**Source:** **Xiaomi, Qualcomm Incorporated, more?**

**Title: On frame submission to the AR Runtime in EDGAR-1 architecture**

## Document for: Agreement

## Agenda Item: 9.5

# 1 Introduction

At the 3GPP meeting #119-e, SA4 published the Permanent Document (PD) for MeCAR v2.0 [1]. During the post 119-e period, discussion on MeCAR took place during scheduled Video SWG telcos. This contribution proposes updates to the clause 4.2.1 Device architecture for EDGAR-1 device category based on these discussions around the interface between AR Scene Manager and the AR Runtime.

# 2 Background on OpenXR rendering

2.1 Rendering cycles

As described in the OpenXR Reference Guide [2], an OpenXR application is composed of different cycles as depicted in Figure 1.



Figure 1 - OpenXR application lifecycle [2]

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

2.2 Swapchains

Swapchains are a generic mechanism for computer systems to manage the generation and the display of images. As commonly defined “a swap chain (also swapchain) is a series of virtual framebuffers utilized by the graphics card and graphics API for frame rate stabilization and several other functions. The swap chain usually exists in graphics memory, but it can exist in system memory as well.” [3].

The OpenXR API allows an application to request the creation of swapchains using the xrCreateSwapchain function according to a specific format supported by the platform. The supported formats can be queried by the xrEnumerateSwapchainFormats function.

# 3 Proposal

3.1 General

The proposal is as follows :

* Add an arrow from the AR Scene Manager to the AR Runtime where frames are submitted.
* Add a Swapchains API seating on the AR runtime to allow submitting the images in the Swapchains.
* Add text describing the usage of Swapchains.

3.2 Proposed updated clause

**===== 1. CHANGE =====**

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 an AR glass**

 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.

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.

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.

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.

**===== 2. CHANGE =====**

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.

Figure 2 provides the technical architecture of the AR UE.



**Figure 2 - 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.

# References

1. S4-220760, MeCAR Permanent Document v2.0, 3GPP TSG SA WG4 119-e Meeting, 11th – 12th May 2022
2. OpenXR 1.0 Reference Guide, <https://www.khronos.org/files/openxr-10-reference-guide.pdf>
3. Wikipedia contributors, "Swap chain," *Wikipedia, The Free Encyclopedia,* <https://en.wikipedia.org/w/index.php?title=Swap_chain&oldid=1053599560> (accessed August 11, 2022).