ETSI TS 126 119 V19.1.0 (2026-04)

TECHNICAL SPECIFICATION

LTE;
5G;
Media Capabilities for Augmented Reality
(3GPP TS 26.119 version 19.1.0 Release 19)

Reference
RTS/TSGS-0426119vj10

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

Intellectual Property Rights

Essential patents

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

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

Trademarks

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

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

Legal Notice

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

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

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

Modal verbs terminology

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

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

Contents

Foreword

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

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

Version x.y.z

where:

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

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

shall indicates a mandatory requirement to do something

shall not indicates an interdiction (prohibition) to do something

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

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

should indicates a recommendation to do something

should not indicates a recommendation not to do something

may indicates permission to do something

need not indicates permission not to do something

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

can indicates that something is possible

cannot indicates that something is impossible

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

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

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

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

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

In addition:

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

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

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

Introduction

The present document provides technologies for the deployment of Augmented Reality (AR) as defined in 3GPP TR 26.928 [2] and the execution of Augmented Reality applications on targeted devices such as those identified in 3GPP TR 26.998 [3].

On the spectrum of eXtended Reality (XR) experiences, Augmented Reality overlay virtual information on top of the user’s perception of the real environment. Those virtual and real components of the scene seamlessly blend together from the user’s perspective. Additionally, some AR experiences can enable interactivity between the user and the virtual components of the scene.

In the present document, the focus lies in the definition of the media capabilities for AR devices, including media format encapsulation capabilities, media codec capabilities, processing function capabilities. The related minimum required performances for different device types are also defined.

1 Scope

The present document defines the supported media formats, codecs, processing functions for XR Devices in UE per XR device type category. The present document addresses the interoperability gaps identified in the conclusions of TR 26.998 [3].

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] 3GPP TR 26.998: "Support of 5G glass-type Augmented Reality / Mixed Reality (AR/MR) devices".

[4] 3GPP TR 26.857: "5G Media Service Enablers".

[5] Khronos, "The OpenXR Specification", https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html.

[6] 3GPP TS 26.506: "5G Real-time Media Communication Architecture (Stage 2)".

[7] ITU-T Recommendation H.264 (08/2021): "Advanced video coding for generic audiovisual services".

[8] ITU-T Recommendation H.265 (08/2021): "High efficiency video coding".

[9] 3GPP TS 26.117: "5G Media Streaming (5GMS); Speech and audio profiles".

[10] ISO/IEC 12113:2022 Information technology Runtime 3D asset delivery format Khronos glTF™2.0

[11] ISO/IEC 23090-14:2023 Information technology Coded representation of immersive media Part 14: Scene description

[12] ISO/IEC 23090-14:2023/Amd 1:2023 Information technology Coded representation of immersive media Part 14: Scene description

[13] ISO/IEC 23090-14:2023/DAmd 2 Information technology Coded representation of immersive media Part 14: Scene description

3 Definitions of terms, symbols and abbreviations

3.1 Terms

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

Anchor: a virtual element for which its position, orientation, scale and other properties are expressed in the trackable space defined by the trackable. A virtual asset's position, orientation, scale and other properties are expressed in relation to an anchor.

Media Session Handler: a set of functions responsible for handling all 5G control plane operations, such as requesting network assistance, discovering and allocating edge resources, etc.

Presentation Engine: a set of composite renderers, rendering the component of the scenes.

Reference Points: geometric points whose position is identified both mathematically and physically.

Trackable: a real-world object that can be tracked by the XR runtime. Each trackable provides a local reference space, also known as a trackable space, in which an anchor can be expressed.

Swapchain: a queue of images shared between the XR Application and the XR Runtime

Swapchain image: image in a swapchain.

XR Application: application running on an XR Device which offers an XR experience based on an XR Runtime.

XR Device: a device capable of offering an XR experience.

XR Runtime: Set of functions provided by the XR Device to the XR Application to create XR experiences.

XR Runtime API: the API to communicate with an XR Runtime.

XR Scene Manager: a set of functions that supports the application in arranging the logical and spatial representations.

XR Session: an application's intention to present XR content to the user.

XR Source Management: management of data sources provided through the XR runtime.

XR System: a collection of resources and capabilities from the XR Runtime exposed to the XR Application for the duration of the XR Session.

XR View: a rendered view of the scene generated by the XR Application and passed on to the XR Runtime during a running XR Session**XR Space**: a frame of reference in which 3D coordinates are expressed.

Warping: correcting the rendered image based on the latest head pose estimation

3.2 Abbreviations

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

API	Application Programming Interface
AR	Augmented Reality
AAC	Advanced Audio Coding
AVC	Advanced Video Coding
CPB	Coded Picture Buffer
DPB	Decoded Picture Buffer
ELD	Enhanced Low Delay
EVS	Enhanced Voice Services
glTF	graphics library Transmission Format
GLB	glTF Binary
HEVC	High Efficiency Video Coding
HMD	Head-Mounted Display
HRD	Hypothetical Reference Decoder
HSS	Hypothetical Stream Scheduler
IVAS	Immersive Voice and Audio Services
JSON	JavaScript Object Notation
MPEG	Moving Picture Expert Group
MPEG SD	MPEG Scene Description

MR	Mixed Reality
OP	Observation Point
SLAM	Simultaneous Localisation And Mapping
UE	User Equipment
VCL	Video Coding Layer
VR	Virtual Reality
XR	eXtended Reality

4 XR concepts and device types

4.1 XR concepts

4.1.1 General

Extended Reality (XR) refers to a continuum of experiences combine real-a and- virtual combined environments in which the user is immersed through one or more devices capable of audio, visual and haptics rendering generated by computers through human-machine interaction. XR encompasses technologies associated with Virtual Reality (VR), Augmented Reality (AR) and Mixed Reality (MR) which constitute the so-called XR continuum. A detailed overview of definitions, concepts and background on XR and AR is provided in TR 26.928 [2] and TR 26.998 [3], respectively.

The terms Augmented Reality, Virtual Reality, Mixed Reality and eXtended Reality as used throughout this document are defined in Clause 4.1 of 3GPP TR 26.928 [2].

4.1.2 XR Device

An XR device is capable of offering an XR experience. An XR Device is assumed to have one or several displays, speakers, sensors, cameras, microphones, actuators, controllers and/or other peripherals that allow to create XR experiences, i.e. experiences for which the user interacts with the content presented in virtual world and/or augmented to the real-world. Example of XR Devices are AR Glasses, a VR/MR Head-Mounted Display (HMD) or a regular smartphone, etc.

4.1.3 XR application

An application which offers an XR experience by making use of the hardware capabilities, including media capabilities, of the XR Device it runs on as well as the network connectivity to retrieve the asset being used by the application is referred to as an XR Application. In the context of this specification, it is primarily assumed that access to the network is provided by 5G System functionalities.

To enable XR experiences, the hardware on an XR Device typically offers a set of functions to perform commonly required XR operations. These operations include, but are not limited to:

- accessing controller/peripheral state,
- getting current and/or predicted tracking positions and pose information of the user,
- receiving or generating pre-rendered views of the scene for final presentation to the user, taking into account the latest user position and pose. Adaptation to the latest user position and pose is also referred to as warping.

4.1.4 XR Runtime

4.1.4.1 General

XR Runtime provides a set of functionalities to XR applications including but not limited to peripheral management, runtime functions as tracking, SLAM, composition and warping etc. The functions are accessible to the XR Application via an API exposed by the XR Runtime referred to as the XR Runtime Application Programming Interface (XR API). The XR Runtime typically handles functionalities such as composition, peripheral management, tracking, Spatial Localization and Mapping (SLAM), capturing and audio-related functions. Further, it is assumed that the hardware and software capabilities of the XR Device are accessible through well-defined device APIs, and in particular the media capabilities are accessible through media APIs.

An overview of an XR Device logical components is shown in Figure 4.1.4-1.

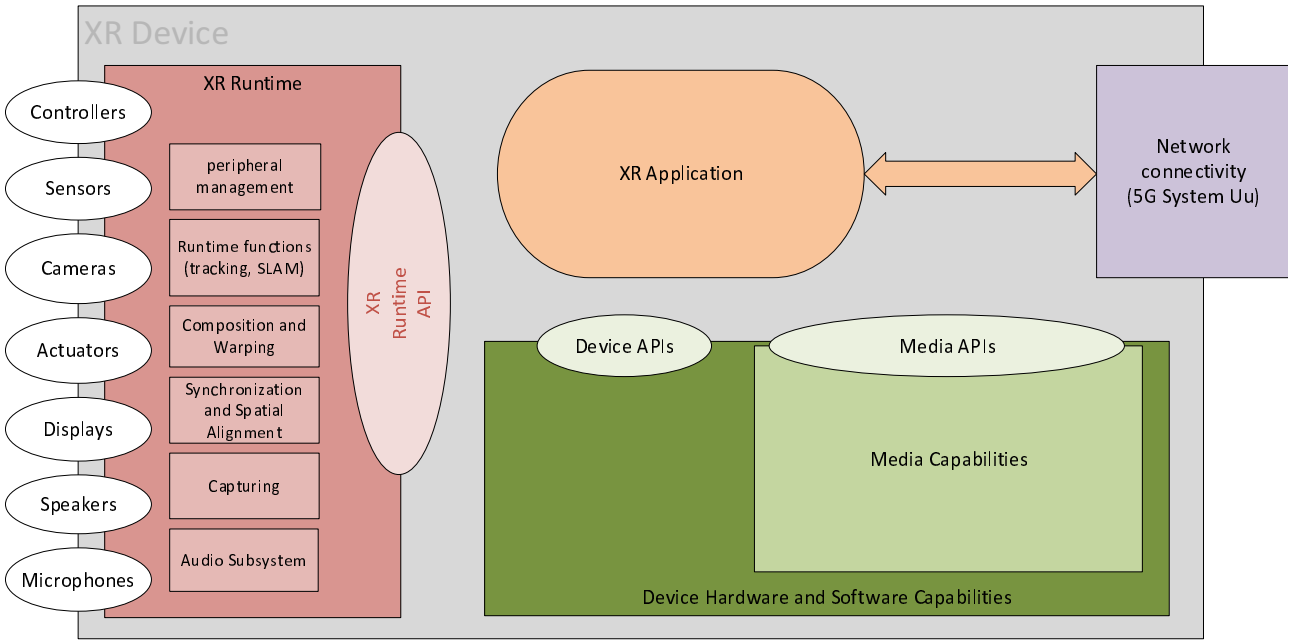

Figure 4.1.4-1 Logical components of an XR Device

This specification relies on a hypothetical XR Runtime and its API in order to define the media capabilities. This way, different implementation of XR runtimes may be compatible with this specification. However, for the purpose of developing this specification, the minimal set of expected functionalities of the XR Runtime has been aligned with the core Khronos' OpenXR specification [5]. Support for other XR Runtime environments is not precluded by this approach. Lastly, a mapping of general functionalities to OpenXR is provided in Annex B.

4.1.4.2 XR session and rendering loop using XR Runtime (informative)

At startup, the XR Application creates an XR Session via the XR Runtime API and allocates the necessary resources from the available resources on the XR Device. Upon success, the XR Runtime begins the life cycle of the XR Session whose cycle is typically made of several states. The purpose of those states is to synchronise the rendering operations controlled by the XR Application with the display operations controlled by the XR Runtime. The rendering loop is thus a task jointly executed by the XR Runtime and the XR Application and synchronised via the states of the XR Session.

The XR Application is responsible of generating a rendered view of the scene from the perspective of the user. To this end, the XR Application produces XR Views which are passed to the XR Runtime at iterations of the rendering loop. The XR Views are generated for one or more poses in the scene for which the XR application can render images. From those views, the view corresponding to the viewer's pose is typically called the primary view. There may be other XR Views defined in the scene, for instance for spectators.

The XR Views are configured based on the display properties of the XR Device. A typical head-mounted XR System has a stereoscopic view configuration, i.e. two views, while a handheld XR Device has a monoscopic view configuration, i.e. a single view. Other view configurations may exist. At the start of session, the XR Application configures the view type based on those device properties which remains the same for the duration of the XR Session.

A XR View may also comprise one more composition layers associated with an image buffer. Those layers are then composed together by the XR Runtime to form the final rendered images.

In addition to layers containing visual data, an XR View may be complemented with a layer provided depth information of the scene associated with this XR View. This additional information may help the XR Runtime to perform pose correction when generating the final display buffer. Another type of layer can be an alpha channel layer useful for blending the XR View with the real environment for video-see through XR devices, e.g. which is the case for AR applications running on smartphones.

For the XR Application to render the XR Views, the XR Runtime provides the viewer pose as well as projection parameters which are typically taken into account by applications to render those different XR Views. The viewer pose and projection parameters are provided for a given display time in the near future. The XR Runtime accepts repeated

calls for prediction updates of the pose, which may not necessarily return the same result for the same target display time. Instead, the prediction gets increasingly accurate as the function is called closer to the given time for which a prediction is made. This allows an application to prepare the predicted views early enough to account for the amount of latency in the rendering while at the same time minimising the prediction error when pre-rendering the views.

In addition, the XR Application communicates with input devices in order to collect actions. Actions are created at initialization time and later used to request input device state, create action spaces, or control haptic events. Input action handles represent 'actions' that the application is interested in obtaining the state of, not direct input device hardware.

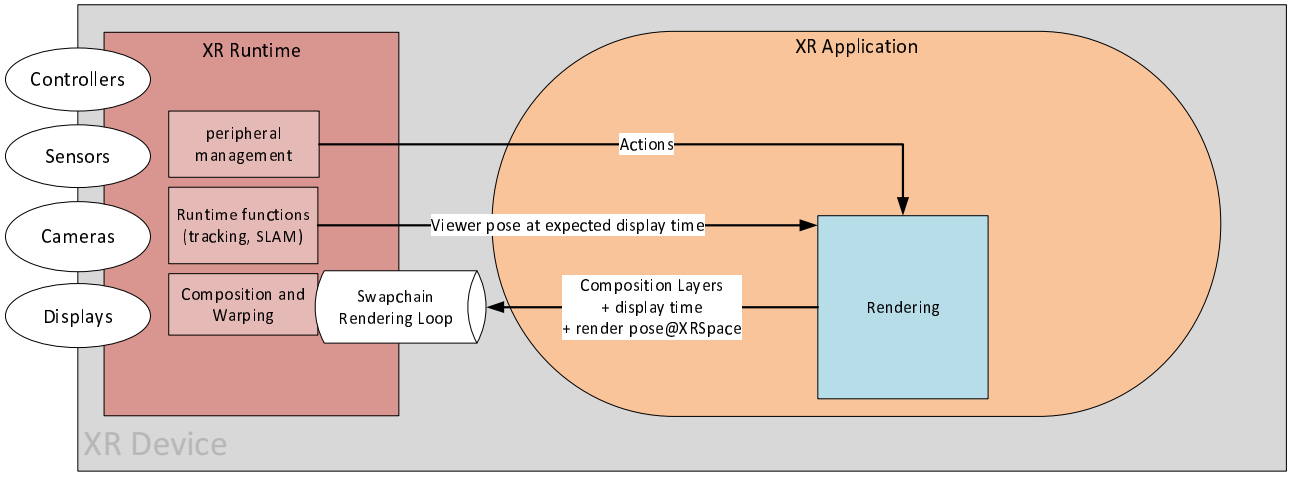

Figure 4.1.4-1 Rendering loop for visual data

4.2 Media pipelines and rendering loop

In the context of this specification, media to be rendered and displayed by the XR Device through the XR Runtime is typically available in an compressed form on the device. Media is accessed using a 5G System, decoded in the device using media capabilities, and then the decoded media is rendered to be provided through swapchains to the XR Runtime as shown in Figure 4.2-1.

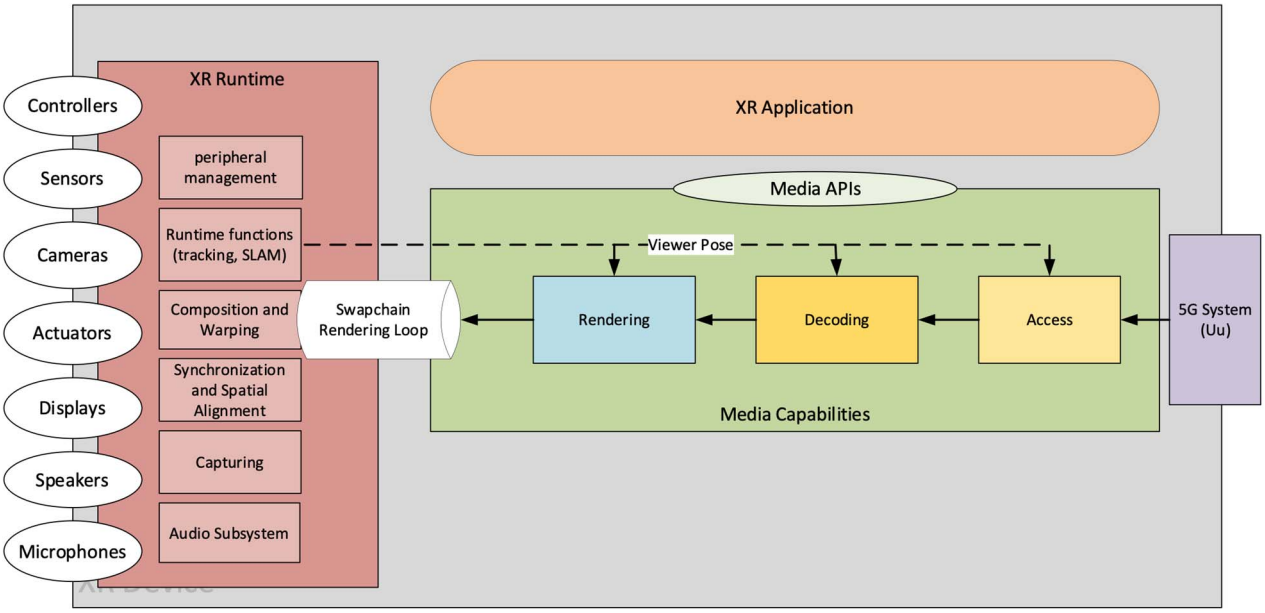

Figure 4.2-1 Media pipelines: Access, decoding and rendering

The rendering function is responsible of generating the content that will be presented by the XR Runtime. This rendering function makes use of rendering loops and provide the results of those loops to the XR Runtime via

swapchains. The application sets up different pipeline that comprise processes for media access, decoding, and view rendering. To configure those pipelines and the properties of the generated views (e.g. number of layers, stereoscopic/monoscope views), the rendering function needs to have access to the information about the current session defined at the initialisation step:

- View configuration
- Blend modes
- XR spaces
- swap chain formats and images
- projection layer types

4.3 Device Types

4.3.1 Device type 1: Thin AR glasses

The thin AR glasses device type represents a type of device which is considered as power-constrained and with limited computing power with respect to the other device types. These limitations typically come from the requirement to design a device with a small and lightweight form factor. Regarding rendering capacity, this device type is expected to rely on remote rendering to be able display complex scenes to the user. For example, such device type may run a split rendering session where the split rendering server delivers pre-rendered views of the scene. However, devices in this category can still operate without external support for applications that do not require complex rendering capabilities, for instance, text messaging, 2D video communication, etc. Lastly, the thin AR glasses offers AR experiences to the user via optical see-through display.

4.3.2 Device type 2: AR glasses

The AR glasses device type represents a type of device which is considered to have higher computation power compared to the thin AR glasses device type. As a result, this AR device type has higher rendering capacities and is generally expected to be capable of rendering scenes without external support, even though remote rendering is not precluded to lower the power consumption on the device or enable the display of scenes beyond the device's rendering capability. Lastly, the AR glasses offers AR experiences to the user via optical see-through display.

4.3.3 Device type 3: XR phone

The XR phone device type represents a type of device which corresponds to a smartphone with capacities and resources sufficient to offer AR experiences. As a result, this device type is capable of rendering scenes without external support. Lastly, the XR phone offers AR experiences to the user via video see-through display.

4.3.4 Device type 4: XR Head Mounted Display (HMD)

The XR HMD device type represents a type of device which corresponds to HMDs capable of offering at least AR experiences but not precluding other types of XR experiences. This device type is expected to be capable of rendering scenes without external support. Lastly, the XR phone offers AR experiences to the user via video see-through display.

5 Device reference architecture and interfaces

5.1 Architecture

The XR Baseline Client represents the functionalities, the peripherals, and the interfaces that are present on a generic XR UE. The XR Baseline Client reference architecture is shown in Figure 5.1-1. The actual device may be realized by a single device, or a combination of devices linked together. The details on how to instantiate an XR Baseline Client in the context of a service or deployment scenario is left for the respective Work Items and Study Items to define.

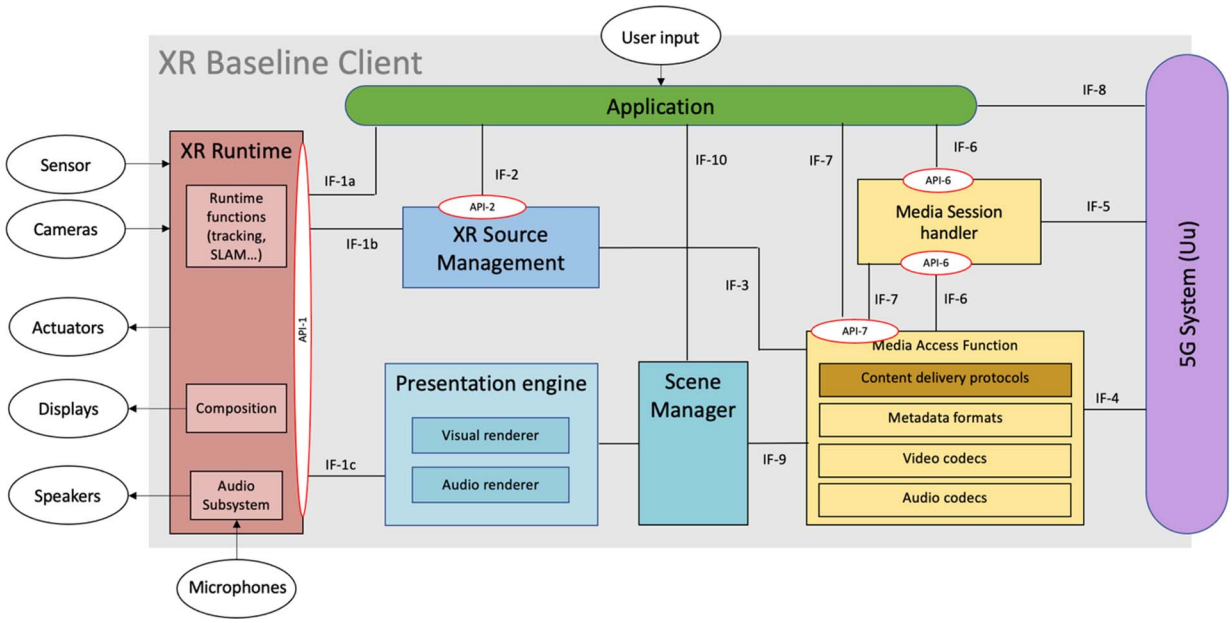

Figure 5.1-1 - XR Baseline terminal architecture and interfaces

5.2 Description of the functional blocks

In terms of functionalities, an XR Baseline Client is composed of:

- An **XR Application**: a software application that integrates audio-visual content into the user's real-world environment
- An **XR Runtime**: a set of functions that interface with a platform to perform commonly required operations, such as accessing the controller/peripheral state, getting current and/or predicted tracking positions, performing spatial computing, as well as submitting rendered frames to the display processing unit and rendered audio to the speakers with a late stage re-projection to the latest pose (see clause 4.1.3).
- An **XR Source Management**: management of data sources provided through the XR runtime such as microphones, cameras, trackers, etc, for instance, making the information available to the XR application or providing it to the MAF for sending in the uplink.
- A **Media Access Function**: A set of functions that enables access to media and other XR-related data that is needed in the Scene manager or XR Runtime to provide an XR experience as well to create delivery formats for information provided by the XR Source Management.
- A **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 XR Runtime.
- A **Presentation Engine**: a set of composite renderers, rendering the component of the scenes, based on the input from the Scene Manager.
- A **Media Session Handler**: a set of functions responsible for handling all 5G control plane operations, such as requesting network assistance, discovering and allocating edge resources, etc. This may be realized as a 5G-RTC MSH, 5GMS Media Session Handler, or any other function. In addition, those functional blocks are integrated together via interfaces. Interfaces may be made of APIs and/or data formats and collectively act as a contract between the two sides of the interface.

In addition, those functional blocks are integrated together via interfaces. Interfaces may be made of APIs and/or data formats and collectively act as a contract between the two sides of the interface.

5.3 Interfaces and APIs

The XR Baseline Client contains the following interfaces:

- **IF-1** for the XR Runtime on one side and the Application (1a), the XR Source Management (1b) and the Presentation Engine (1c). IF-1a-c is implemented as an API (API-1) that exposes functions provided by the XR Runtime. An example of this API is the Khronos OpenXR API.
- **IF-2** describes the functions exposed by the XR Source Management that can be accessed and controlled by the XR application, or possibly other functions in the device. IF-2 is typically implemented as an API.
- **IF-3** lies between the XR Source Management and the Media Access Function and provides serialized information accessible on XR Runtime to the MAF.
- **IF-4** between the Media Access Function and the 5G System for user plane data, such as application data or other graphics data needed by the XR application.
- **IF-5** lies between the UE and the 5G System, implementing control sessions. An example instance of this interface is the RTC-5 interface as defined by TS 26.506 [6].
- **IF-6** connects the Media Session Handler and the Application/MAF. It offers the tools for them to activate 5G media functionality such as network assistance and edge resource discovery. The IF-6 is realized through an API (API-6).
- **IF-7** lies between the XR Application and the Media Access function to configure Media Access. This is typically implemented as an API (API-7) that exposes functions of the MAF.
- **IF-8** is an interface that allows the XR application to make use of 5G System connectivity.
- **IF-9** between the Scene Manager and the Media Access Function.
- **IF-10** between the Scene Manager and the XR Application.

6 General system functions and capabilities

6.1 XR system capabilities

The XR Runtime comprises functions and hardware components present on the XR Device. However, those functions and hardware components are not directly exposed to the XR Application. Instead, the XR Runtime offers its functions and hardware components via an XR System. A single XR Runtime may expose more than one XR Systems for targeting different purposes, e.g., a handheld device may have two XR Systems, one when the user holds the device and one when the device is inserted into an HMD. At the start of the XR Application, the XR Application is expected to query what XR Systems are available on the XR Device and select one of them to create the XR Session.

Table 6.1-1 provides capabilities for XR Runtimes exposed through an XR System. This table does not prescribe support for any specific capabilities of an XR device as defined in clause 4.3. The support of XR System capabilities is defined per device category in clause 10. A mapping of these high-level capabilities to XR frameworks are provided in Annex B.

Table 6.1-1 XR System capabilities

Capability	Description	Parameters	Value type	Parameter definitions
Create a XR System	An application can create a XR System from an XR Runtime.	`xrSystemIdentifier`	integer	Identifier of a given XR System exposed by a XR Runtime.
Query XR System's graphics properties	An application can query an XR System about its	`swapchainSupported`	boolean	Indicates whether the XR System supports the swapchains.
		`maxSwapchainImageHeight`	integer	The maximum swapchain image pixel

	graphics capabilities.			height supported by this XR system.
		`maxSwapchainImageWidth`	integer	The maximum swapchain image pixel height supported by this XR system.
		`maxLayerCount`	integer	The maximum number of composition layers supported by this XR system
Query XR System's tracking properties	An application can query an XR System on the tracking capabilities.	`orientationTracking`	boolean	Indicates whether the XR System supports orientational tracking of the view pose(s), or not.
		`positionTracking`	boolean	Indicates whether the XR system supports positional tracking of the view pose(s),
Enumerate XR System's supported environment blend modes	An application can query an XR System about its supported environment blend modes, see clause [xxx].	`blendMode`	['`opaque`', '`additive`', '`alpha_blend`']	Indicates the type of blend mode supported by the XR System. The value '`opaque`' relates to the opaque blend mode, the value '`additive`' to the additive blend mode and '`alpha_blend`' to the alpha blend mode.
Enumarate supported view configuration types	An application can query an XR System about the its supported primary view configurations.	`viewConfigurationPrimary`	['`monoscopic`', '`stereoscopic`', '`other`']	Indicates the type of primary view configuration of the XR System. The value '`monoscopic`' relates to a single view, the value '`stereoscopic`' to the left and right-eye views and '`other`' to a type undefined in the scope of this specification.
Enumerate the view configuration properties	An application can list the properties associated with different view configurations advertised by an XR System.	`recommendedImageRectWidth`	integer	The optimal width of imageRect to use when rendering this view into a swapchain.
		`maxImageRectWidth`	integer	The maximum width of imageRect supported when rendering this view into a swapchain.
		`recommendedImageRectHeight`	integer	The optimal height of imageRect to use when

				rendering this view into a swapchain
		`maxImageRectHeight`	integer	The maximum height of imageRect supported when rendering this view into a swapchain.
		`recommendedSwapchainSampleCount`	integer	The recommended number of sub-data element samples to create for each swapchain image that will be rendered into for this view.
		`maxSwapchainSampleCount`	integer	The maximum number of sub-data element samples supported for swapchain images that will be rendered into for this view.
Enumerate reference space types	An application can query an XR System about the supported reference space types, described in [xxx].	`referenceSpaceView`	['`view`', '`local`', '`stage`', '`unbounded`', '`user_defined`']	Indicates the type of reference spaces supported by the XR System. The value '`view`' relates to view reference space, the value '`local`' to the local reference space, the value '`stage`' to the stage reference space, the value '`unbounded`'.
Query the spatial range boundaries	An application can query the spatial ranges in which an XR experience may be rendered.	`2DSpatialRangeBoundaries`	rectangle	Provides the rectangle centered on the origin of a given reference space in which the user can freely move.
Enumerate swapchain image formats	An application can query the swapchain image formats supported by an XR System.	`swapchainImageFormatIdentifier`	enumeration	Provides an identifier of a swapchain image format that the XR System supports.
Enumerate swapchain images	An application can list the swapchain images allocated to a swapchain.	`numberSwapchainImages`	enumeration	Provides the number of images allocated for a given swapchain.
		`swapchainImages`	object	Provide the implementation-specific swapchain image objects for a given swapchain.

Enumerate composition layer type	An application can list the composition layer types supported by an XR System.	`compositionLayerProjection`	['`projection`', '`quad`', '`cylinder`', '`cube`', '`equirectangular`', '`depth`']	Indicates the type of composition layers supported by the XR Systems supports. The value '`projection`' represents planar projected images, one rendered for each eye using a perspective projection. The value '`quad`' represents quad composition layers which are useful for rendering user interface elements or 2D content on a planar area in the world. The value '`cylinder`' represents cylinder composition layers which maps the texture onto the inside of a cylinder section. The value '`cube`' represents cube composition layer which consists of a cube map with six views to be rendered by the application. The value '`equirectangular`' represents equirectangular composition layers which consists of an equirectangular image that is mapped onto the inside of a sphere in the world. The value '`depth`' represents depth composition layers which allows submitting depth maps as an extra composition layer to be used by the XR Runtime for pose correction.

7 Visual functions and capabilities

7.1 Decoding capabilities

7.1.1 Single decoder instance

The following video decoding capabilities are defined:

- **AVC-FullHD-Dec**: the capability to decode H.264 (AVC) Progressive High Profile Level 4.0 [7] bitstreams.
- **AVC-UHD-Dec:** the capability to decode H.264 (AVC) Progressive High Profile Level 5.1 [7] bitstreams with the following additional requirements:
  - the maximum VCL Bit Rate is constrained to be 120 Mbps with cpbBrVclFactor and cpbBrNalFactor being fixed to be 1250 and 1500, respectively; and,
  - the bitstream does not contain more than 10 slices per picture.
- **AVC-8K-Dec:** the capability to decode H.264 (AVC) Progressive High Profile Level 6.1 [7] bitstreams with the following requirements:
  - the maximum VCL Bit Rate is constrained to be 120 Mbps with cpbBrVclFactor and cpbBrNalFactor being fixed to be 1250 and 1500, respectively; and,
  - the bitstream does not contain more than 16 slices per picture.
  - the bitstream shall not include horizontal motion vector component values that exceed the range from −2048 to 2047, inclusive, or that have vertical motion vector component values that exceed the range from −512 to 511, inclusive, in units of ¼ luma sample displacement. This constraint should be indicated by using values of `log2_max_mv_length_horizontal` less than or equal to 11 and values of `log2_max_mv_length_vertical` less than or equal to 9.
- **HEVC-FullHD-Dec**: the capability to decode H.265 (HEVC) Main10 Profile, Main Tier, Level 4.1 [8] bitstreams that have general_progressive_source_flag equal to 1, general interlaced_source_flag equal to 0, general_non_packed_constraint_flag equal to 1 and general_frame_only_constraint_flag equal to 1.
- **HEVC-UHD-Dec**: the capability to decode H.265 (HEVC) Main10 Profile, Main Tier, Level 5.1 [8] bitstreams that have general_progressive_source_flag equal to 1, general interlaced_source_flag equal to 0, general_non_packed_constraint_flag equal to 1 and general_frame_only_constraint_flag equal to 1.
- **HEVC-8K-Dec**: the capability to decode H.265 (HEVC) Main10 Profile, Main Tier, Level 6.1 [8] bitstreams that have general_progressive_source_flag equal to 1, general interlaced_source_flag equal to 0, general_non_packed_constraint_flag equal to 1, and general_frame_only_constraint_flag equal to 1 with the following additional requirements:
  - the bitstream does not exceed the maximum luma picture size in samples of 33,554,432; and,
  - the maximum VCL Bit Rate is constrained to be 80 Mbps with CpbVclFactor and CpbNalFactor being fixed to be 1000 and 1100, respectively.

7.1.2 Concurrent decoding capabilities

7.1.2.1 Definition

Concurrent video decoder instances are defined as follows.

For *N* video bitstreams encoded according to a video codec profile, decoding units flow into the coded picture buffer (CPB) for each stream according to a specified arrival schedule and are delivered by the common Hypothetical Stream Scheduler (HSS) that schedules the *N* bitstreams for decoding each of the units. For each access unit

- all data associated with an access unit is removed and decoded instantaneously by the instantaneous decoding process at CPB removal time of the access unit.
- Each decoded picture is placed in the Decoded Picture Buffer (DPB) for being referenced by the decoding process of this stream as well as for output and cropping.
- A decoded picture is removed from the DPB at the time that it becomes no longer needed for inter-prediction reference as well as the output time of the access unit is the largest of all decoded pictures remaining in the group of N decoders

Then at any point time,

- each of the individual streams conforms to the signaled profile/level/tier and HRD parameters of the individual stream.
- The sum of the CPB size conforms to common profile/level/tier signaling
- The aggregate decoder processing speed (samples per seconds) conforms to common profile/level/tier signaling.
- The sum of the DPB size conforms to common profile/level/tier signaling
- The common DPB size conforms to common profile/level/tier signaling

A set of *N* concurrent decoder instances conforms to a given capabilities (defined in clause 7.1.2.2), if a set of up to *N* bitstreams encoded to be decodable by the HRD above, is decodable within the timing limits.

7.1.2.2 Capabilities

Based on the definition in clause 7.1.2.1, the following capabilities are defined:

- **AVC-FullHD-Dec-2**: The capability of supporting up to two (*N*=2) concurrent decoder instances with the aggregate capabilities of *AVC-FullHD-Dec*.
- **AVC-UHD-Dec-4**: The capability of supporting up to four (*N*=4) concurrent decoder instances with the aggregate capabilities of *AVC-UHD-Dec*.
- **HEVC-UHD-Dec-4:** The capability of supporting up to four (*N*=4) concurrent decoder instances with the aggregate capabilities of *HEVC-UHD-Dec*.
- **UHD-Dec-4**: The capability supporting up to four (*N*=4) concurrent decoder instances with either:
  - the aggregate capabilities of *AVC-UHD-Dec-4*;
  - the aggregate capabilities of *HEVC-UHD-Dec-4*; or,
  - the capability of decoding up to 4 bitstreams for which each bitstream does not exceed the capability of being decodable either with *AVC-FullHD-Dec* or *HEVC-FullHD-Dec*.
- **AVC-8K-Dec-8:** The capability of supporting up to eight (*N*=8) concurrent decoder instances with the aggregate capabilities of *AVC-8K-Dec*.
- **HEVC-8K-Dec-8:** The capability of supporting up to eight (*N*=8) concurrent decoder instances with the aggregate capabilities of *HEVC-8K-Dec*.
- **8K-Dec-8**: The capability supporting up to eight (*N*=8) concurrent decoder instances with either:
  - the aggregate capabilities of *AVC-8K-Dec-8*;
  - the aggregate capabilities of *HEVC-8K-Dec-8*; or,
  - the capability of decoding up to:
    - eight bitstreams for which each bitstream does not exceed the capability of being decodable either with *AVC-FullHD-Dec* or *HEVC-FullHD-Dec*; or,
    - four bitstreams for which each bitstream does not exceed the capability of being decodable either with *AVC-UHD-Dec* or *HEVC-UHD-Dec*.

7.2 Encoding capabilities

7.2.1 Single encoder instance

The following video encoding capabilities are defined:

- **AVC-FullHD-Enc:** the capability to encode a video signal to a bitstream that is decodable by a decoder that is *AVC-FullHD-Dec* capable as defined in clause 7.1.1.1 with the following additional constraints:
  - up to 245,760 macroblocks per second;
  - up to a frame size of 8,192 macroblocks;
  - up to 240 frames per second;
  - the chroma format being 4:2:0; and
  - the bit depth being 8 bit;
- **HEVC-FullHD-Enc:** the capability to encode a video signal to a bitstream that is decodable by a decoder that is *HEVC-FullHD-Dec* capable as defined in clause 7.1.1 with the following additional constraints:
  - up to 133,693,440 luma samples per second;
  - up to a luma picture size of 2,228,224 samples;
  - up to 240 frames per second;
  - the Chroma format being 4:2:0; and
  - the bit depth being either 8 or 10 bit;
- **HEVC-UHD-Enc:** the capability to encode a video signal to a bitstream that is decodable by a decoder that is *HEVC-UHD-Dec* capable as defined in clause 7.1.1 with the following additional constraints:
  - up to 534,773,760 luma samples per second;
  - up to a luma picture size of 8,912,896 samples;
  - up to 480 frames per second;
  - the Chroma format being 4:2:0; and
  - the bit depth being either 8 or 10 bit;

7.3 Capability exchange

The capabilities of the device are captured in the data structure defined in 12.5.

8 Audio functions and capabilities

8.1 Audio/Speech Decoding

The following audio/speech decoding capabilities are defined:

- **EVS**: the decoding capability as defined in TS 26.117 [9] clause 5.2.
- **IVAS**: the decoding capability as defined in TS 26.117 [9] clause 5.2.

- **EVS-2**: the decoding capability as defined TS 26.117 [9] clause 5.2.
- **EVS-4**: the decoding capability as defined TS 26.117 [9] clause 5.2.
- **AAC-ELDv2-Dec**: the decoding capability as defined TS 26.117 [9] clause 5.2.
- **AAC-ELDv2-Dec-2**: the decoding capability as defined TS 26.117 [9] clause 5.2.

8.2 Audio/Speech Encoding

The following audio/speech encoding capabilities are defined:

- **EVS**: the sender requirements for the **EVS** Operation Point as defined in TS 26.117 [9] clause 6.2.4.3.
- **IVAS**: the sender requirements for the **IVAS** Operation Point as defined in TS 26.117 [9] clause 6.3.5.3.
- **AAC-ELDv2-Enc:** the sender requirements for the **AAC-ELDv2** Operation Point as defined in TS 26.117 [9] clause 6.3.6.3.

9 Scene processing capabilities

9.1 General

This clause defines scene processing capabilities that enable AR and XR experiences.

In clause 9.2, glTF-based Scene Description capabilities are defined.

NOTE: Functionalities of this specification can be used w/o a standardized scene description, for example by an application providing an interoperability on Scene Description. However, in this case, AR experiences may be limited to the functionalities of the application and may be restricted.

9.2 glTF-based Scene Description capabilities

This clause defines client cabapilities for rendering and presenting Scene Description based on glTF2.0 [10]. Additional extended capabilites are defined, primarily to support real-time media based on the extensions defined in the MPEG-I Scene Description in ISO/IEC 23090-14 [11-13].

The following scene processing capabilities are defined.

- **SD-Rendering-glTF-Core:** The capability to process glTF2.0 [10] scene description files and to render the described scenes with the following restrictions:
  - The glTF 2.0 scene description can either be a standalone JSON file or an encapsulated GLB file.
  - The glTF 2.0 scene description can have an aggregate geometry complexity, including all mesh primitives of the scene, of at least 100k faces and/or points. The requirement on the number of faces in mesh nodes allows for moderate complexity scenes.
  - The glTF 2.0 scene description can have at least 20 materials using static image textures. The requirement on the number of materials with image texture maps allows for moderate complexity scenes.
  - The glTF 2.0 scene description can use glTF2.0 animations and skinning.
  - The glTF 2.0 scene description can use the `KHR_lights_punctual` and `KHR_materials_specular` extensions.
- **SD-Rendering-glTF-Ext1**: On top of the **SD-Rendering-glTF-Core** capability, the capability to process MPEG-I Scene Description documents as specified in [11-13] and to render the described scenes with the following restrictions:

- The MPEG-I Scene Description document can use the `MPEG_media` extension.
- The MPEG-I Scene Description document can use the `MPEG_accessor_timed` and the MPEG_buffer_circular extensions.
- The MPEG-I Scene Description document can use the `MPEG_texture_video` extension.
- The MPEG-I Scene Description document can use the `MPEG_audio_spatial` extension.
- The MPEG-I Scene Description document can use the scene description update mechanism as defined in clause 5.2.4 of [9].
- Processing of scene description updates as defined in ISO/IEC 23090-14 [11-13] to allow the device to contribute to a scene description of a dynamic shared scene.
- The MPEG-I Scene Description document can use at least 4 materials using video textures

NOTE: This functionality implies that at least 4 concurrent video decoder instances are available

- The MPEG-I Scene Description document may use at least 4 object-based mono audio sources, which allows for relatively simple audio support.

NOTE: This functionality implies that at least 4 concurrent mono sources are available as the output of the audio decoding engine instances are available, potentially requiring concurrent audio decoder instances

Editor's Note: the audio-specific minimal support, such as the type of audio sources and the number of audio sources to support as part of this capability, are to be agreed with the Audio SWG.

- **SD-Rendering-glTF-Ext2:** On top of the **SD-Rendering-glTF-Core** capability, the capability to process MPEG-I Scene Description documents as specified in [11-13] with the following restrictions:
  - The MPEG-I Scene Description document can use the `MPEG_anchor` extension.
  - The MPEG-I Scene Description document can use the `EXT_lights_image_based` and the `MPEG_lights_texture_based` extensions.
- **SD-Rendering-glTF-Interactive**: On top of the **SD-Rendering-glTF-Core** capability, the capability to process the following glTF 2.0 extensions defined in [11-13] to enable user input and local interaction processing.
  - The MPEG-I Scene Description document can use the `MPEG_scene_interactivity` extension with the following triggers: TRIGGER_USER_INPUT, TRIGGER_PROXIMITY, TRIGGER_COLLISION, TRIGGER_VISIBILITY.
  - The MPEG-I Scene Description document can use the `MPEG_scene_interactivity` extension with actions: ACTION_ACTIVATE, ACTION_TRANSFORM, ACTION_BLOCK, ACTION_ANIMATION, ACTION_MEDIA, ACTION_MANIPULATE, ACTION_SET_MATERIAL

10 Device types and media profiles

10.1 Introduction

AR experiences may be running on a variety of devices which have different characteristics and capabilities. Certain capabilities may be common to several devices while other capabilities may be unique to a specific device. Therefore, the present specification enables interoperability by collecting the media capabilities and profiles, defined in clauses 6, 7, 8 and 9, per device type. The four device types defined are:

- Device type 1: Thin AR glasses (see clause 10.2)
- Device type 2: AR glasses (see clause 10.3)
- Device type 3: XR phone (see clause 10.4)

- Device type 4: XR HMD (see clause 10.5)

NOTE: A given physical device may be compliant to more than one device types.

10.2 Device type 1: Thin AR glasses

10.2.1 General

As defined in 4.3.1, the thin AR glasses device type represents a type of device which is considered to be power-constrained and with limited computing power with respect to the other device types. These limitations typically come from the requirement to design a device with a small and lightweight form factor. Regarding rendering capacity, this device type is expected to rely on remote rendering to be able display complex scenes to the user. For example, such device type may run a split rendering session where the split rendering server delivers pre-rendered views of the scene. However, devices in this category can still operate without external support for applications that do not require complex rendering capabilities, for instance, text messaging, 2D video communication, etc. Lastly, the thin AR glasses offers AR experiences to the user via optical see-through display.

10.2.2 XR System support

An XR Device complying to the thin AR glasses device has an XR System with at least the following capabilities as based on clause 6.1:

- `orientationTracking` is '`true`'
- `positionTracking` is '`true`'
- Value '`additive`' of the enumeration `blendMode`
- Values '`monoscopic`' and '`stereoscopic`' of the enumeration `viewConfigurationPrimary`
- Values '`view`', '`local`' and '`stage`' of the enumeration `referenceSpace`
- If `swapchainSupported` is '`true`', `numberSwapchainImages` is equal to 2
- Values '`projection`' and '`quad`' of the enumeration `compositionLayer`

NOTE: For the definition of those capabilities, please refer to clause 4.1.3.

10.2.3 Video capabilities support

An XR Device complying to device type 1 shall support the following decoding capabilities from clause 7:

- **AVC-FullHD-Dec**
- **AVC-FullHD-Dec-2**

An XR Device complying to device type 1 shall support the following encoding capabilities:

- **AVC-FullHD-Enc**

An XR Device complying to device type 1 should support the following decoding capabilities:

- **HEVC-FullHD-Dec**

An XR Device complying to device type 1 should support the following encoding capabilities:

- **HEVC-FullHD-Enc**

10.2.4 Audio/Speech capabilities support

An XR Device complying to device type 1 device shall support the following decoding capabilities from clause 8:

- **EVS**
- **AAC-ELDv2-Dec**

An XR Device complying to device type 1 should support the following decoding capabilities:

- **IVAS-SR**
- **EVS**

An XR Device complying to device type 1 may support the following decoding capabilities:

- **IVAS-L1**
- **EVS-4**
- **AAC-ELDv2-Dec-2**

An XR Device complying to device type 1 shall support the following encoding capabilities:

- **EVS**

An XR Device complying to device type 1 should support the following encoding capabilities:

- **IVAS-L1**
- **AAC-ELDv2-Enc**

10.2.5 Scene Description capabilities support

A device of type 1 should support glTF-based scene description as defined in clause 9.2.

If gltf-based scene description is supported, the following requirements and recommendation hold:

- The **SD-Rendering-gltf-Core** capabilities should be supported.

10.3 Device type 2: AR glasses

10.3.1 General

As defined in 4.3.2, the AR glasses device type represents a type of device which is considered to have higher computation power compared to the thin AR glasses device type. As a result, this device type has higher rendering capacities and is generally expected to be capable of rendering scenes without external support, even though remote rendering is not precluded to lower the power consumption on the device or enable the display of scenes beyond the device's rendering capability. Lastly, the AR glasses offers AR experiences to the user via optical see-through display.

10.3.2 XR System support

An XR Device complying to the AR glasses device type has offers XR System with at least the following capabilities from clause 6.1:

- `orientationTracking` is '`true`'
- `positionTracking` is '`true`'
- Value '`additive`' of the enumeration `blendMode`
- Value '`stereoscopic`' of the enumeration `viewConfigurationPrimary`
- Values '`view`', '`local`' and '`stage`' of the enumeration `referenceSpace`
- If `swapchainSupported` is '`true`', `numberSwapchainImages` is equal to 2

- Values '`projection`' and '`quad`' of the enumeration `compositionLayer`

NOTE: For the definition of those capabilities, please refer to clause 4.1.3.

10.3.3 Video capabilities support

An XR Device complying to device type 2 shall support the following decoding capabilities from clause 7:

- **AVC-UHD-Dec**
- **AVC-UHD-Dec-4**
- **HEVC-UHD-Dec**

An XR Device complying to device type 2 shall support one of the following encoding capabilities:

- **AVC-FullHD-Enc**
- **HEVC-FullHD-Enc**

An XR Device complying to device type 2 should support the following decoding capabilities:

- **HEVC-UHD-Dec-4**
- **UHD-Dec-4**

10.3.4 Audio/Speech capabilities support

An XR Device complying to device type 2 device shall support the following decoding capabilities from clause 8:

- **EVS**
- **AAC-ELDv2-Dec**

An XR An XR Device complying to device type 2 device shall support the following decoding capabilities from clause 8:

- **EVS**
- **AAC-ELDv2-Dec**

An XR Device complying to device type 2 should support the following decoding capabilities:

- **IVAS-SR**
- **IVAS-L1**
- **EVS-2**

An XR Device complying to device type 2 may support the following decoding capabilities:

- **EVS-4**
- **AAC-ELDv2-Dec-2**

An XR Device complying to device type 2 shall support the following encoding capabilities:

- **EVS**

An XR Device complying to device type 2 should support the following encoding capabilities:

- **IVAS-L1**
- **AAC-ELDv2-Enc**

10.3.5 Scene Description capabilities support

A device of type 2 should support glTF-based scene description as defined in clause 9.2.

If gltf-based scene description is supported, the following requirements and recommendation hold.

- The **SD-Rendering-gltf-Core** capabilities shall be supported
- The **SD-Rendering-gltf-ext1** capabilities should be supported
- The **SD-Rendering-gltf-ext2** capabilities may be supported
- The **SD-Rendering-gltf-interactive** capabilities may be supported

10.4 Device type 3: XR phone

10.4.1 General

As defined in 4.3.1, the XR phone device type represents a type of device which corresponds to a smartphone with capacities and resources sufficient to offer AR experiences. As a result, this device type is capable of rendering scenes without external support. Lastly, the XR phone offers AR experiences to the user via video see-through display.

10.4.2 XR System support

An XR Device complying to the XR phone device type offers an XR System with at least the following capabilities:

- `orientationTracking` is '`true`'
- `positionTracking` is '`true`'
- Values '`opaque`' and '`alpha_blend`' of the enumeration `blendMode`
- Values '`monoscopic`' and '`stereoscopic`' of the enumeration `viewConfigurationPrimary`
- Values '`view`', '`local`' and '`stage`' of the enumeration `referenceSpace`
- If `swapchainSupported` is '`true`', `numberSwapchainImages` equal to 2
- Values '`projection`' and '`quad`' of the enumeration `compositionLayer`

NOTE: For the definition of those capabilities, please refer to clause 6.1.

10.4.3 Video capabilities support

An XR Device complying to device type 3 shall support the following decoding capabilities from clause 7:

- **AVC-UHD-Dec**
- **AVC-UHD-Dec-4**
- **HEVC-UHD-Dec**
- **HEVC-UHD-Dec-4**
- **UHD-Dec-4**

An XR Device complying to device type 3 shall support one of the following encoding capabilities:

- **AVC-UHD-Enc**
- **HEVC-UHD-Enc**

An XR Device complying to device type 3 should support the following decoding capabilities:

- **AVC-8K-Dec**
- **HEVC-8K-Dec**
- **8K-Dec-8**

10.4.4 Audio/Speech capabilities support

An XR Device complying to device type 3 device shall support the following decoding capabilities from clause 8:

- **EVS**
- **AAC-ELDv2-Dec**

An XR Device complying to device type 3 should support the following decoding capabilities:

- **IVAS**
- **EVS-2**

An XR Device complying to device type 3 may support the following decoding capabilities:

- **EVS-4**
- **AAC-ELDv2-Dec-2**

An XR Device complying to device type 3 shall support the following encoding capabilities:

- **EVS**

An XR Device complying to device type 3 should support the following encoding capabilities:

- **IVAS**
- **AAC-ELDv2-Enc**

10.4.5 Scene Description capabilities support

A device of type 3 should support gltf-based scene description as defined in clause 9.2.

If gltf-based scene description is supported, the following requirements and recommendation hold.

- The **SD-Rendering-gltf-Core** capabilities shall be supported
- The **SD-Rendering-gltf-ext1** capabilities should be supported
- The **SD-Rendering-gltf-ext2** capabilities should be supported
- The **SD-Rendering-gltf-interactive** capabilities should be supported

10.5 Device type 4: XR HMD

10.5.1 General

As defined in 4.3.1, the XR HMD device type represents a type of device which corresponds to HMDs capable of offering at least AR experiences but not precluding other types of XR experiences. This device type is expected to be capable of rendering scenes without external support. Lastly, the XR phone offers AR experiences to the user via video see-through display.

10.5.2 XR System support

An XR Device complying to the XR HMD device type offers an XR System with at least the following capabilities from clause 6.1:

- `orientationTracking` is '`true`'
- `positionTracking` is '`true`'
- Value '`additive`' or values '`opaque`' and '`alpha_blend`' of the enumeration `blendMode`
- Values '`monoscopic`' and '`stereoscopic`' of the enumeration `viewConfigurationPrimary`
- Values '`view`', '`local`' and '`stage`' of the enumeration `referenceSpace`
- If `swapchainSupported` is '`true`', `numberSwapchainImages` is equal to 2
- Values '`projection`' and '`quad`' of the enumeration `compositionLayer`

NOTE: For the definition of those capabilities, please refer to clause 6.1.

10.5.3 Video capabilities support

An XR Device complying to device type 4 shall support the following decoding capabilities from clause 7:

- **AVC-UHD-Dec**
- **AVC-UHD-Dec-4**
- **HEVC-UHD-Dec**
- **HEVC-UHD-Dec-4**
- **UHD-Dec-4**

An XR Device complying to device type 4 shall support one of the following encoding capabilities:

- **AVC-UHD-Enc**
- **HEVC-UHD-Enc**

An XR Device complying to device type 4 should support the following decoding capabilities:

- **AVC-8K-Dec**
- **HEVC-8K-Dec**
- **8K-Dec-8**

10.5.4 Audio/Speech capabilities support

An XR Device complying to device type 4 device shall support the following decoding capabilities from clause 8:

- **EVS**
- **AAC-ELDv2-Dec**

An XR Device complying to device type 4 should support the following decoding capabilities:

- **IVAS**

- **EVS-2**

An XR Device complying to device type 4 may support the following decoding capabilities:

- **EVS-4**
- **AAC-ELDv2-Dec-2**

An XR Device complying to device type 4 shall support the following encoding capabilities:

- **EVS**

An XR Device complying to device type 4 should support the following encoding capabilities:

- **IVAS**
- **AAC-ELDv2-Enc**

10.5.5 Scene Description capabilities support

A device of type 4 should support gltf-based scene description as defined in clause 9.2.

If gltf-based scene description is supported, the following requirements and recommendation hold.

- The **SD-Rendering-gltf-Core** capabilities shall be supported
- The **SD-Rendering-gltf-ext1** capabilities should be supported
- The **SD-Rendering-gltf-ext2** capabilities should be supported
- The **SD-Rendering-gltf-interactive** capabilities should be supported

11 QoE metrics

11.1 Metrics and Observation Points

11.1.1 Overview

The Observation Points (OPs) are defined to support the definition of the corresponding metrics. This specification defines four observation points as shown in Figure 11.1.1-1. The metrics collection function, as part of the Media Session Handler, is responsible of collecting specific information observed at each OP in order to generate the metrics. This function has also access to the 5G System such that the metrics can be reported to an external entity.

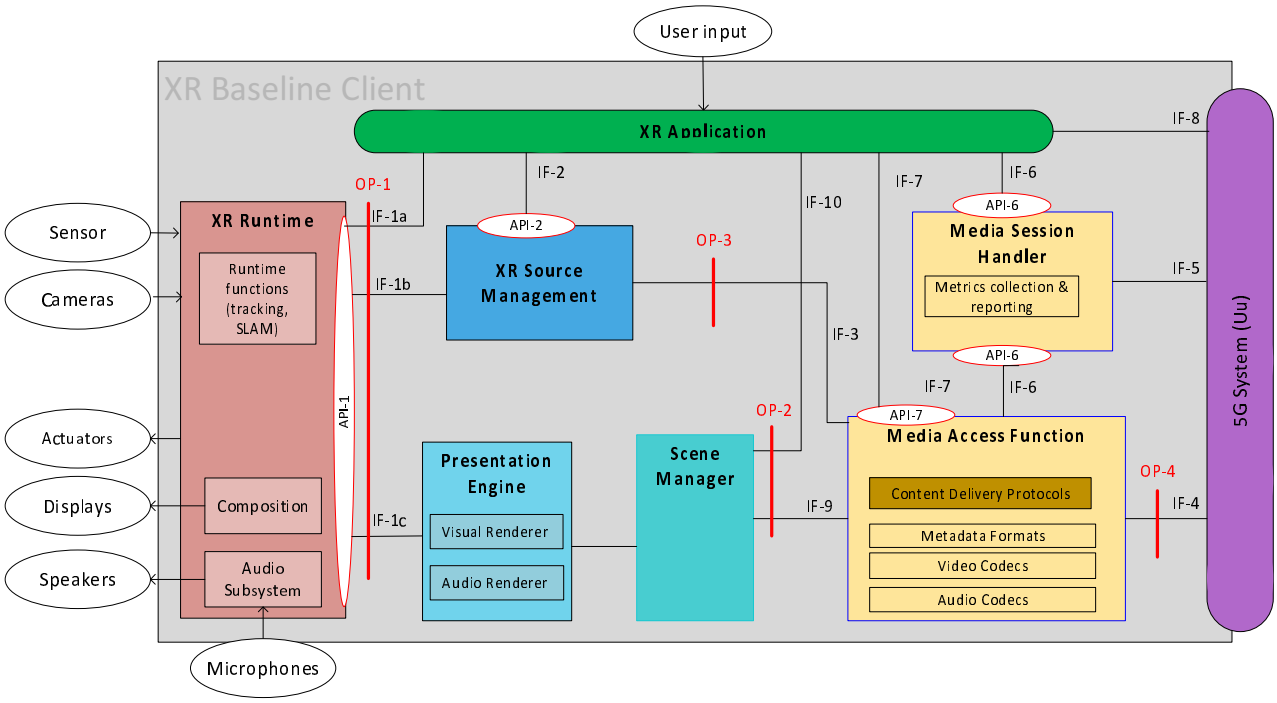

Figure 11.1.1-1 - Observation Points in the XR Baseline Client

11.1.2 Observation Point 1: XR Runtime information

Observation point 1 (OP-1) is derived from the XR Runtime API. The OP-1 observes information exchanged between the XR Runtime on one side and the XR Source Management, the Presentation Engine and the application on the other side, i.e. on IF-1.

On observation point 1, the following observed information is defined:

- XR runtime clock (XR runtime clock)
- Actual presentation/display time
- user input actions and the time when the action is made
- Rendering loop timing information: to observe the start of the scene update by the scene manager and the start of the rendering process.

11.1.3 Observation Point 2

Observation point 2 (OP-2) observes information at the input of the Scene Manager, i.e. on IF-9 for data received from the Media Access Function and the IF-10 for information exchanged between the Scene Manager and the application.

11.1.4 Observation Point 3

Observation point 3 (OP-3) is derived from the API which exchanges information between the XR Source Management and the Media Access Functions. It corresponds to the IF-3 interface.

11.1.5 Observation Point 4

Observation point 4 (OP-4) observes information between the Media Access Function and the 5G System, i.e. on IF-4 interface.

11.2 Metrics Definitions

11.2.1 Latency metrics

To enable good XR experiences, it is relevant to monitor latencies such as the pose-to-render-to-photon.

Beyond the sense of presence and immersiveness, the age of the content and user interaction delay are of the uttermost importance for immersive and non-immersive interactive experiences, i.e. experiences for which the user interaction with the scene impacts the content of scene (such as online gaming).

Table 11.2.1-1 provides time information that may be gathered to compute the latency metrics. The observation points to collect the time information are indicated per device type.

Table 11.2.1-1: Time information for latency metrics

Time information	Definition	Observation Point
estimatedAtTime (ref. T1)	The time when the viewer pose prediction is made. It corresponds to the time when the predicted viewer pose is collected using the XR runtime API-1 by the application or the XR Source Manager. This time is expressed in XR system time clock.	OP-1
lastChangeTime	The time when the user action is made. It corresponds to the lastChangeTime field defined in the action format in Table 5.1.3-1. This time is expressed in XR system time clock.	OP-1
sceneUpdateTime (ref. T6)	The time when the Scene Manager starts to update the 3D scene graph according to the viewer pose and the user actions. This time is expressed in wall clock time	OP-1
startToRenderAtTime (ref. T3)	The time when the renderer starts to render the scene according to the viewer pose. This time is expressed in wall clock time	OP-1
actualDisplayTime (ref. T2.actual)	The actual display time of the rendered frame in the swapchain. The estimation of the actual display time is available through the XR runtime. This time is expressed in XR system time clock.	OP-1

The latency metrics are specified in Table 11.2.1-2. The formula to compute the latencies are defined using the collected time information.

Table 11.2.1-2: Latency metrics

Latency metric	Description
poseToRenderToPhoton	The time duration, in units of milliseconds, between the time to query the pose information from the XR runtime to the renderer (the renderer uses this pose to generate the rendered frame) and the display time of the rendered frame. It can be computed as follows: actualDisplayTime – estimatedAtTime
renderToPhoton	The time duration, in units of milliseconds, between the start of the rendering by the Presentation Engine and the display time of the rendered frame. It can be computed as follows: actualDisplayTime – startToRenderAtTime

roundtripInteractionDelay	The time duration, in units of milliseconds, between the time a user action is initiated and the time the action is presented to the user. It can be computed as follows: actualDisplayTime – lastChangeTime
userInteractionDelay	The time duration, in units of milliseconds, between the time a user action is initiated and the time the action is taken into account by the content creation engine in the scene manager. It can be computed as follows: sceneUpdateTime – lastChangeTime
ageOfContent	The time duration, in units of milliseconds, between the time the content is created in the scene by the Scene Manager and the time it is presented to the user. It can be computed as follows: actualDisplayTime – sceneUpdateTime
sceneUpdateDelay	The time duration, in units of milliseconds, spent by the Scene Manager to update the scene graph. It can be computed as follows: startToRenderAtTime – sceneUpdateTime

NOTE: The metrics renderToPhoton, userInteractionDelay, and ageOfContent are considered being expressed into a single time format (e.g., wall clock time).

12 Metadata formats

12.1 General

Several applications may require the exchange of real-time metadata information for about the XR session. For instance, split rendering applications and immersive communication services may require the UE to share Pose and action information pertaining to the user's current pose, to their input (e.g. pulling a trigger on the XR controller) or trackable pose. This clause defines the metadata formats for timed metadata of an XR session.

12.2 Pose format

The Pose format is used to share pose information, e.g. about predicted poses with the network.

Each predicted pose shall contain the associated predicted display time and an identifier of the XR space that was used for that pose.

Depending on the view configuration of the XR session, there could be different pose information for each view.

The payload of the message shall follow the structure defined in Table 12.2-1.

Table 12.2-1 - Pose format

Name	Type	Cardinality	Description
poseInfo	Object	1..n	An array of pose information objects, each corresponding to a target display time and XR space.

poseTime	number	1..1	The time for which the current poses are predicted. This time is expressed in XR system time clock.
xrSpaceId	number	0..1	An identifier for the XR space in which the poses are expressed.
poses	Object	1..n	An array that provides a list of the poses. For view poses, the first pose corresponds to the left view and the second to the right view.
trackableSpaceId	number	0..1	A unique identifier of the XR space **of the trackable** that was agreed upon during session setup. The pose corresponds to the origin of that trackableSpaceId expressed in the XR space identified by xrSpaceId. This is only applicable for trackable pose.
orientation	Object	1..1	Represents the orientation of the pose as a quaternion based on the reference XR space identified by xrSpaceId.
x	number	1..1	Provides the x coordinate of the quaternion.
y	number	1..1	Provides the y coordinate of the quaternion.
z	number	1..1	Provides the z coordinate of the quaternion.
w	number	1..1	Provides the w coordinate of the quaternion.
position	Object	0..1	Represents the position of the pose relative to the XR space identified by xrSpaceId.
x	number	1..1	Provides the x coordinate of the position vector.
y	number	1..1	Provides the y coordinate of the position vector.
z	number	1..1	Provides the z coordinate of the position vector.
confidence	number	0..1	Provides a confidence score that reflects the probability for this pose prediction to be correct. For the current pose or a pose in the past, the confidence value would be 1. The confidence can take a value between 0 and 1.
estimatedAtTime	number	0..1	The wall clock time when the pose estimation was made. (ref. T1)
fov	Object	0..1	Indicates the four sides of the field of view used for the projection of the corresponding XR view. This field is only present if these field of view values have changed from the last sent values.

			This is only applicable for view poses
angleLeft	number	1..1	The angle in radians of the left side of the field of view. For a symmetric field of view this value is negative.
angleRight	number	1..1	The angle in radians of the right side of the field of view.
angleUp	number	1..1	The angle in radians of the top part of the field of view.
angleDown	number	1..1	The angle in radians of the bottom part of the field of view. For a symmetric field of view this value is negative.

12.3 Action format

Actions are grouped into action sets which may be activated and deactivated during the lifetime of an XR session. The action sets and actions are negotiated at the start of the split rendering session.

The content of the action message type shall the structure defined in Table 12.3-1.

Table 12.3-1 - Action format

Name	Type	Cardinality	Description
actionSets	Object	1..n	An array of active action sets, for which there is at least an action that has a state change.
actions	number	1..n	An array of objects that conveys information about the actions of the parent action set.
identifier	string	1..1	A unique identifier of the action.
subactionPath	string	1..1	The sub-action path for which the state has changed. It abstracts a binding between an action and the hardware input associated to it by the XR runtime.
state	object	1..1	The state of the action that had a change in state.
lastChangeTime	number	1..1	The timestamp of the last change to the state of this action.
currentStateBool	Bool	0..1	The current Boolean state of the action
currentStateNum	number	0..1	The current numerical state of the action.
currentStateVec2	Array	0..1	An array of numerical state values for the action.

12.4 Available Visualization Space format

The XR Application may define a three-dimensional space within the user's real-word space that is suitable for rendering virtual objects called the Available Visualization Space. Such a space is defined with a shape which is either cube or sphere with the corresponding size and coordinates. In the case that the virtual scene is rendered by a remote entity (e.g. split rendering), this Available Visualization Space may be transmitted to this remote entity so that the

composed AR objects remain within the defined Available Visualization Space. The method of calculating the Available Visualization Space is out of the scope of this document.

The content of the availableVisualizationSpace type shall follow the format defined in Table 12.4-1.

Table 12.4-1 – Available Visualization Space

Name	Type	Cardinality	Description
availableVisualizationSpace	Object	0..1	An object defining the coordinate of the available visualization space.
xrSpaceId	number	0..1	An identifier for the XR space in which the available visualization space is expressed.
cuboid	Object	0..1*	The available visualization space in form of cuboid. The 3D coordinates are expressed in the XR Space identified by xrSpaceId.
x	float	1	Offset of the available visualization space starting point in the x direction. The value is in meters.
y	float	1	Offset of the available visualization space starting point in the y direction as defined by the Open XR coordinate system. The value is in meters.
z	float	1	Offset of the available visualization space starting point in the z direction. The value is in meters.
width	float	1	The width of available visualization space in the x direction as defined by the Open XR coordinate system. The value is in meters.
height	float	1	The height of available visualization space in the y direction. The value is in meters.
depth	float	1	The depth of available visualization space in the z direction. The value is in meters.
sphere	Object	0..1*	The available visualization space in form of a sphere. The 3D coordinates are expressed in the XR Space identified by xrSpaceId
x	float	1	Offset of the available visualization space center in the x direction as defined by the Open XR coordinate system. The value is in meters.
y	float	1	Offset of the available visualization space center in the y direction. The value is in meters.

z	float	1	Offset of the available visualization space center in the z direction. The value is in meters.
radius	float	1	The radius of available visualization space. The value is in meters.
*Only one of cuboid or sphere object shall exists.			

With this Available Visualization Space, a user may for instance avoid the virtual objects to be occluding other real objects in the scene, e.g. TV sets, people, etc., since an AR experience is achieved by the integration of visual objects into the user environment.

12.5 Device capabilities signalling

Device capabilities may be signalled using the format defined in Table 12.5-1.

Table 12.5-1 – Device capabilities exchange format

Name	Type	Cardinality	Description
deviceCapabilities	Object	0..1	Provides the supported device capabilities.
deviceType	string	1...N	A list of device type identifiers formatted as URN defined in table A-1 in Annex A. For each signalled device type identifier, the associated capabilities are supported by the sending device.
additionalCapabilities	string	0...N	A list of additional media capability identifiers formatted as URN defined in table A-2 in Annex A For each signalled media capability identifier, the associated capabilities are supported by the sending device.

Annex A (informative): Registration Information

A.1 3GPP Registered URIs

The clause documents the registered URIs in this specification following the process in https://www.3gpp.org/3gpp-groups/core-network-terminals-ct/ct-wg1/uniform-resource-identifier-uri-list

Table A-1 lists all registered URN values for deice type identifiers as well as

- a brief description of its functionality;
- a reference to the specification or other publicly available document (if any) containing the definition;
- the name and email address of the person making the application; and
- any supplementary information considered necessary to support the application.

Table A-1: 3GPP Registered URNs for device type identifiers

URN	Description	Reference	Contact	Remarks
`urn:3GPP:26119:18:device-type-1`	An identifier for the device type 1 defined in this specification.	TS 26.119, clause 4.3.1	Emmanuel Thomas thomase@xiaomi.com	none
`urn:3GPP:26119:18:device-type-2`	An identifier for the device type 2 defined in this specification.	TS 26.119, clause 4.3.2	Emmanuel Thomas thomase@xiaomi.com	none
`urn:3GPP:26119:18:device-type-3`	An identifier for the device type 3 defined in this specification.	TS 26.119, clause 4.3.3	Emmanuel Thomas thomase@xiaomi.com	none
`urn:3GPP:26119:18:device-type-4`	An identifier for the device type 4 defined in this specification.	TS 26.119, clause 4.3.4	Emmanuel Thomas thomase@xiaomi.com	none

Table A-2 lists all registered URN values for video capability identifiers as well as

- a brief description of its functionality;
- a reference to the specification or other publicly available document (if any) containing the definition;
- the name and email address of the person making the application; and
- any supplementary information considered necessary to support the application.

Table A-1: 3GPP Registered URNs for video capability identifiers

URN	Description	Reference	Contact	Remarks
urn:3GPP:26119:18:AVC-FullHD-Dec	An identifier for the capability **AVC-FullHD-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-UHD-Dec	An identifier for the capability **AVC-UHD-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-8K-Dec	An identifier for the capability **AVC-8K-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-FullHD-Dec	An identifier for the capability **HEVC-FullHD-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-UHD-Dec	An identifier for the capability **AVC-UHD-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-UHD-Dec	An identifier for the capability **HEVC-UHD-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-8K-Dec	An identifier for the capability **HEVC-8K-Dec** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-FullHD-Dec-2	An identifier for the capability **AVC-FullHD-Dec-2** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-UHD-Dec-4	An identifier for the capability **AVC-UHD-Dec-4** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-UHD-Dec-4	An identifier for the capability **HEVC-UHD-Dec-4** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:UHD-Dec-4	An identifier for the capability **UHD-Dec-4** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-8K-Dec-8	An identifier for the capability **AVC-8K-Dec-8** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-8K-Dec-8	An identifier for the capability **HEVC-8K-Dec-8** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:8K-Dec-8	An identifier for the capability **8K-Dec-8** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AVC-FullHD-Enc	An identifier for the capability **AVC-FullHD-Enc** defined this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-FullHD-Enc	An identifier for the capability **HEVC-FullHD-Enc** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:HEVC-UHD-Enc	An identifier for the capability **HEVC-UHD-Enc** defined in this specification.	TS 26.119, clause 7.1	Emmanuel Thomas thomase@xiaomi.com	none

Table A-3 lists all registered URN values for audio capability identifiers as well as

- a brief description of its functionality,
- a reference to the specification or other publicly available document (if any) containing the definition,
- the name and email address of the person making the application, and

- any supplementary information considered necessary to support the application.

Table A-3: 3GPP Registered URNs for audio capability identifiers

URN	Description	Reference	Contact	Remarks
urn:3GPP:26119:18:EVS:Dec	An identifier for the capability **EVS** defined in this specification.	TS 26.119, clause 8:1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:IVAS:Dec	An identifier for the capability **IVAS** defined in this specification.	TS 26.119, clause 8:1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:EVS-2:Dec	An identifier for the capability **EVS-2** defined in this specification.	TS 26.119, clause 8:1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:EVS-4:DEC	An identifier for the capability **EVS-4** defined in this specification.	TS 26.119, clause 8:1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AAC-ELDv2-Dec	An identifier for the capability **AAC-ELDv2-Dec** defined in this specification.	TS 26.119, clause 8:1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AAC-ELDv2-Dec-2	An identifier for the capability **AAC-ELDv2-Dec-2** defined in this specification.	TS 26.119, clause 8:1	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:EVS-Enc	An identifier for the capability **EVS** defined in this specification.	TS 26.119, clause 8:2	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:IVAS:Enc	An identifier for the capability **IVAS** defined in this specification.	TS 26.119, clause 8:2	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:AAC-ELDv2-Enc	An identifier for the capability **AAC-ELDv2-Enc** defined in this specification.	TS 26.119, clause 8:2	Emmanuel Thomas thomase@xiaomi.com	none

Table A-4 lists all registered URN values for scene processing capability identifiers as well as

- a brief description of its functionality,
- a reference to the specification or other publicly available document (if any) containing the definition,
- the name and email address of the person making the application; and
- any supplementary information considered necessary to support the application.

Table A-4: 3GPP Registered URNs for scene processing capability identifiers

URN	Description	Reference	Contact	Remarks
urn:3GPP:26119:18:SD-Rendering-glTF-Core	An identifier for the capability **SD-Rendering-glTF-Core** defined in this specification.	TS 26.119, clause 9.2	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:SD-Rendering-glTF-Ext1	An identifier for the capability **SD-Rendering-glTF-Ext1** defined tin his specification.	TS 26.119, clause 9.2	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:SD-Rendering-glTF-Ext2	An identifier for the capability **SD-Rendering-glTF-Ext2** defined in this specification.	TS 26.119, clause 9.2	Emmanuel Thomas thomase@xiaomi.com	none
urn:3GPP:26119:18:SD-Rendering-glTF-Interactive	An identifier for the capability **SD-Rendering-glTF-Interactive** defined in this specification.	TS 26.119, clause 9.2	Emmanuel Thomas thomase@xiaomi.com	none

Annex B (informative): XR Runtime interface

B.1 Introduction

This annex describes the XR Runtime functions to be used with the 3GPP capabilities defined in the presented document. Clause B.2.2 focused the mapping of the 3GPP capabilities with the OpenXR runtime. Clause B.2.2 extracted relevant information from the OpenXR specification [5] regarding the rendering operations.

B.2 Capability mapping to OpenXR

B.2.1 Mapping overview

Capability	Corresponding OpenXR capability	Parameters	Corresponding OpenXR object
Create an XR System	xrGetSystem()	`xrSystemIdentifier`	XrSystemId* systemId;
Query XR System's graphics properties	xrGetSystemProperties()	`swapchainSupported`	Implicit, since the OpenXR specification support of swapchain by design.
		`maxSwapchainImageHeight`	uint32_t maxSwapchainImageHeight;
		`maxSwapchainImageWidth`	uint32_t maxSwapchainImageWidth;
		`maxLayerCount`	uint32_t maxLayerCount;
Query XR System's tracking properties	xrGetSystemProperties()	`orientationTracking`	XrBool32 orientationTracking;
		`positionTracking`	XrBool32 positionTracking;
Enumerate XR System's supported environment blend modes	xrEnumerateEnvironmentBlendModes()	Value '`opaque`' of `blendMode`	XrEnvironmentBlendMode* environmentBlendModes; There is one element of environmentBlendModes whose value is equal to XR_ENVIRONMENT_BLEND_MODE_OPAQUE.
		Value '`additive`' of `blendMode`	XrEnvironmentBlendMode* environmentBlendModes; There is one element of environmentBlendModes whose value is equal to XR_ENVIRONMENT_BLEND_MODE_ADDITIVE.

		Value '`alpha_blend`' of `blendMode`	XrEnvironmentBlendMode* environmentBlendModes; There is one element of environmentBlendModes whose value is equal to XR_ENVIRONMENT_BLEND_MODE_ALPHA_BLEND.
Enumerate supported view configuration types	xrEnumerateViewConfigurations()	Value '`monoscopic`' of `viewConfigurationPrimary`	XrViewConfigurationType* viewConfigurationTypes; There is one element of viewConfigurationTypes whose value is equal to XR_VIEW_CONFIGURATION_TYPE_PRIMARY_MONO.
		Value '`stereoscopic`' of `viewConfigurationPrimary`	XrViewConfigurationType* viewConfigurationTypes; There is one element of viewConfigurationTypes whose value is equal to XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO.
		Value '`other`' of `viewConfigurationPrimary`	XrViewConfigurationType* viewConfigurationTypes; There is one element of viewConfigurationTypes whose value is strictly greater than XR_VIEW_CONFIGURATION_TYPE_PRIMARY_STEREO and strictly lower than XR_VIEW_CONFIGURATION_TYPE_MAX_ENUM.
Enumerate the view configuration properties	xrEnumerateViewConfigurationViews()	`recommendedImageRectWidth`	uint32_t recommendedImageRectWidth;
		`maxImageRectWidth`	uint32_t maxImageRectWidth;
		`recommendedImageRectHeight`	uint32_t recommendedImageRectHeight;
		`maxImageRectHeight`	uint32_t maxImageRectHeight;
		`recommendedSwapchainSampleCount`	uint32_t recommendedSwapchainSampleCount;
		`maxSwapchainSampleCount`	uint32_t maxSwapchainSampleCount;
Enumerate reference space types	xrEnumerateReferenceSpaces()	Value '`view`' of `referenceSpace`	XrReferenceSpaceType* spaces; There is one element of spaces whose value is equal to XR_REFERENCE_SPACE_TYPE_VIEW.

		Value '`local`' of `referenceSpace`	XrReferenceSpaceType* spaces; There is one element of spaces whose value is equal to XR_REFERENCE_SPACE_TYPE_LOCAL.
		Value '`stage`' of `referenceSpace`	XrReferenceSpaceType* spaces; There is one element of spaces whose value is equal to XR_REFERENCE_SPACE_TYPE_STAGE.
		Value '`unbounded`' of `referenceSpace`	XrReferenceSpaceType* spaces; There is one element of spaces whose value is equal to XR_REFERENCE_SPACE_TYPE_UNBOUNDED_MSFT.
		Value '`user_defined`' of `referenceSpace`	
Query the spatial range boundaries	xrGetReferenceSpaceBoundsRect()	`2DSpatialRangeBoundaries`	XrExtent2Df* bounds;
Enumerate swapchain image formats	xrEnumerateSwapchainFormats	`swapchainImageFormatIdentifier`	int64_t* formats;
Enumerate swapchain images	xrEnumerateSwapchainImages()	`numberSwapchainImages`	uint32_t* imageCountOutput;
		`swapchainImages`	XrSwapchainImageBaseHeader* images;
Enumerate composition layer type	N/A	Value '`projection`' of `compositionLayer`	Part of the core specification
		Value '`quad`' of `compositionLayer`	Part of the core specification
	xrEnumerateInstanceExtensionProperties()	Value '`cylinder`' of `compositionLayer`	XrStructureType type; The variable type has the value XR_TYPE_COMPOSITION_LAYER_CYLINDER_KHR.
		Value '`cube`' of `compositionLayer`	XrStructureType type; The variable type has the value XR_TYPE_COMPOSITION_LAYER_CUBE_KHR.
		Value '`equirectangular`' of `compositionLayer`	XrStructureType type; The variable type has the value XR_TYPE_COMPOSITION_LAYER_EQUIRECT_KHR or

			XR_TYPE_COMPOSITION_LAYER_EQUIRECT2_KHR.
		Value '`depth`' of `compositionLayer`	XrStructureType type; The variable type has the value XR_TYPE_COMPOSITION_LAYER_DEPTH_INFO_KHR.

B.2.2 XR views and rendering loop

Those composition layers are drawn in a specified order, with the 0th layer drawn first. Layers are drawn with a "painter's algorithm," with each successive layer potentially overwriting the destination layers whether or not the new layers are virtually closer to the viewer. Composition layers are subject to blending with other layers. Blending of layers can be controlled by the alpha channel information present in the image buffer of each layer. In addition, the image buffer of the layer may be limited by a maximum width and a maximum height when rendering them such that they fit into the capabilities of the swapchains.

For visual rendering, the following applies:

1) To present images to the user, the runtime provides images organized in swapchains for the application to render into.

2) The XR Runtime may support different swapchain image formats and the supported image formats may be provided to the application through the runtime API. XR Runtimes typically support at least sRGB formats. Details may depend on the graphics API specified when creating the session.

3) *Swapchain* images may be 2D or 2D Array. Arrays allow to extract a subset of the 2D images for rendering. Multiple swapchain handles may exist simultaneously, up to some limit imposed by the XR runtime. Swap chain parameters include:

   - texture format identifier, a graphics API specific version of a format, for example sRGB.
   - width and height, expressing the pixel count of the images sent to the swapchain
   - faceCount, being the number of faces, which can be either 6 (for cubemaps) or 1
   - indication whether the swapchain is dynamic, i.e. updated as part of the XR rendering loop or static, i.e. the application releases only one image to this swapchain over its entire lifetime.
   - access protection, indicating that the swapchain's images are protected from CPU access

4) Once a session is running and in focussed state as introduced in clause 4.1.2, the following rendering loop is executed following Figure 4.1.4

   a) The XR Application retrieves the action state, e.g. the status of the controllers and their associated pose. The application also establishes the location of different trackables.

   b) Before an application can begin writing to a swapchain image, it first waits on the image to avoid writing to it before the Compositor has finished reading from it. Then an XR application synchronizes its rendering loop to the runtime. In the common case that an XR application has pipelined frame submissions, the application is expected to compute the appropriate target display time using both the predicted display time and predicted display interval. An XR Runtime is expected to provide and operate a swapchain that supports a specific frame rate.

   c) Once the wait time completes, the application initiates the rendering process. In order to support the application in rendering different views the XR Runtime provides access to the viewer pose and projection parameters that are needed to render the different views. The view and projection info is provided for a particular display time within a specified XR space. Typically, the target/predicted display time for a given frame.

   d) the application then performs its rendering work. Rendering work may be very simple, for example just directly copying data from the application into the swap chain or may be complex, for example iterating over

the scene graph nodes and rendering complex objects. Once all views/layers are rendered, the application sends them to the XR Runtime for final compositing including the expected display time as well as the associated render pose.

e) An XR Runtime typically supports (i) planar projected images rendered from the eye point of each eye using a perspective projection, typically used to render the virtual world from the user's perspective, and (ii) quad layer type describing a posable planar rectangle in the virtual world for displaying two-dimensional content. Other projection types such as cubemaps, equirectangular or cylindric projection may also be supported.

f) The XR application offloads the composition of the final image to an XR Runtime-supplied compositor. By this, the rendering complexity is significantly lower since details such as frame-rate interpolation and distortion correction are performed by the XR Runtime. It is assumed that the XR Runtime provides a compositor functionality for device mapping. A Compositor in the runtime is responsible for taking all the received layers, performing any necessary corrections such as pose correction and lens distortion, compositing them, and then sending the final frame to the display. An application may use multiple composition layers for its rendering. Composition layers are drawn in a specified order, with the 0^{th} layer drawn first. Layers are drawn with a "painter's algorithm," with each successive layer potentially overwriting the destination layers whether or not the new layers are virtually closer to the viewer. Composition layers are subject to blending with other layers. Blending of layers can be controlled by layer per-texel source alpha. Layer swapchain textures may contain an alpha channel. Composition and blending is done in RGBA.

g) After the compositor has blended and flattened all layers, it then presents this image to the system's display. The composited image is then blend with the user's view of the physical world behind the displays in one of three modes, based on the application's chosen environment blend mode:

- OPAQUE. The composition layers are displayed with no view of the physical world behind them. The composited image is interpreted as an RGB image, ignoring the composited alpha channel. This is the typical mode for VR experiences, although this mode can also be supported on devices that support video passthrough.

- ADDITIVE: The composition layers are additively blended with the real world behind the display. The composited image is interpreted as an RGB image, ignoring the composited alpha channel during the additive blending. This is the typical mode for an AR experience on a see-through headset with an additive display, although this mode can also be supported on devices that support video passthrough.

- ALPHA_BLEND. The composition layers are alpha-blended with the real world behind the display. The composited image is interpreted as an RGBA image, with the composited alpha channel determining each pixel's level of blending with the real world behind the display. This is the typical mode for an AR experience on a phone or headset that supports video passthrough.

h) Meanwhile, while the XR Runtime uses the submitted frame for compositing and display, a new rendering process may be kicked off for a different swap chain image.

B.2.3 Available Visualization Space implementation

B.2.3.1 Using OpenXR_XR_FB

The openXR XR_FB_scene extension allows to define the boundary room and also boundary space and objects in the space:

1. xrGetSpaceBoundingBox3DFB provides the defined rectangular cube XrRect3DfFB by defining the offset XrOffset3DfFB values x,y, z and the extend XrExtent3DfFB values width, height and depth in the x,y,z dimensions.

2. xrGetSpaceSemanticLabelsFB optionally provides a way to describe the semantic meaning of an space entity. It is recommended to use the label "3GPP-AvailableVisualizationSpace" when it is used to describe available visualization space.

B.2.3.2 Using xrComputeNewSceneMSFT

The XR_MSFT_scene_understanding extension allows defining the bounding volume in 3 forms:

1. XrSceneSphereBoundMSFT for defining a spherical available visualization space
2. XrSceneOrientedBoxBoundMSFT for defining a cuboid available visualization space. Note that the bounding box is defined by its center and its edge to edge dimensions around its center. Therefore, these values shall be translated to the values defined in 6.2.4.

Also note that the scene components outside of the available visualization space may be excluded from rendering by the runtime.

Annex C (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
2022-04	SA4#118e	S4-220504				Draft TS skeleton from the editor	0.1.0
2023-05	SA4#124	S4-231042				Introduction, Prerequisites including XR device architecture, metadata formats, visual capabilities, device types description, OpenXR annex (S4-230920)	0.2.0
2023-08	SA4#125	S4-231559				QoE Metrics (S4-231457), visualization space (S4-231454), clarifications (S4-231548), XR system capabilities (S4-231540), Device types (S4-231542)	0.3.0
2023-11	SA4#126	S4-231976				MSE metadata (S4-231861), QoE metrics (S4-231957), Audio capabilities (S4-231945), Video capabilities (S4-232031), Scene description (S4-232022)	0.4.0
2023-12	SA#102	SP-231300				Version 1.0.0 created by MCC	1.0.0
2024-01	SA4#127	S4-XXXXX				Draft TS restructuring (S4-240123), various fixes (S4-240294), Media capabilities (S4-240125, S4-240437), timing information format (S4-240223), metadata definition (S4-240366), capability signalling (S4-240425), Definition (S4-240434)	1.1.0
2024-03	SA#103	SP-240032				Version 2.0.0 created by MCC	2.0.0
2024-03						Version 18.0.0 created by MCC	18.0.0
2025-10	-	-	-	-	-	Update to Rel-19 version (MCC)	**19.0.0**
2026-03	SA#111	SP-260121	0003	2	A	Assignment of IVAS levels to device types	**19.1.0**

History

Version	Date	Status
V19.0.0	October 2025	Publication
V19.1.0	April 2026	Publication