**Source: Xiaomi (Rapporteur)**

**Title: MeCAR Permanent Document v3.0**

**Version: 3.0.0**

**Agenda Item: 14.4**

**Document for: Agreement**

**Work Item information:**

|  |  |
| --- | --- |
| **Work Item Name** | Media Capabilities for Augmented Reality |
| **Acronym** | MeCAR |
| **Unique ID** | 950015 |
| **Target release** | 18 |

**Source details:**

|  |  |  |
| --- | --- | --- |
| **Rapporteur** | **Name** | Emmanuel THOMAS |
| **Company** | Xiaomi |
| **Email** | thomase@xiaomi.com |
| **Editor** | **Name** | Gilles TENIOU |
| **Company** | Tencent |
| **Email** | teniou@tencent.com |

**Revision history:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Meeting** | **Subject/Comment** |
| 1.0.0 | 2022-04-14 | SA4#118e | * Added   + Working assumptions   + Example media capabilities |
| 2.0.0 | 2022-05-20 | SA4#119e | * Added   + Clause 2 on definitions, including Optical see-through device and Video see-through device   + Clause 4 on device categories, including draft EDGAR-1 architecture (4.2)   + Clause 5.3 on media capability validation framework, including Khronos 3D Commerce conformance example (5.3.1), a possible media capability evaluation framework (5.3.2) and its MeCAR scope (5.3.3)   + Clause 6 on sensor and user environment data types including the view-related information (6.2)   + Clause 7 on activities in external organizations, including the AVTCORE WG in IETF (7.1) |
| 3.0.0 | 2022-08-26 | SA4#120e | * Added:   + Clause 5.3 Examples of media capabilities based on existing specifications, based on TS 26.511.   + Clause 3.6 Transparency information.   + Clause 3.7 Media Type Categories and Characteristics   + Clause 6.2 User surroundings information   + Clause 5.2 Display capability |

# Contents

[Contents 3](#_Toc112399293)

[1 Introduction 5](#_Toc112399294)

[2 Definitions, symbols and abbreviations 5](#_Toc112399295)

[2.1 Definitions 5](#_Toc112399296)

[3 Working assumptions 6](#_Toc112399297)

[3.1 Prioritization of AR optical see-through 6](#_Toc112399298)

[3.2 Device design types 6](#_Toc112399299)

[3.2.1 General 6](#_Toc112399300)

[3.2.2 Device design type 1 6](#_Toc112399301)

[3.2.2 Device design type 2 8](#_Toc112399302)

[3.2.3 Device design type 3 9](#_Toc112399303)

[3.3 General functional architecture 9](#_Toc112399304)

[3.4 5G\_STAR EDGAR-type device architecture 10](#_Toc112399305)

[3.5 Media Access Function for AR 10](#_Toc112399306)

[3.6 Transparency information 11](#_Toc112399307)

[3.6.1 Interest of Transparency information 11](#_Toc112399308)

[3.6.2 Processing transparency information 12](#_Toc112399309)

[3.6.3 Carrying transparency information 12](#_Toc112399310)

[3.7 Media Type Categories and Characteristics 13](#_Toc112399311)

[3.7.1 General 13](#_Toc112399312)

[3.7.2 Media types definition 14](#_Toc112399313)

[3.7.3 Real time characteristics 15](#_Toc112399314)

[3.7.4 Traffic characteristics 15](#_Toc112399315)

[3.8 Visual rendering in OpenXR 15](#_Toc112399316)

[4 Device categories 18](#_Toc112399317)

[4.1 General 18](#_Toc112399318)

[4.2 Augmented Reality User Equipment (AR UE) 18](#_Toc112399319)

[4.2.1 Device architecture 18](#_Toc112399320)

[5 Media capabilities 20](#_Toc112399321)

[5.1 Categories of media capabilities 20](#_Toc112399322)

[5.2 Examples of media capabilities 20](#_Toc112399323)

[5.3 Display capability 21](#_Toc112399324)

[5.4 Examples of media capabilities based on existing specifications 22](#_Toc112399325)

[5.4.1 Possible 2D video profiles candidates 22](#_Toc112399326)

[5.5 Media capability validation framework 25](#_Toc112399327)

[5.5.1 Example framework by Khronos on 3D Commerce conformance (glTF viewer) 25](#_Toc112399328)

[5.5.2 Possible capability evaluation framework 27](#_Toc112399329)

[5.5.3 Possible scope of media capability 28](#_Toc112399330)

[6 Sensor and user environment data types 28](#_Toc112399331)

[6.1 General 28](#_Toc112399332)

[6.2 View-related information 28](#_Toc112399333)

[6.3 User surroundings information 29](#_Toc112399334)

[7 Relevant activities in external organizations 29](#_Toc112399335)

[7.1 IETF AVTCORE WG 29](#_Toc112399336)

[8 Technical status 30](#_Toc112399337)

[8.1 List of elements open for work 30](#_Toc112399338)

[8.2 List of completed elements 30](#_Toc112399339)

[8.3 List of remaining elements 30](#_Toc112399340)

[8.4 List of open issues 30](#_Toc112399341)

[9 References 30](#_Toc112399342)

# 1 Introduction

During SA4#117-e the New Work Item on “Media Capabilities for Augmented Reality” (MeCAR) in S4-220332 was agreed and afterwards approved in by SA#95e in SP-220242.

The media capabilities of AR devices typically contribute to three main functionalities that are simple media rendering, split-rendering, for which a pre/scene-rendering of the scene and views is carried out in the cloud/edge and uplink streaming of sensor and device data to the network in order to support network-based processing of device sensor information.

To support basic interoperability for AR applications in the context of 5G System based delivery, a set of well-defined media capabilities are essential to create the conditions of a successful ecosystem. Therefore, MeCAR work item defines those media capabilities for AR devices in a service-independent manner. In particular, the following objectives are considered:

* Define at least one AR device category that addresses the constraints of an EDGAR-type AR glass
  + Note: Additional device categories may be defined, but with lower priority.
* For each AR device category
  + Define a reference terminal architecture regarding media capability aspects for this AR device category
  + Define media types and formats produced and consumed by the AR device, including basic scene descriptions, audio, graphics and video as well as sensor information and metadata about user and environment.
  + Define the integration of the relevant existing 3GPP codecs into the reference terminal architecture
  + Define decoding capabilities, including support for multiple parallel decoders
  + Define encoding capabilities
  + Define security aspects related to the media capabilities
  + Define the required, recommended and optional media capabilities for this AR device category
* Integrate IVAS into suitable AR device categories, once IVAS is available
* Define capability exchange mechanisms based on complexity of AR media and capability of device to support EAS KPIs for provisioning of edge/cloud resources
  + Note: Identify a suitable existing capability framework, or if it does not exist, we need to work with the broader industry (e.g., IETF, KHRONOS, W3C, etc.) to get this done.
* Identify which QoE metrics from VR QoE metrics can be reused or enhanced for AR media (e.g., resolution per eye, Field of view (FOV), round-trip interaction delay, etc.) and define relevant KPIs that are dedicated to AR/MR
* Specify additional relevant KPIs and simple QoE Metrics for AR media
* Specify encapsulations into RTP, ISOBMFF and CMAF
* Specify the relevant codec-level parameters for session setup and negotiation of the media delivery and provide instantiations for SDP and DASH MPD
* Enable AR media in 5G Media Streaming by defining suitable 5GMS profiles based on AR media capabilities
* Define typical traffic characteristics for AR media

# 2 Definitions, symbols and abbreviations

## 2.1 Definitions

**Optical see-through device**:Device providing a view of the surrounding world by letting the light from the real world directly reaches the user’s eyes through an optical system.

**Video see-through device**: Device providing a view of the surrounding world by capturing the light from the real world and then presenting it through a display system to the user.

# 3 Working assumptions

## 3.1 Prioritization of AR optical see-through

Optical see-through devices have the advantage of not requiring any additional video streams for rendering the surrounding environment. The natural light passing through the lenses of the device provides the user with a clear and natural view of the surrounding environment and does not add additional rendering latencies since the only media data that needs to be streamed by the device is the AR data. Therefore, even display systems with limited capabilities can offer the user a high level of experience.

Prioritizing optical see-through devices will allow us to focus on the overlaid AR data that is displayed on the glasses and synchronized (time and space) with the real word. It will make possible to offer a good level of user experience relying on relatively lower KPIs, in terms of FoV, resolutions, etc., and leads to lighter constraints on the design of the glasses, making easier the emergence of near to mid-term solutions.

## 3.2 Device design types

### 3.2.1 General

The MeCAR Work Item aims at being in line with device manufacturing constraints so that the specifications as output of MeCAR when published can be as relevant as possible for the AR ecosystem. As a result, the following clause lists typical device designs that are considered to be the illustrative of the current or future manufactured AR devices. Those device types are not meant to constitute interoperability points in the MeCAR specifications but rather serve as reality check during the development of MeCAR work to verify this desired alignment. If and when gaps are identified in terms of media capabilities between those device types and the current draft specification, it would be expected to update the draft specification for a better alignment with those current and future AR devices.

The current list of device design types are :

* Device type 1: Standalone physically-constrained AR glasses
* Device type 2: 5GUE-therered physically-constrained AR glasses
* Device type 3: 5GUE-powered lightweight AR glasses

At present, the device design types are numbered in increasing order of media capability performances. However, this is not a fixed rule and more device design types in the future may be listed which would differ based on other criteria as long as they represent a realisation of AR Glasses with a different design.

### 3.2.2 Device design type 1

Looking at existing AR Glasses, based on the study in TR 26.998 [1] and based on information from chipset manufacturers on existing and emerging devices, an AR Glass designed for AR experiences does integrate complex functionalities and many of those relate to capabilities. Figure 1 is a picture providing an overview of an AR glass.



Hinge

SoC Media

Connectivity

Eye Tracking + Camera/Sensor Aggregator

Figure 1 - Overview of AR glasses of device design type 1

Typical functions of such a AR glass consists of:

* Peripheries including
  + Displays
  + Cameras
  + Microphones
  + Sensors
  + Camera/Sensor Aggregators
  + Perception functionality: Eye Tracking, Face Tracking, etc.
* SoC Media
  + Display Processing
  + GPU functionalities: Composition/Reprojection
  + Decoding
  + Decryption
  + Camera Front ends
  + Perception functionality: 6DoF, etc.
  + Encoding
* Connectivity
  + Wi-Fi, Bluetooth, 5G, etc.

An interesting aspect to consider from the above is that the device consists of different thermal islands, hence division in multiple chips in the headset is highly desirable. This means that both minimizing the power consumption per thermal island as well as minimizing the overall power consumption is an essential design constraint for the device battery life.

In addition, such type of devices require to partition workloads to remote devices or the cloud to some extent to balance the power load. Based on this, media capabilities are also possibly required on UE that acts as a hub for a tethered glass. Architectures and processing for this will be discussion SmartAR. The main target device in the MeCAR work item remains glasses as shown above.

It should be noted that such AR glasses are predominantly served with media that can directly be rendered by the peripheries, or produce media captured on the device and sent to remote processing.

It is considered that for media capabilities related to this primary AR category, only capabilities of the SoC media are to be part of the media capability definitions. We also note that the XR experience observed by the user depends on more aspects than the media capabilities, such as the display, the optics, the quality of the sensors, the stability of the connection and so on. However, such aspects are not considered to be part of the media capabilities for AR.

Initial System-on-Chip (SoC) media will likely rely on existing hardware, for example from lower end mobile chipsets. Some people consider XR even a hack that uses existing components in a smart manner. However, a core aspect of XR experiences different from traditional mobile devices is the concurrent operation of multiple encoders and/or decoders to address different sensors, eye buffers, layers and so on, as well as the rendering to GPU instead of directly going to the display.

Only over time, such hardware will get added specific functionalities, but not in the near and mid-term. Expected in the future are higher render and display resolutions, multi-layer composition, etc.

Given that many functionalities are defined through Khronos OpenXR, defining capabilities for example by mandating or recommending support of certain APIs or parameter settings on API may be relevant. In some cases it may not even be possible to define capabilities, but for example rely on test signals and benchmarking requirements that estimate the performance of a device.

Diagram

Description automatically generated

Figure 2 - Example architecture of AR glasses based on device design type 1

Based on these observations, an initial main objective of a standard is to create near to mid-term interoperability for media capabilities based existing and emerging media SoCs.

### 3.2.2 Device design type 2

Similar as in the case for the device design type 1 introduced in 3.2.1, the AR Glasses runs an AR/MR application that uses the capabilities of the glass to create a service.

However, the AR glasses does not provide a 5G connection, but is tethered to a 5G device. The AR glasses may only use the 5G connectivity of the phone, or it may use capabilities on the phone for additional processing support. As a result, this device type is expected to have higher media capabilities compared to the device type 1 which is the most constrained one.

Waterfall chart

Description automatically generated

Figure 3 - Example architecture of AR glasses based on device design type 2

### 3.2.3 Device design type 3

The AR Glasses is tethered to a 5G device that includes the application and the XR functions. The tethering may be wireless or wired, but it is proprietary.

The 5G device runs the application that uses the Media access and rendering capabilities of the 5G device to run an AR/MR experience. The AR glass is connected to the 5G Device, but the XR runtime API is exposed to the 5G device/phone.

Diagram

Description automatically generated

Figure 4 - Example architecture of AR glasses based on device design type 3

In this case, the connection between the phone and glass is handled by a proprietary system that tethers the XR Runtime API running on the 5G device to the XR Runtime core functions on the glass. The overall function is referred to as XR Link.

In order to determine the media capabilities of such a device, it is assumed that the media access and rendering functions of a high-end smart phone can be used.

## 3.3 General functional architecture

For any type of AR devices targeted by MeCAR, the functional architecture depicted in Figure 5 is applicable.



Figure 5 - General functional architecture of AR device

## 3.4 5G\_STAR EDGAR-type device architecture

From TR 26.998 [1], the architecture of the EDGAR device type was defined as illustrated in Figure 6. Note that EDGAR in TR 26.998 stands for EDGe-dependent AR (EDGAR) UE.



Figure 6 - Architecture of 5G\_STAR EDGAR-type device

## 3.5 Media Access Function for AR

The Media Access Function defined in TR 26.998 [1] supports the AR UE to access and stream media. Figure 7 depicts its different functions and buffer elements.



Figure 7 - Media Access function for AR as defined in TR 26.998 [1]

## 3.6 Transparency information

### 3.6.1 Interest of Transparency information

It is desirable to support the transmission of transparency information (alpha\_channel) in addition to the colour (e.g., RGB) information. Augmented reality services may overlay of virtual objects on the real world which are accessed directly through the “optical see-through” glasses. The overlay is not a full picture but only part of it, the other pixels of the picture being transparent or partially transparent, in case of a shadow effect for instance.

Figure 8 below depicts the overlay of a virtual dragon on the table of a real living room. If the whole video is overlayed, the dragon may appear in the middle of a rectangle corresponding to the video size. This is illustrated on the left picture. With additional transparency information, only the part of the video corresponding to the dragon is overlayed, as illustrated on the right picture.

A picture containing floor, indoor, window, living

Description automatically generatedA picture containing indoor, floor, window, living

Description automatically generated

Figure 8 - Video overlay without (left) and with (right) transparency information

### 3.6.2 Processing transparency information

[Editor’s note] This chapter is a preliminary draft. Further study is required on this topic.

Depending on how the transparency information is carried, it may be processed at different functional blocs.

For example, transparency information may be processed in the Media Access Function. This may be the case if the transparency information is carried in media tracks. This also includes the decoder if transparency information is carried as auxiliary pictures. Transparency information may also be processed by an AR scene manager, when the transparency information is carried in the scene description.

Wherever the processing is located, transparency information is forwarded to the display so that only the appropriate parts of the picture are displayed.

### 3.6.3 Carrying transparency information

#### 3.6.3.1 Carriage as auxiliary pictures in the video stream

The carriage of transparency information may be achieved by using the concept of auxiliary pictures defined by both the AVC and HEVC codecs.

##### 3.6.3.1.1 AVC

The AVC ([H.264](https://www.itu.int/rec/dologin_pub.asp?lang=f&id=T-REC-H.264-202108-I!!PDF-E&type=items)) specification provides guidelines for carrying transparency information.

It defines the concept of alpha-blending in clause 3.5 : *“A process not specified by this Recommendation | International Standard, in which an auxiliary coded picture is used in combination with a primary coded picture (…) the samples of an auxiliary coded picture are interpreted as indications of the degree of opacity (or, equivalently, the degrees of transparency) associated with the corresponding luma samples of the primary coded picture*.” AVC specification precises in clause 3.7 that “*An auxiliary coded picture must contain the same number of macroblocks as the primary coded picture. Auxiliary coded pictures have no normative effect on the decoding process*.” It also mentions (clause 3.1) that “*In addition to the primary coded picture, an access unit may also contain (…) one auxiliary coded picture*”.

Clause 7.3.2.1.2 (Sequence parameter set extension RBSP syntax) of the AVC specification defines fields related to alpha blending (alpha\_incr\_flag, alpha\_opaque\_value and alpha\_transparent \_value) and the semantics are detailed in clause 7.4.2.1.2 (Sequence parameter set extension RBSP semantics). The same clause also explains that “aux\_format\_idc equal to 1 indicates that exactly one auxiliary coded picture is present in each access unit of the coded video sequence, and that for alpha blending purposes the decoded samples of the associated primary coded picture in each access unit should be multiplied by the interpretation sample values of the auxiliary coded picture in the access unit in the display process after output from the decoding process.”

##### 3.6.3.1.2 HEVC

HEVC also defines how to carry an alpha channel in the same video bitstream as the base video.In this case, each frame contains two parts: a base layer containing the video, and an alpha layer containing the alpha channel information. Both layers are compressed using the HEVC codec. The two layers are signalled by a specific HEVC syntax element, namely a specific alpha channel information SEI message has to be added, so that the decoder knows how to interpret the auxiliary pictures. A decoder incapable of handling this SEI message only decodes the base layer.

The concept of auxiliary picture is defined in Annex F of the HEVC ([H.265](https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-H.265-202108-I!!PDF-E&type=items)) specification:

F.3.5 auxiliary picture: A *picture* that has no normative effect on the *decoding process* of *primary pictures*, and with a nuh\_layer\_id value such that AuxId[ nuh\_layer\_id ] is greater than 0*.*

In the same Annex, Table F.2 details the different types of auxiliary pictures:

Table F.1 – Mapping of AuxId to the type of auxiliary pictures

|  |  |  |  |
| --- | --- | --- | --- |
| **AuxId** | **Name of AuxId** | **Type of auxiliary pictures** | **SEI message describing interpretation of auxiliary pictures** |
| 1 | AUX\_ALPHA | Alpha plane | Alpha channel information |
| 2 | AUX\_DEPTH | Depth picture | Depth representation information |
| 3..127 |  | Reserved |  |
| 128..159 |  | Unspecified |  |
| 160..255 |  | Reserved |  |

#### 3.6.3.2 Carriage in ISOBMFF-based formats

ISO/IEC 14496-12 (ISO Base Media File Format) defines a general format which is used as a basis for defining other carriage formats such as MP4, CMAF, HEIF, AVIF, etc. It defines general concepts for the transport and carriage of data in an ISOBMFF-based container format and therefore nothing specific for the transport of auxiliary pictures is mentioned. Among many other things, ISOBMFF specifies the concept of non-timed items that can be used to store static images, and the concept of tracks that can be used to define the carriage of timed video. The format is flexible enough to allow carriage of auxiliary picture information, such as transparency, in the same structure (e.g., using a single item or a single video track) as well as using multiple tracks or items, with dependencies signaled by the item or track reference concept. For example, in the latter case an auxiliary video track needs to be used together with the 'auxl' track reference type as defined in clause 8.3.3.

HEIF (High Efficiency Image File format, ISO/IEC 23008-12) builds on top of ISOBMFF and defines a format for carriage of images and image sequences. Because the format is derived from ISOBMFF, transparency information can be carried in a single item or using multiple items, although the latter requires an 'auxl' item reference type similar to the video example above.

MIAF (Multi Image Application Format, ISO/IEC 23000-22), which defines additional constraints for HEIF, describes in its clause 7.3.5 how MIAF auxiliary image items can be used to carry alpha planes. In particular, it mentions that “*MIAF renderers shall interpret alpha planes and should support alpha blending using alpha-plane auxiliary images. This is especially important for image overlays*”. It also details in its clause 6.7 (MAIF renderer processing model) the interpretation of alpha planes.

#### 3.6.3.3 Carriage in Scene Descriptions

Possible carriage of transparency information or associated transparency information via the scene description has also to be considered. In particular, this may be a solution when an auxiliary picture carrying transparency information is available and not to rely on the SEI messages which not be decoded by all decoders (see HEVC chapter).

As an example for carrying transparency information in scene descriptions, clause 3.9 on Materials in the glTF 2.0 specification has a subclause “additional-texture” and a subclause “alpha-coverage” which describe how to use the fourth (alpha) component, if present, of a texture.

## 3.7 Media Type Categories and Characteristics

### 3.7.1 General

AR/MR applications are expected to deal with various media types that have different characteristics and levels of dependency on real-time transport. This clause introduces how and where some of the AR/MR media types are defined and provides an assessment on their levels of dependency on real-time transport and size/bitrate characteristics.

### 3.7.2 Media types definition

Table 2 - AR/MR Media Type definitions

|  |  |  |
| --- | --- | --- |
| **AR/MR**  **Media Type** | **Definition** | **Media Type Description (Examples)** |
| **Scene Description** | Clause 4.4.2 of 3GPP TR 26.998[1]  Scene description is used to describe the composition of a 3D scene, referencing and positioning the different 2D and 3D assets in the scene typically using a tree or a graph structure. | **Type**: glTF2.0, JSON format  **Organization**: MPEG-I and Khronos |
| **Spatial Description** | Clause 4.4.7.3 of 3GPP TR 26.998[1]  Visual features, keyframes, and spatial maps are used for mapping the real world, typically as part of the SLAM process. | **Type**: For example:  [https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html#XR\_FB\_spatial\_entity](https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html" \l "XR_FB_spatial_entity)  **Organization:** OpenXR |
| **3D Visual Model** | Clause 4.6.3.5.2 of 3GPP TR 26.928[2]  3D visual object description as a list of vertices, faces and other elements, along with associated attributes. | **Type**: PoLYgon  **Organization:** None |
| **AR Anchor** | The AR anchor is meant to identify a point in the user space to be used to anchoring a visual object (2D or 3D) | **Type**: Metadata allowing accurate overlaying/rendering of text, graphics or video contents to support Use Case 8 of TR 26.928.  **Organization:** None |
| **User Pose** | Clause 4.4.3.1 of 3GPP TR 26.998[1]  Representation of the user position and orientation | **Type**: It consists of a quaternion for orientation and a 3D vector for position. Timestamp is represented by a 64 bit monotonically increasing nano-second-based integer.  **Organization**: Khronos OpenXR |
| **FOV** | Y.6.2.3 of 3GPP TS 26.114[3]  The Field of View (FOV) is the extent of observable world at any given moment | **Type**: It consists of vertical fov and horizontal fov.  **Organization:** None |
| **Viewport** | Y.7.2 of 3GPP TS 26.114[3]  The viewport corresponds to the projection of the user View onto a target display | **Type**: It shall contain all of the parameters Viewport\_azimuth, Viewport\_elevation, Viewport\_tilt, Viewport\_azimuth\_range and Viewport\_elevation\_range  **Organization:** None |
| **Gesture** | TBD | **Type**: A array of finger joint position.  For example: [https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html#XR\_EXT\_hand\_tracking](https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html" \l "XR_EXT_hand_tracking)  **Organization:** OpenXR |
| **Body action** | TBD | **Type**: bvh format.  **Frequency**: at least 1kHz  **Organization:** BioVision company |
| **Facial expression** | OpenXR | **Type**: An array of key point position.  For example: [https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html#XrSystemFacialTrackingPropertiesHTC](https://www.khronos.org/registry/OpenXR/specs/1.0/html/xrspec.html" \l "XrSystemFacialTrackingPropertiesHTC)  **Organization:** None |
| **Sensor information** | TBD | **Type**: TBD  **Organization:** None |
| **Camera information** | TBD | **Type**: The camera parameters such as focal length, principal points, calibration parameters and the pose of the camera all contribute in understanding the relevance between points in the volumetric scene and pixels in the captured image.  **Organization:** None |
| **Projection information** | Clause Y.3 of 3GPP TS 26.114 and H264 protocol  Parameters associated to the projection from the 3D scene to the user display | **Type**: Projection mapping information (indicating the projection format in use, e.g., Equirectangular projection (ERP) or Cubemap projection (CMP)), for the projection sample location remapping process as specified in clauses 7.5.1.3 and 5.2 of ISO/IEC 23090-2 [179]  **Organization:** None |

[Editor’s note]: The list of Media Type is a starting point and other types of AR media will be added.

### 3.7.3 Real time characteristics

Table 3 – Real Time Characteristics of AR/MR Media Types

|  |  |
| --- | --- |
| **AR/MR Metadata**  **Media Type** | **Real Time Characteristics** |
| **Scene Description** | Non timed |
| **Spatial Description** | Sparsely timed (event based) |
| **3D Model** | Non timed |
| **AR Anchor** | Sparsely timed (event based) |
| **User Pose** | Continuous |
| **FOV** | Non timed |
| **Viewport** | Continuous |
| **Gesture** | Continuous |
| **Body action** | Continuous |
| **Facial expression** | Continuous |
| **Sensor information** | Non timed |
| **Camera information** | Sparsely timed (event based) |
| **Projection information** | Sparsely timed (event based) |

### 3.7.4 Traffic characteristics

Table 4 – Typical size and bitrate of AR/MR Media Types

|  |  |
| --- | --- |
| **AR/MR Metadata**  **Media Type** | **Typical size or bitrate** |
| **Scene Description** | TBD |
| **Spatial Description** | TBD |
| **3D Model** | TBD |
| **AR Anchor** | TBD |
| **User Pose** | TBD |
| **FOV** | TBD |
| **Viewport** | TBD |
| **Gesture** | TBD |
| **Body action** | TBD |
| **Facial expression** | TBD |
| **Sensor information** | TBD |
| **Camera information** | TBD |
| **Projection information** | TBD |

## 3.8 Visual rendering in OpenXR

As described in the OpenXR Reference Guide [7], an OpenXR application is composed of different cycles as depicted in Figure 9.

Diagram

Description automatically generated

Figure 9 - OpenXR application lifecycle [7]

After creating an OpenXR session, the application starts a frame loop. The frame loop is executed for every frame. The frame loop consists of the following steps:

1. Synchronize actions: this step consists of retrieving the action state, e.g. the status of the controller buttons and the associated pose. During this step, the application also establishes the location of different trackables. The application may also send haptics feedback.
2. Start a new frame: this step starts with waiting for a frame to be provided by the XR runtime. This step is necessary to synchronize the application frame submission with the display. The xrWaitFrame function returns a frame state for the requested frame that includes a predictedDisplayTime, which is a prediction of when the corresponding composited frame will be displayed. This information is used by the application to request the predicted pose at display. Once the xrWaitFrame function completes, the application calls xrBeginFrame to signal the start of the rendering process.
3. Retrieve rendering resources: the application starts by locating the views in space and time by calling the xrLocateViews function, provided with the predicted display time and the XR space. It then acquires the swap chain image associated with every view of the composition layer. It waits for the swap chain image to be made available so it can write into it.
4. Rendering: the application then performs its rendering work. This is for instance what the scene manager is tasked with. It iterates over the scene graph nodes and renders each object to the view. This step usually uses a Graphics Framework such Vulkan, OpenGL, or Direct3D to perform the actual graphics operations.
5. Release resources: once the rendering is done for a view, the application releases the corresponding swap chain image. Once all views are rendered, it sends them for display by calling the xrEndFrame function.

In terms of rendering operation, the relevant part is located between the call to xrBeginFrame and the call to xrEndFrame on the bottom right part of the diagram.

When the application calls the xrEndFrame function, the application provides the structure XrFrameEndInfo which contains all necessary information to render the frame that is:

* The time at which this frame should be displayed.
* The mode to be used for blending the user’s envriromnent with the submitted frame
* One or more layers which composes the submitted frame

As documented in the OpenXR specification:

“XrFrameEndInfo may reference swapchains into which the application has rendered for this frame. From each XrSwapchain only one image index is implicitly referenced per frame, the one corresponding to the last call to xrReleaseSwapchainImage.”

This describes how the XR runtime and the application can exchange visual data, i.e. via the use of swapchains.

A key feature of the XR runtime is its ability to perform layer composition. A Compositor in the runtime is responsible for taking all the received layers from xrEndFrame calls, 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. The number of supported composition layers may be queried by the application.

OpenXR supports different types of layers, with the main ones being:

* Projection Composition Layer: represents planar projected images, one rendered for each eye using a perspective projection.
* Quad Composition Layer: is useful for rendering user interface elements or 2D content on a planar area in the world.
* Cube Composition Layer: consists of a cube map with 6 views to be rendered by the application.
* Equirectangular Composition Layer: consists of an equirectangular image that is mapped onto the inside of a sphere in the world.
* Depth Composition Layer: provides an extra composition layer to allow applications to submit depth maps to assist with the pose correction of projected images of a project layer.

Figure 10 depicts an example of a projection composition layer and the resulting composited distorted image (image courtesy of Khronos).

A screenshot of a video game

Description automatically generated

Figure 10 - Example illustrating composition of a stereoscopic image submitted to the Compositor

Another relevant configuration when setting up the XR session is the choice of the view configuration, which depends on the target device and its capabilities. Mono and Stereo are natively supported by all XR runtimes. Some advanced types like the primary quad, defined as a vendor extension provide support for foveated rendering.

# 4 Device categories

## 4.1 General

This clause collects the work carried out in the MeCAR Work Item related to the various device categories.

## 4.2 Augmented Reality User Equipment (AR UE)

### 4.2.1 Device architecture

[Editor’s note] At SA4#119, this section was added while further improvements were improved. In particular, the audio rendering may need further clarification in terms of interfaces and roles between the AR/MR Application, the AR Scene Manager and the AR Runtime.

[Editor’s note] At SA4#120, it was identified to be high priority for the upcoming telcos to stabilise the AR UE architecture so that the work can progress in lower level of details such as:

* Identifying the subsets of API that can be part of the AR Runtime API
* Identifying the mandatory aspects present in OpenXR related to media, e.g. projection format for the images in the, etc…

Figure 11 provides the technical architecture of the AR UE.

Diagram

Description automatically generated

Figure 11 - Device architecture of AR UE

The AR UE is regular 5G UE with 5G connectivity provided through an embedded 5G modem and 5G system components. The AR UE also features several sensors and user controllers relevant for AR experiences that are cameras, microphones, speakers, display and generic user input. The AR/MR Application is responsible for orchestrating the various device resources to offer the AR experience to the user. In particular, the AR/MR Application can leverage three main internal components on the device which are:

* The Media Access Functions (MAF)
* The AR Runtime
* The AR Scene Manager

The AR/MR Application can communicate with those three components via dedicated APIs called the MAF-API, the AR Scene Manager API and the AR Runtime API. Among other functionalities, those APIs enables the AR/MR Application to discover and query the media capabilities in terms of support as well as available resources at runtime. Regarding rendering, the AR/MR application obtains the head pose information from the AR Runtime which is then provided to the AR Scene Manager. Based on this information, the AR Scene Manager determines the objects visible to the user at a given point in time or more generally the objects that may be needed to be rendered in the next rendering cycles. The AR Scene Manager then submits the rendered views to the AR Runtime as frames written to the images of the Swapchains, of which formats where configured beforehand by the AR/MR Application using the information provided by the AR Runtime API. From those images in the Swapchains, the AR Runtime then generates the left and right eye buffers possibly based on late adjustment techniques using updated head pose information, if available, commonly known as late stage reprojection (LSR).

Once the AR/MR application is running, the downlink media flows from the 5G System to the MAF in compressed form and then from The MAF to the AR Scene Manger in a decoded form. In parallel, the AR UE is capable of establishing an uplink data flow from the AR Runtime to the MAF wherein the data may be in an uncompressed form and then from the MAF to the 5G System wherein the MAF may have compressed the data in order to facilitate the expected transmission over the network.

# 5 Media capabilities

## 5.1 Categories of media capabilities

Media capabilities may be defined for those categories:

* Audio
  + Capture
  + Playback
  + Codec
  + Formats
  + Framework (multiple codecs, etc.)
* Camera
  + RGB
  + Depth
* Display
  + Processing
  + Number of Displays
  + Bit depth
  + Color format
* GPU
  + Functionalities/APIs
  + Performance
* Security
  + Content Protection
  + Cryptography
  + Key Management
* Non-media sensors
  + Types: Accelerometer, Magnetometer, Gyroscope, ambient light
  + Access for example through OpenXR APIs
* Video
  + Playback/Decoding
  + Processing
  + Recording/Encoding
  + Formats (bit depth, color components, chroma subsampling, etc.)
  + Framework (multiple codecs, etc.)
* Runtime
  + APIs
  + Performance

## 5.2 Examples of media capabilities

Given the categories listed in clause 3.1, the following are examples of media capabilities for those categories:

* Video
  + Playback/Decoding
    - H.264 High, Main and Baseline profile
    - H.265 Main and Main 10 Profile
    - Maximum processing: Up to 8,294,400 Macroblocks per second (corresponding to 8192x4320 @ 60fps)
    - HEIF
  + Processing
  + Recording/Encoding
    - H.264 High, Main and Baseline profile
    - H.265 Main and Main 10 Profile
    - Maximum processing: Up to 3,888,000 Macroblocks per second (corresponding to 3840x2160 @ 120fps)
    - Low-latency encoding
    - Error-robustness, slicing, intra refresh, long term prediction
  + Formats
    - 8-Bit: NV12, UBWC, YV12, RGBA888
    - 10-Bit: UBWC TP10, P010
  + Framework (multiple codecs, etc.)
    - Maximum number of combined encoding and decoding instances: 16
* GPU
  + Functionalities
    - tbd
  + Performance
    - Examples
      * 3D Triangle Rate
      * 3D Pixel Draw Rate
      * Texture Fetch Rate
      * Z reject rate (pixels/sec)
    - The issue is that GPU capabilities are more defined through benchmarks. A way to address is to define a set of test signals that a GPU needs to be able to handle in real-time.
* Audio
  + Capture
  + Playback
  + Codec
  + Formats
  + Framework (multiple codecs, spatial audio support etc.)
    - Low-Latency: input, output, roundtrip
    - Game Audio Playback up to 8/16/32 simultaneous streams

## 5.3 Display capability

The capability of a display for AR glasses may be characterised by its ability to reproduce colour and the ability to change its opacity comprised in to a given opacity range.

The followings are the relevant information to describe the capability of such display.

* display
  + Reference colour space (For example, [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR\_FB\_color\_space](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html%23XR_FB_color_space))
  + Perceptible colours (For example, [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR\_KHR\_composition\_layer\_color\_scale\_bias](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html%23XR_KHR_composition_layer_color_scale_bias))
  + Coordinate of primary colours in the reference colour space (For example, [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR\_FB\_color\_space](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html%23XR_FB_color_space))
  + Colour map (For example, [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XrPassthroughColorMapMonoToRgbaFB](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html%23XrPassthroughColorMapMonoToRgbaFB))
  + Opacity range (For example, [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XrPassthroughStyleFB](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html%23XrPassthroughStyleFB))
* sensor
  + Ambient light intensity (For example, <https://www.w3.org/TR/ambient-light>)
  + Ambient light intensity range

NOTE: The ambient light parameters from W3C have been primarily defined for regular displays. It is still to be checked if they are applicable to AR glasses.

The reference colour space is the colour space in which the display is compatible.

The perceptible colours is a list of the colours perceptible or reproducible. Primary colours such as R, G, B and their coordinates in the reference colour space are listed. The display may provide the colour space coordinate according to dedicated reference condition, or according to measured ambient light.

The colour map is the array of colours that remapped by the device. It is a list of colours and their target RGBA values in the reference colour space. The application may provide the colours map according to dedicated display capability of a device.

The opacity range is the range of supported opaque/transparency level of the display. From fully opaque as 1.0 to fully transparent as 0.0, the display may provide its capability on blocking light rays from outside.

The ambient light intensity is the intensity of the ambient light measured at a given point in time.

The ambient light intensity range is the minimum and maximum level of light in the unit of lux that the sensor can measure and provide.

## 5.4 Examples of media capabilities based on existing specifications

In the next chapter, different video profiles are listed which may be considered as relevant for consideration in MeCAR. These candidates do not reflect the final recommendations which MeCAR will produce. Additionally the lists may be refined and updated as the work in MeCAR progresses.

### 5.4.1 Possible 2D video profiles candidates

2D video encoding and decoding capabilities have of course to be considered. First, a basic scenario for AR is the virtual TV-set, where a 2D video stream is displayed as an overlay to the real world (for instance on the wall of the living room). Then, 2D video encoding/decoding capabilities may be used to display 3D objects on AR glasses, using the stereoscopic effect, by feeding the two eyes buffers of the AR glasses with the appropriate 2D videos.

It is referred here to video capabilities which have been referenced in clause 4.2 of TS 26.511 (“5G Media Streaming, Profiles, Codecs and Formats”) [5]. No video profiles published in TS 26.118 (“Virtual Reality (VR) profiles for streaming applications”) [6] are here listed. They indeed refer to a 360° immersive context, which may not be relevant for the optical see-through device types.

Lastly, the here listed candidates do not preclude of the result of the work which shall be carried on within MeCAR. Especially, if both encoding and decoding profiles are listed as they are referenced in clause 4.2 of TS 26.511 [5], requirements for encoding and decoding capabilities may differ.

#### 5.4.1.1 AVC

##### 5.4.1.1.1 Decoding

TS 26.511 [5] has defined the following decoding capabilities for H.264 (AVC4) in its clause 4.2.1.1:

- **AVC-HD-Dec**: the capability to decode H.264 (AVC) Progressive High Profile Level 3.1 [2] bitstreams, for which the maximum VCL Bit Rate is constrained to be 14 Mbps with cpbBrVclFactor and cpbBrNalFactor being fixed to be 1,000 and 1,200, respectively.

- **AVC-FullHD-Dec**: the capability to decode H.264 (AVC) Progressive High Profile Level 4.0 [2] bitstreams.

- **AVC-UHD-Dec**: the capability to decode H.264/AVC Progressive High Profile Level 5.1 [2] bitstreams for H.264/AVC with the following additional restrictions and requirements:

- the maximum VCL Bit Rate is constrained to be 120 Mbps with cpbBrVclFactor and cpbBrNalFactor being fixed to be 1250 and 1500, respectively.

- the bitstream does not contain more than 10 slices per picture.

NOTE: High Profile for H.264/AVC excludes Flexible macro-block order, Arbitrary slice ordering, Redundant slices, Data partition.

##### 5.4.1.1.2 Encoding

The corresponding encoding capabilities (in the sense that their bitstream are decodable by the decoder with the same name) have been defined:

- **AVC-HD-Enc:**

- up to 108,000 macroblocks per second;

- up to a frame size of 3,600 macroblocks;

- up to 120 frames per second;

- the chroma format being 4:2:0; and

- the bit depth being 8 bits;

- **AVC-FullHD-Enc:**

- 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 bits;

- **AVC-UHD-Enc:**

- up to 983,040 macroblocks per second;

- up to a frame size of 36,864 macroblocks;

- up to 480 frames per second;

- the chroma format being 4:2:0; and

- the bit depth being 8 bits.

#### 5.4.1.2 HEVC

##### 5.4.1.2.1 Decoding

TS26.511 [5] has defined the following decoding capabilities for H.265 (HEVC) in its clause 4.2.2.1:

- **HEVC-HD-Dec**: the capability to decode H.265 (HEVC) Main Profile, Main Tier, Level 3.1[3] 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-FullHD-Dec**: the capability to decode H.265 (HEVC) Main10 Profile, Main Tier, Level 4.1[3] 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[3] 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[3] 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 further limitations:

- the bitstream does not exceed the maximum luma picture size in samples of 33,554,432,

- the maximum VCL Bit Rate is constrained to be 80 Mbps with CpbVclFactor and CpbNalFactor being fixed to be 1000 and 1100, respectively.

Note that HEVC\_8K-Dec is mentioned as a reminder of TS 26.511 references. It may not be relevant in the context of AR glasses. As said in introduction to this chapter, this discussion is left for further study.

##### 5.4.1.2.2 Encoding

The corresponding encoding capabilities (in the sense that their bitstream are decodable by the decoder with the same name) have been defined:

- **HEVC-HD-Enc**:

- up to 33,177,600 luma samples per second;

- up to a luma picture size of 983,040 samples;

- up to 120 frames per second;

- the Chroma format being 4:2:0; and

- the bit depth being 8 bits;

- **HEVC-FullHD-Enc**:

- 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 bits;

- **HEVC-UHD-Enc**:

- 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 bits.

## 5.5 Media capability validation framework

### 5.5.1 Example framework by Khronos on 3D Commerce conformance (glTF viewer)

#### 5.5.1.1 General

The Khronos group defines many specifications that rely on hardware capabilities and, in particular, its specifications are largely powered by Graphics Processing Units (GPU). As a result, the deployment of Khronos specification depends significantly on the ability for a vendor to evaluate whether its products meets the requirement of those specifications.

To this end, Khronos offers the Khronos 3D Commerce Viewer Certification Program which “enables any company to demonstrate that their viewer is capable of accurately displaying 3D Products that have been created using the 3D Commerce asset creation guidelines”.

The relevant part in the context of MeCAR is the certification process described in [2].

Diagram

Description automatically generated

Figure 12 - Khronos' 3D commerce certification process

#### 5.5.1.2 Relevant steps in the MeCAR context

From this certification process only a subset of those steps are relevant for us which are:

* Viewer Test Package
  + What does it contain? What are the file formats?
* Run Certifications Test
  + How are those test described? Are the test objective or subjective? On which criteria and/or metrics do they rely on?
* Generates Results packages
  + How are expressed, in format, the performance of a 3D viewer against the tests? Is the result binary, i.e. passed/not passed? Or a score on a given scale with a minimum threshold?

To answer, those questions more documentation is available at the Khronos Group 3DC Certification repository [3]. The following was found based on the available documentation.

* Viewer Test Package
  + The package contains a list of glTF models [4]:
    - AnalyticalCubes
    - AnalyticalGrayscale
    - AnalyticalSpheres
    - GreenChair
    - Mixer
    - Shoe
    - TennisRacquet
    - WickerChair
* Run Certifications Test
  + The test plan defines how the tested viewer must operate to render the test models:
    - “The Certification Program Test Plan document defines the detailed requirements for generating the certification images.”
  + Some test are verified by mathematical functions some by humans.
    - “Certification renders will be evaluated programmatically and through human checks”
    - Example of subjective test:
      * “Strings should appear translucent outside of the blue star area”
    - Example of objective test:
      * “When scored by the evaluation tool included in the repository an SSIM or PSNR lower than their respective thresholds will automatically flag the image for review.”
* Generates Results packages
  + To evaluate whether a glTF viewer is conformant, the tested renders must generate images from the glTF model and those images are programmatically verified against reference renders.
    - “All certification images must be 1024x1024 and displayed according to the embedded cameras. The five retail models have three cameras each. One of the analytical models (spheres) is displayed in four different IBLs. All certification images need to be created according to the rules specified in the Test Plan document.”
  + How are expressed, in format, the performance of a 3D viewer against the tests? Is the result binary, i.e. passed/not passed? Or a score on a given scale with a minimum threshold?

#### 5.5.3.3 Takeaways from the certification process

Here are some takeaways from the certification test:

* A set of test models is essential for defining the test and the evaluation criteria.
* Objective tests are a minimum to pass but subjective tests via human verification are here to confirm for hard cases, e.g. transparency, reflection, etc.
* For objective tests, PSNR or SSIM is used to evaluated the rendered images from the test models.
* The tests are limited to static images and not rendering of the models over time.

### 5.5.2 Possible capability evaluation framework

In the context of MeCAR, the goal is not to certify a device but to define the media capabilities that are required at minimum for a given device category. The figure below depicts a possible workflow for implementing the evaluation of graphics capabilities in rendering glTF models and scenes.



Figure 13 - Possible framework for defining media and graphics capabilities

The first type of requirements is the playback of the test vectors. The test vectors are composed of a set of glTF tests models and scenes as well as pose traces. The MeCAR UE is supposed to render views of those glTF test models under the given poses coded in the test pose traces). The second type of requirements is whether the playback of the test vectors is correct. To this end, the generated views could be considered as a rendered videos (similar to the rendered image in the Khronos example). Such videos could be then checked against a reference video for the given test vector. The video validator could verify for the entire video:

* correct number of frames
* correct frame rate
* correct coded resolution of frames
* correct chroma sampling
* correct bit depth
* correct disparity between left and right views
* correct timing with respect to real-time rendering constraints

For each frame, the video validator could verify that each rendered image does not deviate too much from the reference image in the reference video. To validate the real-time nature of the rendering, the test run environment should also limit the time allowed to run the test scene.

### 5.5.3 Possible scope of media capability

In contrast to the Khronos example, the goal in MeCAR is not to establish a certification process. As a result, we would define the scope of the MeCAR graphics capability that does not fully cover the framework described in clause 3.1. The possible scope would cover he following elements:

* The glTF test models (possibly included media assets).
* The test pose traces associated with the glTF test models. The pose traces could be specific to each glTF test model.
* The test plan that defines the criteria to evaluate the rendered video (resolution, number of frames, etc…)
* Optionally, the generation of the reference rendered videos could be included to facilitate the reuse of this framework. However, since MeCAR may not define the reference scene render, providing these reference rendered video may actually go beyond MeCAR scope. This should be further discussed.

# 6 Sensor and user environment data types

## 6.1 General

This clause collects data types that can be consumed or produced by a MeCAR device and relates to Sensor and user environment data types.

## 6.2 View-related information

The device may generate static or dynamic data streams related to view information. The different types of data are listed as follows.

* Device pose information
  + Position in user’s volumetric space (maybe relative coordinate to user’s space origin)
  + Position in global space (maybe absolute coordinate)
  + Viewing direction in user’s volumetric space (maybe relative direction to user’s space origin)
  + Viewing direction in global space
  + Device pose timestamp
* Device frustum information
  + Field of view
  + Aspect ratio
  + Z-near
  + Z-far
  + Resolution of display
  + Screen to eye distance
  + Eye gaze point (on display or space)

The device may also receive media content representing pre-rendered views of a scene along with descriptive information. The different types of data are listed as follows.

* Pre-rendered media
* Codec, resolution, and profile of the pre-rendered media
* Estimated pose used to generate the pre-rendered media
* Estimated presentation time used to generate the pre-rendered media
* Frustum information of the pre-rendered media (in case device does not provide its frustum information)

## 6.3 User surroundings information

User surroundings is a structural and geometric representation of user's environment based on primitives. Examples of primitives are for instance mesh, plane and 3D object. The information on the user surrounding is listed as follows:

* User surroundings information
  + Primitive (including mesh, plane, and other shape of objects. For example, XR\_FB\_triangle\_mesh [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR\_FB\_triangle\_mesh](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html" \l "XR_FB_triangle_mesh) and XR\_MSFT\_scene\_understanding [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR\_MSFT\_scene\_understanding](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html" \l "XR_MSFT_scene_understanding))
  + Level of detail (for example, [https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XrMeshComputeLodMSFT](https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html" \l "XrMeshComputeLodMSFT))

# 7 Relevant activities in external organizations

## 7.1 IETF AVTCORE WG

MPEG has published a group of standards, under the umbrella of Visual Volumetric Video-based Coding (V3C). The V3C family of standards covers the aspects of encoding, storage, and transport of volumetric video and consists of three documents:

1. ISO/IEC 23090-5 Visual volumetric video-based coding (V3C) and video-based point cloud compression (V-PCC)
2. ISO/IEC 23090-12 MPEG Immersive video, which specifies the compression of volumetric video content captured by multiple cameras; and
3. ISO/IEC 23090-10 Carriage of visual volumetric video-based coding (V3C) data.

A V3C encoder converts volumetric video frames, i.e., 3D volumetric information, into a collection of 2D images, and associated data, known as atlas data. The converted 2D images are subsequently encoded using existing video or image/video codecs, while the atlas data is encoded with mechanisms specified in ISO/IEC 23090-5.

The RTP payload format for V3C atlas component is under development in the IETF AVTCORE WG (see Internet draft <https://datatracker.ietf.org/doc/draft-ilola-avtcore-rtp-v3c/>). The draft provides information on how the association between the V3C atlas component and the V3C video components can be done on SDP level, e.g., by defining groups of RTP streams to contain V3C encoded data (RFC 5888), or by defining a way to bundle multiple RTP streams in a single transport (RFC 8843).

The authors of the RTP payload format for V3C keep a public repository of the project where the latest status of the work can be followed: <https://github.com/laurilo/draft-ilola-avtcore-rtp-v3c>.

# 8 Technical status

## 8.1 List of elements open for work

The work-in-progress elements are:

* Reference terminal architecture for EDGAR-type
* Encoding/Decoding capabilities for EDGAR-type
* Media types and formats for EDGAR-type
* AR Audio Capabilities

## 8.2 List of completed elements

The completed elements thus far are:

* None

## 8.3 List of remaining elements

The following elements listed in the work plan remains to be started:

* To be started at SA4#121
  + Capability exchange mechanisms to support edge provisioning
  + Typical traffic characteristics for AR media
  + Addition of AR Media Capabilities for 5G Media Streaming
  + KPIs and simple QoE Metrics for AR media
* To be started at SA4#122
  + Integration of 3GPP codecs in the EDGAR-type architecture
  + Security aspects related to the media capabilities of the EDGAR-type
  + Encapsulations into RTP, ISOBMFF and CMAF
  + Codec-level parameter for SDP and DASH
  + Advanced Media Capabilities for AR media

## 8.4 List of open issues

The current open issues that are identified are:

* Audio integration aspects needs further attention in the development of the device architecture. Audio experts already initiated discussion over email thread during SA4#119e. Contribution and joint work with audio experts are encouraged in the upcoming telcos.

# 9 References

1. 3GPP TR 26.998, “Support of 5G Glass-type Augmented Reality / Mixed Reality (AR/MR) devices”
2. 3D Commerce Viewer Certification Program, <https://www.khronos.org/3dcommerce/certification/>
3. Khronos Group 3DC Certification documents, <https://github.com/KhronosGroup/3DC-Certification/>
4. Khronos Group 3DC Certification models, <https://github.com/KhronosGroup/3DC-Certification/tree/main/models>
5. 3GPP TS 26.511, “5G Media Streaming (5GMS); Profiles, codecs and formats”
6. 3GPP TS 26.118, “Virtual Reality (VR) profiles for streaming applications”
7. OpenXR 1.0 Reference Guide, <https://www.khronos.org/files/openxr-10-reference-guide.pdf>