**3GPP TSG SA WG4#123 S4-230813**

**Online, 17th – 22nd April 2023**

|  |
| --- |
| *CR-Form-v12.0* |
| **PSEUDO CHANGE REQUEST** |
|  |
|  | **26**.**806** | **CR** | pseudo | **rev** | **-** | **Current version:** | **1.2.1** |  |
|  |
| *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] Key Issue #5: Compute distribution across UE and network for tethered glasses** |
|  |  |
| ***Source to WG:*** | Qualcomm Incorporated |
| ***Source to TSG:*** |  |
|  |  |
| ***Work item code:*** | FS\_SmarTAR |  | ***Date:*** | 2023-05-16 |
|  |  |  |  |  |
| ***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 CR's revision history:*** |  |

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

## 6.5 Key Issue #5: Compute distribution across UE and network for tethered glasses

### 6.5.1 Introduction

In the tethered display AR Glasses context, the compute functions are distributed across the AR Glasses, as well as possibly the UE (phone) and the network. Even within the network the compute may be done in an edge or in the cloud. The decision how to distribute the compute across different network entities highly depends on the use case and application, the available capabilities in different network entities, the available network connections, economical reasons and possibly many other constraints. Generally, the situation may even change over time, for example due to changes in the application, varying network connections or load re-distribution. It is not expected that a specification will solve the distribution of the compute resources. However, what is essential is that the decision-making entity has as much static and dynamic information in order to make informed decisions.

This clause provides some background on different distribution scenarios. The main focus is the derivation of relevant static and dynamic status and capability information to establish proper workflows.

### 6.5.2 Background

Based on TS 26.119 [17], once a session is running and in focussed state a rendering loop is executed following Figure 6.5.2:

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 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 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:

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.



Figure 6.5.2-1 Rendering loop for visual data

Once this loop is running, the rendering can statically or dynamically be adjusted as long as operation northbound of the rendering loop is consistent.

### 6.5.3 Assumptions

According to TS 26.119 [17], media to be rendered and displayed by the XR device through the XR runtime is typically not available in uncompressed form on the device. In contrast, media is accessed using a 5G System, decoded in the device using media capabilities, and the decoded media is rendered to then be provided through swapchains to the XR Runtime as shown in Figure 6.5.3-1.



Figure 6.5.3-1 Media pipelines: Access, decoding and rendering

The rendering function is responsible to adapt the content to be presentable by the by the XR Runtime by making use of a rendering loop and using swapchains. The application configures pipeline of different processes, namely the media access, the decoding and the rendering.

For certain applications and scenes, the rendering capabilities on the glass, or the phone may not be sufficient in order to address the application requirements and pre-rendering is done in the network.

### 6.5.4 Problem Statement

In all considered architectures, the compute functions are distributed across the AR Glasses, as well as possibly the phone and the 5G network. Even within the network the compute may be done in an edge or in the cloud. The decision how to distribute the compute across different network entities highly depends on the use case and application, the available capabilities in different network entities, the available network connections, economical reasons and possibly many other constraints. Generally, the situation may even change over time, for example due to changes in the application, varying network connections or load re-distribution.

### 6.5.5 Potential Solutions

The solution is obvious, namely the distribution of processing functions across different entities. However, it is not expected that a specification will solve the distribution of the compute resources, but that a decision-making entity has sufficient static and dynamic information in order to make informed decisions.

The decision-making entity may for example be a Media Session Handler functions that configures the workflows. In order to do such configuration, the Media Session Handler needs to collect static and real-time information on the capabilities as well as real-time metrics from:

- tethering glass

- phone running the 5G System and Media Session Handler

- 5G Edge Server

- Cloud Server

Collected static and dynamic information includes, but is not limited to,

- link quality (bitrate, QoS),

- available encoding and decoding resources,

- available rendering capabilities,

- operational QoE metrics and logs

Reconfiguration of such workflows needs to be possible.

### 6.5.6 Conclusions

Based on the considerations, the Split Rendering architecture is expected to be extended to address:

* A workflow configuration management
* This workflow management collects static and real-time information.
* The workflow management is able to re-configure the rendering workflow.