**3GPP TSG- Meeting #**

**, , -**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.1* | | | | | | | | |
| **PSEUDO CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  |  | **CR** |  | **rev** |  | **Current version:** |  |  |
|  | | | | | | | | |
| *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:*** |  | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** |  | | | | | | | | | |
| ***Source to TSG:*** |  | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** |  | | | | |  | ***Date:*** | | |  |
|  |  | | | |  | |  | | |  |
| ***Category:*** |  |  | | | | | ***Release:*** | | |  |
|  | *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 can be found in 3GPP [TR 21.900](http://www.3gpp.org/ftp/Specs/html-info/21900.htm). | | | | | | | | *Use one of the following releases: Rel-8 (Release 8) Rel-9 (Release 9) Rel-10 (Release 10) Rel-11 (Release 11) … Rel-15 (Release 15) Rel-16 (Release 16) Rel-17 (Release 17) Rel-18 (Release 18)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | Completion of Technical Report | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | Add a paragraph tasks and issues related to multiple Media Decoders handling by the AR/MR application. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | Lack of tight integration between the AR/MR application and hardware decoding resource | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 4.4.1, 4.4.6 (new) | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **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:*** | |  | | | | | | | | |

## \*\*\* Start change 1 \*\*\*

[4.4.a] ISO/IEC 23090-13:2021 CD: “Information technology — Coded representation of immersive media — Part 13: Video Decoding Interface for Immersive Media”

## \*\*\* End change 1 \*\*\*

## \*\*\* Start change 2 \*\*\*

## 4.4 AR content formats and codecs

### 4.4.1 Overview

5G AR/MR application provider provides a 5G AR/MR service to 5G AR/MR application. A 5G AR/MR service consists of AR/MR content and description of supported processing by the 5G immersive service architecture. An AR/MR content is agnostic to a service architecture and consists of one or more AR/MR objects, which of each usually corresponds to an immersive media in clause 4.4.4. Delivery of an immersive media adaptive to device capability and network bandwidth can be described by a delivery manifest in clause 4.4.5. An AR/MR content consists of one or more AR objects, and may include their spatial and temporal compositions. A spatial and temporal composition of AR/MR objects can be described by a scene description in clause 4.4.2. Processing of AR/MR functions in 5GMS AS may require additional metadata in clause 4.4.3 to properly recognize user’s pose and surroundings.

Editor’s note) A further study is required on cascading of AR service entities.

- For example, whether an AR object can have another AR object as a component, whether an AR content can have another AR content as a component, and whether an AR service can have another AR service as a component.

AR/MR functions include encoding, decoding, rendering and compositing of AR/MR object, after which localization and correction is performed based on the user’s pose information.

STAR-based architecture has both basic AR functions and AR/MR functions on the device. EDGAR-based architecture has only basic AR functions on the device.

Since AR/MR functions are on-device for the STAR-based architecture, immersive media including 2D media can be considered as the input media for the architecture.

Examples of immersive media are 2D/3D media such as overlay graphics and drawing of instructions (UC#16 in Annex A.1), 3D media such as furniture, a house and an animated representation of 3D modeled person (UC#17 in Annex A.2), a photorealistic volumetric video of a person (UC#18 in Annex A.3), a 3D volumetric representation of conference participants (UC#19 in Annex A.4), 2D video, and volumetric information and simple textual overlays (UC#20 in Annex A.5).

For the EDGAR-based architecture, basic AR functions are on-device therefore 2D media and additional information (such as depth map) generated from immersive media renderer can be considered as the input media for basic AR functions. A rasterized and physically-based rendering (PBR) image is an example of 2D media.

A study into the existing technologies to be considered as inputs to each function and device type are identified and presented as a non-exclusive list below.

- several visual media representation formats were documented in clause 4.4.4.

- several delivery manifests were documented in clause 4.4.5.

- several scene description formats were documented in clause 4.4.2.

- metadata such as user pose information and camera information were documented in clause 4.4.3, respectively.

 Figure 4.4.1-1: Media Functions and formats

CoordinateCoor

changes

In order to integrate real-time media into AR scenes, a Media Access Function (MAF) provides the ability to access media and adds it to the AR scene. The MAF instantiates and manages Media Pipelines. A media pipeline typically handles content of an attribute/component of an object/mesh that is part of the scene graph. The media pipeline produces content in the format indicated by the scene description file. For real-time media, the formatted frame is then pushed into the circular buffer. Media Pipelines are typically highly optimized and customized for the type and format of media that is being fetched. Typically, for one scene, multiple media decoders of the same media type are needed to run in parallel. If the media decoders share the same hardware decoding platform on the UE, the MAF may also coordinate the different instances of media decoders to optimise the use of the hardware platform thus avoiding negative effects of resource competition or possible synchronization issues. MPEG-I Video Coding Interface (ISO/IEC 23090-13 [4.4.a]) is an example specification that can fulfil this task of coordination. More information available in clause 4.5.6. General considerations and challenges related to Media Decoder management is described in clause 4.4.6. Media Pipelines also maintain sync information (time and space) and passes that information as buffer metadata to the scene manager.

## \*\*\* End change 2 \*\*\*

## \*\*\* Start change 3 \*\*\*

### 4.4.6 Multiple Media Decoders management and coordination

The use of hardware video decoding platform is essential for the decoding of AR/MR content when it comes to power consumption, fast and scheduled decoding as well as battery usage. Modern hardware video decoding platform typically offer the capability to instantiate multiple decoders of the same media type at the same time and run multiple decoding instances in parallel. A typical example is the decoding of different components of the same AR/MR object, or the presentation of multiple objects in a scene. As a result, AR/MR application typically runs several decoder instances, in some cases using the same codec for different instances, in others different codecs for different streams. Note that this issue not only exists for video, but for any media type, in particular also for object-based audio. Under this high demand, there may be a resource competition and scheduled issues for the hardware decoding platform.

From an application perspective, there are different cases as well. There may exist cases for which even several applications are competing for the hardware decoding platform, for example an application renders a scene, but other applications provide overlays and notifications on top of the existing scene. A possible solution is to handle the coordination at the operating system level by setting priority to each application.

However, a single AR/MR application accessing and managing several decoding instances is a more typical and prominent case. It is thus important that the performance of the different decoder instances running are in line with the expectations and the needs of the AR/MR applications such that the AR/MR applications can optimise the usage of the hardware decoding platform when possible.

The first question from the AR/MR application point of view is to determine the number of decoder instances to instantiate. To this end, the AR/MR application may determine the number of AR/MR objects to be presented as well as the number of elementary streams contained in each AR/MR object. The hardware decoding platform is typically exposing a capability query API which lists the supported codec. This information enables the AR/MR application to calculate how many AR/MR objects can be simultaneously decoded and with which quality. In addition, there can be cases wherein different elementary streams from the same AR/MR object may be jointly decoded as part of a single elementary stream hence streamlining the rest of the pipeline by effectively decreasing the number of decoder instances and output buffers needed. When this is the case, the AR/MR application may instruct the hardware decoding platform to merge those input elementary streams.

At runtime, the AR/MR application expects the decoded frames for each AR/MR object to be ready at the same point in time so that further processing of this AR/MR object can happen without loss of frames or delay introduced due to buffering. However, the concurrent decoder instances may exhibit different performance in terms of decoding delay for each frame. Therefore, it is useful for the AR/MR application to be able to signal to the hardware video decoding platform that certain decoder instances form a group that should be treated collectively in terms of output synchronisation.

## \*\*\* End change 3 \*\*\*

## \*\*\* Start change 4 \*\*\*

### 4.5.6 MPEG-I Video Decoding Interface for Immersive Media

The aim of VDI (MPEG-I part 13) is to address the challenges for media applications to handle multiple decoder instances in parallel, especially in the case of immersive media. To this end, the scope of the VDI specification covers the interface between a media application and the Video Decoding Engine (VDE) sitting on the device.

Typically, hardware decoder are exposed via API to the application. Propeirtery API exist but also standardised one auch as Khronos OpenMax and Khronos Vulkan Video extension. However, those APIs allow the instantiation of video decoder instances independently from each other up to the point where the hardware decoding platform can no longer sustain the application requests, for instance due to lack of memory.

Extensions specified in MPEG-I VDI (ISO/IEC 23090-13) allow the AR/MR application to query the capacity to simultaneously decode multiple operation points (generally specified by profile, tier and levels). This allows a better predictability of what bitstreams can be decoded by the application.

Additionally, VDI also defines bitstream manipulation functions for the video codecs HEVC, VVC and EVC that enable the merging and the splitting of bitstreams. This aspect of elementary stream manipulation is covered by the so-called input formatting function in MPEGI VDI. This way, an application can adapt the number of the decoder needed when several input bitstreams are to be decoded to the extent the merging operations has been enabled by the proper encoding constraints.

## \*\*\* End change 4 \*\*\*