**3GPPSA4#120-e S4-220967**

**E-meeting, 17 – 26 August 2022**

|  |
| --- |
| *CR-Form-v12.0* |
| **PSEUDO CHANGE REQUEST** |
|  |
|  | **26**.**806** | **CR** | pseudo | **rev** | **-** | **Current version:** | **0.2.1** |  |
|  |
| *For* [***HE******LP***](http://www.3gpp.org/3G_Specs/CRs.htm#_blank)*on using this form: comprehensive instructions can be found at* [*http://www.3gpp.org/Change-Requests*](http://www.3gpp.org/Change-Requests)*.* |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Proposed change affects:*** | UICC apps |  | ME |  | Radio Access Network |  | Core Network |  |

|  |
| --- |
|  |
| ***Title:***  | **[FS\_SmarTAR] Updates to Stand-Alone Architecture and Media Handling** |
|  |  |
| ***Source to WG:*** | Qualcomm Incorporated |
| ***Source to TSG:*** |  |
|  |  |
| ***Work item code:*** | FS\_SmarTAR |  | ***Date:*** | 11/08/2022 |
|  |  |  |  |  |
| ***Category:*** | **B** |  | ***Release:*** | Rel-18  |
|  | *Use one of the following categories:****F*** *(correction)****A*** *(mirror corresponding to a change in an earlier release)****B*** *(addition of feature),* ***C*** *(functional modification of feature)****D*** *(editorial modification)*Detailed explanations of the above categories canbe found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | *Use one of the following releases:Rel-10 (Release 10)Rel-11 (Release 11)Rel-12 (Release 12)**Rel-13 (Release 13)Rel-14 (Release 14)Rel-15 (Release 15)Rel-16 (Release 16)* *Rel-17 (Release 17)* *Rel-18 (Release 18)* |
|  |  |
| ***Reason for change:*** |  |
|  |  |
| ***Summary of change:*** |  |
|  |  |
| ***Consequences if not approved:*** |  |
|  |  |
| ***Clauses affected:*** |  |
|  |  |
|  | **Y** | **N** |  |  |
| ***Other specs*** |  |  |  Other core specifications  | TS/TR ... CR ...  |
| ***affected:*** |  |  |  Test specifications | TS/TR ... CR ...  |
| ***(show related CRs)*** |  |  |  O&M Specifications | TS/TR ... CR ...  |
|  |  |
| ***Other comments:*** | This pCR assumes that S4-220966 is agreed |
|  |  |
| ***This CR's revision history:*** |  |

**===== CHANGE =====**

## 4.4 Device Architectures for Tethered Glasses

### 4.4.1 General

Based on the guiding use case in clause 4.2.2 as well as the discussions in TR 26.998 [2], this clause identifies the architectures and media handling for different tethered AR glasses.

Looking at existing AR Glasses, based on the study in TR 26.998 [2] 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 4.4.1-1 is a picture providing an overview of an AR glass.

Hinge

SoC Media

Connectivity

Eye Tracking + Camera/Sensor Aggregator

Figure 4.4.1-1 - Overview of an AR glass

 Typical functions of such a AR glass consists of:

* Peripherals 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. Such 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 the main subject of discussion in this Technical Report.

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

Figure 4.4.1-2 provides an 5G AR Framework. In this context the AR/MR application is offered several functionalities on the device as well as connectivity options to create an XR experiences as defined in TR 26.998, clause 4.2, namely:

* XR Runtime: The XR Runtime is a device-resident software or firmware that implements a set of XR APIs to provide access to the underlying AR/MR hardware, including capability discovery, session management, input and sensors, composition, and other XR functions. An example for such APIs is provided by OpenXR.
* XR Scene Manager: A Scene Manager is a software component that is able to process a scene description and renders the corresponding 3D scene. To render the scene, the Scene Manager typically uses a Graphics Engine that may be accessed by well-specified APIs such as defined by Vulkan, OpenGL, Metal, DirectX, etc. Spatial audio is also handled by the Scene Manager based on a description of the audio scene. Other media types may be added as well.
* Media Access Functions: supports the application to access and stream media. For this purpose, a Media Access Function includes: media processing, codecs, content delivery protocols, content protection, QoS control, metrics collection and reporting, etc.
* 5G System: supports the AR/MR application to access the network through the 5G system, either directly or through the MAF.



Figure 4.4.1-2 – 5G AR Framework

In the following, two different approaches for tethering AR Glasses are identified, identifying how:

- Tethered Standalone AR Glasses: In this case, the AR Glass runs an AR/MR application that uses the capabilities of the glass to create a service. The AR glass is tethered to a 5G device and potentially uses the capabilities of the phone to support the application. For details refer to clause 4.4.2.

- Display AR Glasses: In this case, the AR Glass is tethered to a 5G device that includes the application and the XR functions. The 5G device runs the application that uses the 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 the 5G device/phone. For details refer to clause 4.4.3.

### 4.4.2 Tethered Standalone AR Glasses

Figure 4.4.2-1 provides the technical architecture of a typical stand-alone AR glass device. The AR Glass runs an AR/MR application that uses the capabilities of the glass to create a service. The AR glass is tethered to a 5G device and potentially uses the capabilities of the phone to support the application.



Figure 4.4.2-1 – Tethered Standalone AR glass-based device architecture

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 XR Runtime
* The XR Scene Manager

The AR/MR Application can communicate with those three components via dedicated APIs called the MAF-API, the XR Scene Manager API and the XR 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.

The XR runtime features several sensors and user controllers relevant for AR experiences such as cameras, microphones, speakers, display and generic user input. The XR Runtime typically also deals with the composition of primitive buffers that are mapped to the eye buffer display taking into account device characteristics as well as the latest pose information to apply late stage reprojection.

The XR scene manager is typically very lightweight and with no or very limited GPU capabilities. It maps raw media primitive buffers such as texture and depth information

Once the AR/MR application is running, the downlink media is accessed by the MAF in compressed form and then from then MAF to the AR Scene Manger in primitives. The device may also establish 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 remote device, it is typically compressed the data in order to facilitate the expected transmission over the network.

In order to analyse the use cases and tethering architectures in more details, the following assumptions are made:

* Video Playback and decoding: H.265 Main 10 Profile with maximum processing: up to 8,294,400 Macroblocks per second (corresponding to 8192x4320 @ 60fps)
* Video recording and encoding: H.265 Main 10 Profile with 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.
* Maximum number of combined encoding and decoding instances: 16 for video, audio tbd
* Audio capabilities that allow to encode several PCM signals with low-latency and to decode multiple audio PCM signals in parallel.
* The scene manager is very lightweight and passes through primitive buffers to be consumed by swap chains of the XR run-time. Swapchain images are typically 2D RGB.

Editor’s Note: More detailed assumptions on the rendering capabilities needs to be documented