with the world space.
3. The user selects a routine of preference in the app
4. The application sends a request for the entry point to the selected content. The Application Provider responds with an entry point to a scene description and a list of requirements for optimal processing of the scene.
5. The application determines that EDGE support is required and sends a request to the MSH to discover an appropriate Edge AS that can serve the application.
6. The MSH sends the requirements to the AF and receives a list of candidate EAS(s)
7. The MSH selects an appropriate EAS from the list of candidates.
8. The MSH provides the location of the EAS to the application.
9. The application connects to the EAS and provides initialization information. The initialization information contains: the URL to the scene description entry point or the actual scene description, its current processing capabilities, supported formats and protocols, etc.
10. The EAS configures the server application accordingly and generates a customized entry point for the client. The formats depend on the capabilities of the UE. The EAS adjusts the amount of processing performed by the EAS based on the current capabilities of the application. For example, The EAS may perform scene lighting and ray tracing and then generate a lightweight 3D scene description for the application. A less-capable UE may receive a more flattened scene, that contains stereo eye views and some depth information.
11. The App initializes the Scene Manager with the new low-complexity entry point.
12. The rest of the steps are similar to steps 6-10 from the STAR call flow.

A.4 Use Case 19: AR Conferencing

Use Case Description: AR Conferencing	
<p>This clause describes an AR conferencing use-case that allows participants in a 3D volumetric representation, e.g. point clouds or meshes, in order to provide an immersive conferencing experience.</p>	
<h3>3.2.1 AR Conferencing (1:1)</h3> <p>Bob and Alice want to make an AR conferencing call. Both are wearing AR glasses. Bob is located in Stockholm while Alice is located in Aachen. One or more cameras are placed in each location and are filming Bob and Alice, respectively. Bob can see a 3D volumetric representation of Alice on his AR headset and Alice can see a 3D volumetric representation of Bob on her AR headset. Bob and Alice can enjoy a truly immersive audio-visual experience.</p>  <p>The diagram illustrates a 1:1 AR conferencing setup. On the left, under the label 'Aachen', there is a box containing a small video feed of Alice at the top, an icon of a person wearing AR glasses labeled 'AR glasses Alice' below it, and an icon of a camera labeled 'Camera' to the right. On the right, under the label 'Stockholm', there is a similar box containing a small video feed of Bob at the top, an icon of a person wearing AR glasses labeled 'AR glasses Bob' below it, and an icon of a camera labeled 'Camera' to the right.</p>	
<p>Figure A.4-1: AR Conferencing (1:1)</p>	
<h3>3.2.1 AR Conferencing (1:many)</h3> <p>Bob and Alice are invited to an escalation meeting. Bob is able to physically attend the meeting, whereas Alice is virtually joining the meeting. Alice can be seen by Bob and other participants as a 3D volumetric representation on their AR glasses. Bob and other participants can interact with the 3D volumetric representations (e.g. rotate, zoom-in, resize). Alice can see and interact with Bob and other participants. Alice may use a laptop, phone, AR or VR device to visualize participants in the office. All participants can enjoy a truly immersive audio-visual experience.</p>  <p>The diagram illustrates a 1:many AR conferencing setup. On the left, under the label 'Office', there is a large box containing a 'Hologram Alice' (a small video feed of Alice) at the top left, an icon of a person wearing AR glasses labeled 'AR glasses other' at the top right, and an icon of a person wearing AR glasses labeled 'AR glasses Bob' at the bottom left. On the right, under the label 'Remote', there is a box containing a small video feed of Alice at the top left and an icon of a camera labeled 'Camera' at the top right.</p>	
<p>Figure A.4-2: AR Conferencing (1:many)</p>	
Categorization	
<p>Type: AR</p> <p>Degrees of Freedom: 3DoF+ or 6DoF</p> <p>Delivery: Conversational</p>	

Device: AR glasses
Preconditions
<ul style="list-style-type: none"> - The participants are located in a room that is equipped with cameras that allow the capturing of participants including depth information. The movements can be captured by other means (e.g. AR glasses or phone camera). - The participants are wearing AR glasses that allow the 3D volumetric representation of other participants.
Requirements and QoS/QoE Considerations
<p>The network is required support the delivery of 3D volumetric streams for real-time conversational services:</p> <ul style="list-style-type: none"> - Support of different volumetric user representation formats. - bitrates and latencies that are sufficient to stream volumetric user representations under conversational real-time constraints.
Feasibility
<p>The bandwidth and latency requirements for AR conferencing using 3D volumetric representations present a challenge to mobile networks. The complexity of the 3D volumetric representations is challenging for the endpoints and introduces additional delay for processing and rendering functions. Intermediate edge or cloud components are needed.</p> <p>In the following some indicative values of potential solution and transmission format for different types of user representation:</p> <ul style="list-style-type: none"> - A point cloud stream has raw bandwidth requirement of up to 2 Gbps. The transmission bandwidth is expected to be lower after encoding and optimization. - Preliminary data from MPEG V-PCC codec evaluation indicates compression ratios "in the range of 100:1 to 300:1"[40]. For dynamic sequences of 1M points per frame this could result into an encoding bitrate of "8 Mbps with good perceptual quality" [40]. For conversational services, we expect lower compression ratios. - 2D/RGB+Depth: >2.7Mbps (1 camera @ 30fps with total resolution of 1080x960 [37]), >5.4Mbps (2 Camera @ 30fps with total resolution of 1080x1,920 [38]). - 3D Mesh: ~30 Mbps @ 20-25 FPS (with a voxel grid resolution of 64x128x64 and 12-15k vertices) [39]. - Preliminary data from 3D GPCC show that bitrates in the range of 5-50 Mbps @ 30 fps with varying octree depth and varying JPEG QP are expected [39].
Potential Standardization Status and Needs
<p>The following aspects may require standardization work:</p> <ul style="list-style-type: none"> - Standardized formats for 3D volumetric representation of participants on AR glasses. - Cloud APIs for processing and rendering of 3D volumetric streams. - Conversational methods for call initiation. - Spatial audio formats and associated metadata. - Metadata for Spatial characteristics of the AR environment (e.g. positioning of users).

A.5 Use Case 20: AR IoT control

Use Case Description: AR IoT control
Many IoT devices are present in the home and several of them such as smart light bulbs, smart curtains, air conditioning systems, heaters or multimedia devices are present in multiple numbers in different rooms within the

home. While some IoT devices are at a fixed position and rarely move (light bulbs, heaters, curtains...) others are portable and nomadic by nature (portable speakers, vacuum cleaner robot, wearable devices...).

There are today many protocols that can be used for IoT (Wifi, Bluetooth, Zigbee™, io-homecontrol™, Z-wave™ but also LTE, NB-IoT and now Sidelink).and in the future there will be more and more IoT devices with 5G connectivity. While it is likely all IoT devices within a single home are interconnected through at least one (or more) gateway, D2D communications may also be used when available as it is likely to be more battery efficient for AR glasses.

As the user walk through his home, his AR glasses regularly scan the indoor environment to track the user's position in the home. While there could be several users wearing AR glasses in the same room, the environment reconstruction may also be using volumetric information from other IoT devices in the room (for instance security cameras with depth sensors). In terms of data exchange, upload data to 5G network may be video, depth-maps, sparse point clouds and also sensor information such as gyroscope or accelerometer.

Additionally, thanks to AR glasses' scanning, the home IoT system keeps track of IoT devices positions. Actual identification of an IoT device may also be done in the AR glasses themselves

Typically, the IoT home system runs on the edge network and information sent back to the AR glasses includes metadata information about IoT devices and environment, simple textual overlays for UIs. More advanced UIs would also probably make use of elements such as video or 3D objects

And finally, for D2D communications with 5G enabled IoT devices, control and status information is also exchanged between IoT devices and the AR glasses.

The use case addresses several scenarios:

- The user controls a specific IoT device by just looking at it through the glasses and operates it via an AR displayed user interface and user controls such as touch or voice controls available on the glasses.
- Since home IoT system can know in real-time the position of the user in the home as well as which IoT devices are present in the same room as the user, the user can control IoT device with simple voice control without even targeting a specific IoT device. For instance, the user can say: "switch off the light" to operate the main light of the room he is currently in.

Categorization

Type: AR

Degrees of Freedom: 6DoF

Delivery: Interactive, Split, device-to-device

Device: AR Glasses

Preconditions

- IoT Home Application is installed on the AR glasses or phone connected to AR glasses
- The application uses existing HW capabilities on the device, rendering functionalities as well as sensors (audio, video, Lidar). Inside-out Tracking is available thanks to AR glasses sensors and may also be enhanced thanks to similar sensing capabilities on IoT devices.
- Connectivity to the network is provided on the glasses or through the connected phone.
- Wayfinding and SLAM is provided to locate user and map the environment and may be provided with split-processing (tethered device or 5G edge network).
- AR and AI functionalities are provided for example for Image & Object Recognition in order to track IoT devices positions.
- Connectivity to IoT devices may be device-to-device (Bluetooth, Sidelink, ...), device to local IoT gateway (WiFi) or device-to-edge (5G NR).

Requirements and QoS/QoE Considerations

5G's low-latency high-bandwidth capabilities are used to provide real-time operation of IoT devices and high-speed upload of sensor information (video, Lidar point clouds).

Continuous connectivity is required for the sharing of local information to keep tracking user's position and position of IoT devices in the home so that home IoT system can maintain a real-time map of the home and its user.

The underlying AR maps are expected to be accurate and up to date.

In order to optimize energy consumption on the glasses, split processing is favoured and efficient compression technology is required for exchanges with the home IoT system.

Device-to-device communication may also be used with some IoT devices.

Feasibility

- Google Visual Positioning Service: <https://www.roadtovr.com/googles-visual-positioning-service-announced-tango-ar-platform/>
- XR clients continuously send sensing data to a cloud service. The service constructs a detailed and timely map from client contributions and provides the map back to clients. Example is Google's Visual Positioning Service
- Drivenet Maps – Open Data real-time road Maps for Autonomous Driving from 3D LIDAR point clouds: <https://sdi4apps.eu/2016/03/drivenet-maps-open-data-real-time-road-maps-for-autonomous-driving-from-3d-lidar-point-clouds/>
- An XR HMD receives a detailed reconstruction of a space, potentially captured by a device(s) with superior sensing and processing capabilities. An example of navigation is given in the MPEG-I use case document for point cloud compression (w16331, clause 2.6)
- Xiaomi MIOT Ecosystem – Around the MiHome application, users can control all MIOT devices from many brands : <https://xiaomi-mi.com/ecosystem/>
 - MIOT devices can be controlled locally or through remote locations thanks to cloud servers. Some MIOT devices such as security cameras or vacuum cleaner robots have the capability to track objects and capture the environment (Audio, Video, Lidar) and make it available to the IoT home system.

Potential Standardization Status and Needs

The following aspects may require standardization work:

- Data representations for AR map of the home
- Collected sensor data to be uploaded to edge or tethered device (with dedicated compression solutions)
- Scalable streaming and storage formats for AR maps
- Content delivery protocols to access AR maps and content items
- Content delivery protocols to send AR UI elements when IoT application runs on tethered device or edge.
- Network conditions that fulfil the QoS and QoE Requirements
- Device-to-device communications.

A.6 Use Case 21: AR gaming

Use Case Name
AR gaming
Description

By using AR technology, AR gaming is to apply virtual models to the real world and then provides game entertainment experience in a world of virtual-reality combination, which is difficult to experience in reality. People can interact with virtual objects controlled by game operations in the foreground of the real world. For many consumers, immersion is a key factor in their enjoyment of games, and AR can help users achieve such experiences exactly.

For example, Alice wears AR glasses at home and opens an AR golf game application. The AR glasses display a virtual sand table of golf course and golf ball through the spatial location on the floor of her house. Alice needs to use a certain gesture to hit the golf ball and make it go into the hole. Finally, the AR golf game application can calculate Alice's score.

AR gaming can also support multi-player games. Alice invites Bob and Clare to play an online AR shooting game in her living room. They wear AR glasses, log in to the AR shooting game application, select the multi-player team mode, and then they can control the shooting action through a gesture or the tethering device. The virtual target set in the shot game will appear in the living room, and have corresponding response according to their operation and display in their AR glasses. Then they can complete the game task and upgrade.

Categorization

Media Type: Visual / Audio, 3D

Media Source Type: CGI (Computer-Generated Imagery)

Delivery: Uplink, Download, Interactive

Device Type: XR5G-A2, XR5G-A3, XR5G-A4, XR5G-A5

Preconditions

- A game application on an AR device.
- Connectivity to the network
- Rendering according to user data
- Gesture acquisition or other control mode
- Spatial Computing Service
- Content synchronization

Requirements

- QoS:
 - Sufficiently low latency for rendering.
 - low packet loss rate
- QoE:
 - Gaming time before vertigo
 - Game jams

Feasibility and Industry Practices

There are some AR game demonstration examples using AR glasses.

1. The Tech Behind Tilt Five™ 's AR Gaming System



<https://www.youtube.com/watch?v=Z3qAio315ak>

2. *Super Mario Bros™* Recreated as Life Size Augmented Reality Game



<https://www.youtube.com/watch?v=QN95nNDtxjo>

During the 2018 World Mobile conference, China Mobile, Tencent and Huawei jointly announced the completion of AR game experimental verification based on 5G enhanced bandwidth stable delay network slice.

Potential Standardization Status and Needs

- Network conditions that fulfill the QoS and QoE requirements
- Architectures for computing support in the network
- Cloud APIs for group authentication and multiuser synchronization.
- Standardized format and delivery protocols of AR actions and 2D/3D objects
- Standardized format and delivery protocols of posture information
- Metadata for Spatial characteristics of the AR environment (e.g. positioning of users).
- Rendering of overlying AR actions and posture information for virtual and real superposition

A.7 Use Case 22: Shared AR Conferencing Experience

Use Case Description: Shared AR Conferencing experience

This clause describes an AR conferencing use case that allows participants to participate in a shared virtual conference room experience. For each participant, the other participants' video objects are registered on the participant's AR scene, creating the sense that the conference room is held in the physical location of the participant. The arrangement of participants (i.e. their location relative to each other) in the virtual room are the same which creates a consistent sense of conference room layout for all participants.

Bob, Alice, and Tom are participating in a virtual meeting. Bob is having the call at his home's kitchen and sees Alice and Tom with their 3D volumetric representation on his AR glasses. In Bob's view, Alice and Tom are sitting at Bob's kitchen table with Alice on left and Tom on the right-hand side. Alice is in her office conference room. She sees Bob and Tom their 3D volumetric representation on her AR glasses. For Alice, Tom is sitting on the left and Bob on the right-hand side of her conference table. Finally, Tom is having the call at the airport lounge. For him, Bob is sitting on a chair on the right and Alice is sitting on the couch on the left side. While the real worlds of Bob, Alice, and Tom are different, in all scenes the participants are seated in the same arrangement relative to each other. Therefore, when Alice turns to Tom, Tom and Bob see the consistent views of Alice as if they are in the same physical room.

Categorization

Type: AR

Degrees of Freedom: 3DoF+ or 6DoF

Delivery: Conversational

Device: AR glasses
Preconditions
- The participants are wearing AR glasses that allow the 3D volumetric representation of other participants.
Requirements and QoS/QoE Considerations
<p>The network is required to support the delivery of 3D volumetric streams for real-time conversational services:</p> <ul style="list-style-type: none"> - Support of creating a composed scene in the network - Support of different volumetric user representation formats. - Bitrates and latencies that are sufficient to stream volumetric user representations under conversational real-time constraints.
Feasibility
<p>The bandwidth and latency requirements for AR conferencing using 3D volumetric representations present a challenge to mobile networks. The complexity of the 3D volumetric representations is challenging for the endpoints and introduces additional delay for processing and rendering functions. Intermediate edge or cloud components are needed.</p> <p>In the following are some indicative values of a potential solution and transmission format for different types of user representation:</p> <ul style="list-style-type: none"> - A point cloud stream has raw bandwidth requirement of up to 2 Gbps. The transmission bandwidth is expected to be lower after encoding and optimization. - Preliminary data from MPEG V-PCC codec evaluation indicates compression ratios in the range of 100:1 to 300:1 [40]. For dynamic sequences of 1M points per frame this could result into an encoding bitrate of “8 Mbps with good perceptual quality” [40]. For conversational services, we expect lower compression ratios. - 2D/RGB+Depth: >2.7Mbps (1 camera @ 30fps with total resolution of 1080x960 [37]), >5.4Mbps (2 Camera @ 30fps with total resolution of 1080x1,920 [38]). - 3D Mesh: ~30 Mbps @ 20-25 FPS (with a voxel grid resolution of 64x128x64 and 12-15k vertices) [39]. - Preliminary data from 3D GPCC show that bitrates in the range of 5-50 Mbps @ 30 fps with varying octree depth and varying JPEG QP are expected [39].
Potential Standardization Status and Needs
<p>The following aspects may require standardization work:</p> <ul style="list-style-type: none"> - Standardized formats for 3D volumetric representation of participants on AR glasses. - Cloud APIs for processing and rendering of 3D volumetric streams. - Conversational methods for call initiation. - Spatial audio formats and associated metadata. - Metadata for Spatial characteristics of the AR environment (e.g. positioning of users).

Annex B:

Change history

Change history							
Date	Meeting	TDoc	CR	Rev	Cat	Subject/Comment	New version
2020-08	SA4#110					Initial draft	0.0.3
2020-11	SA4#111	S4-201496				Agreement during SA4#111e: Use cases added	0.1.0
2021-01	postSA4#111 VIDEO Adhoc	Document withdrawn by mistake				SA4#111e late agreements: S4-201410; S4-201497; S4-201508	0.2.0
2021-02	SA4#112	S4-210113				SA4#111e late agreements: S4-201410; S4-201497; S4-201508 Clause 4.2, 4.3, Annex A.5 updated (Agreement from SA4#111e) Clause 4.2 further updated (Agreement from telcos prior to SA4#112e)	0.3.0
2021-02	SA4#112	S4-210213				Editorial updates on the Change history. Basis for integration of SA4#112 agreements	0.3.1
2021-02	SA4#112	S4-210215				SA4#112 agreements: S4-210214; - Editorial correction in Annex A	0.4.0
2021-02	SA4#112	S4-210267				SA4#112 further agreements: S4-210221; S4-210224; S4-210269. - AR IoT use case - Clause 4.2.2: updates on device type name and functional structure - Structure of Clause 6	0.5.0
2021-04	SA4#113	S4-210510				SA4#post-112e agreements: S4aV200656, S4aV200670 - Clause 6.2.4: Procedure & call flows - Clause 4.2.2.2, 4.2.2.4: updates on possible support from 5G cloud/edge	0.6.0
2021-04	SA4#113e	S4-210554				SA4#113e agreements: S4-210505, S4-210553	0.7.0
2021-05	SA4#114e	S4-210874				SA4#114e agreements: S4-210755, S4-210756, S4-210857, S4-210877, S4-210878, S4-210879, S4-210880, S4-210884, S4-210885, S4-210892	0.8.0
2021-08	SA4#115e	S4-211200				SA4#post-114e telco agreements: S4aV210698, S4aV210717, S4aV210718, S4aV210724 SA4#115e agreements: S4-2111032, S4-211189, S4-211190, S4-211191, S4-211192, S4-211193, S4-211194, S4-211195, S4-211196, S4-211197, S4-211198, S4-211199, S4-211211, S4-211212, S4-211214, S4-211321	0.9.0
2021-09	SA#93-e					For presentation to plenary	1.0.0
2021-09	postSA4#115 telco	S4aV210764				SA4#115e agreements: S4-211213	1.0.1
2021-09	postSA4#115 telco	S4aV210778				SA4#post-115e agreements: S4aV210755	1.0.2
2021-10	postSA4#115 telco	S4aV210813				SA4#post-115e agreements: S4aV210802	1.0.3
2021-10	postSA4#115 telco	S4aV210813				SA4#post-115e agreements: S4aV210814	1.0.4
2021-11	SA4#116e	S4-211545				SA4#116e agreements: S4-211372, S4-211388, S4-211407, S4-211484, S4-211546, S4-211547, S4-211548, S4-211549, S4-211550, S4-211551, S4-211559, S4-211560, S4-211683, S4-211570, S4-211571, S4-211572	1.1.0
2022-01	postSA4#116 telco	S4aV220849				SA4#post-116e agreements: S4aV220849	1.1.1
2022-02	postSA4#116 telco	S4-220060				SA4#post-116e agreements: S4aV220856, S4aV220859, S4aV220863	1.1.2
2022-02	SA4#117e	S4-220198				SA4#117e agreements: S4-220131, S4-220199, S4-220200, S4-220201, S4-220203, S4-220208, S4-220209, S4-220210, S4-220211, S4-220223	1.2.0
2022-03	SA#95-e	SP-220254				Presentation to SA for approval	2.0.0
2022-03	SA#95-e	SP-220254				Under change control	17.0.0
2022-09	SA#97-e	SP-220761	0001	1	F	Corrections to TR26.998	17.1.0

History

Document history		
V17.0.0	April 2022	Publication
V17.1.0	October 2022	Publication