**3GPP TSG-SA WG4 Meeting #122S4-230188**

**Athens, Greece, 20 – 24 February 2023**

**Source: Apple, Tencent, Samsung**

**Title: Pseudo-CR Correction for XR framework reference and privacy for TS 26.565**

**Spec: 3GPP TS 26.565 v0.0.1**

**Agenda item: 8.6**

**Document for: Agreement**

**1. Introduction**

This CR focuses on addressing two issues:

1. The split rendering architecture exchanges sensitive user information outside of the UE (e.g. between Split rendering EAS). It is pivotal that privacy and security of this sensitive data is analysed and ensured. A place holder note is proposed for now.
2. It has been agreed via S4-221624 that XR related specifications will not be restricted to a specific runtime framework like OpenXR. Specific text in draft v0.2.0 for TS 26.565 in S4-221507 creates ambiguity in this regard and is proposed to be updated to avoid this.

**2. Proposal**

It is proposed to integrate the marked up changes in updated draft TS 26.565.

\* \* \* First Change \* \* \* \*

# 5 Reference Architecture and Procedures

Editor’s Note: The terminology and reference-point naming convention need to be aligned in this document.

Editor’s Note: Privacy and security aspects of sensitive user data exchanged between UE and UE-external functions need to be considered.

\* \* \* Next Change \* \* \* \*

### 5.2.2 Call flow for Split Rendering session setup

The split rendering operation can be described by the call flow as depicted in the Figure 5.2.2.X:



Figure 5.2.2.X: High-level call flow for split rendering session setup.

The steps are:

1. The Presentation Engine discovers the split rendering EAS and sets up a connection to it. It provides information about its rendering capabilities and runtime configuration, e.g. that the OpenXR configuration may be used for this purpose.
2. In response, the split rendering EAS creates a description of the split rendering output and the input it expects to receive from the UE.
3. The Presentation Engine requests the buffer streams from the MAF, which in turn establishes a connection to the split rendering EAS to stream pose and retrieve split rendering buffers.
4. The Source Manager retrieves pose and user input from the XR runtime.
5. The Source Manager shares the pose predictions and user input actions with the split rendering EAS.
6. The split rendering EAS uses that information to render the frame.
7. The rendered frame is encoded and streamed down to the MAF.

\* \* \* Next Change \* \* \* \*

#### 8.2.2.2 Downlink Formats

The supported view configurations are:

* Mono: a single view
* Stereo: one view per eye

The following composition layers are supported:

* Projection: projection of the scene to a 2D plane using a perspective camera
* Quad: a 2d surface that is composed in the 3D space by the XR runtime
* Equirectangular: an equirectangular projection of the 3D space that is usually used to provide a background
* Cubemap: a set of 6 swapchain images that represent a projection of the 3D scene onto a cube

Each swapchain image will have the following properties:

* Format: RGB, RGB with Alpha (RGBA), and single-channel Depth formats with different precisions. RGB may be recovered from the coded YUV video stream. Depth information may be coded a separate video stream.
* Dimension: width and height of the swapchain image
* Mipmap: count of the level of detail of the swapchain image. The swapchain images maybe created at the UE side. Some Graphics Engines expect that the image dimensions are a power of 2.

For audio, the following formats are to be supported:

* Stereo audio mixed and binauralized based on the viewer’s current pose
* HOA audio mixed based on the viewer’s current position that extracted from the pose

\* \* \* Next Change \* \* \* \*

#### 8.2.2.3 Uplink Formats

The rendering process relies on the reception of pose predictions and user input. The pose information is formatted as follows:

* An array of multiple pose predictions
* Each pose prediction consists of a position and orientation component as a 3D (coordinates) and 4D (quaternion) vectors respectively.
* The prediction timestamp associated with the predicted pose
* An XR space for which the pose is created. If not present, this defaults to the viewer’s XR space.

\* \* \* End of Changes \* \* \* \*