**3GPP TSG-S4 Meeting # 127 S4-240063**

**Sophia-Antipolis, FR, 29th January - 2nd February 2024**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.2* | | | | | | | | |
| **PSEUDO CHANGE REQUEST** | | | | | | | | |
|  | | | | | | | | |
|  | **26.565** | **CR** |  | **rev** |  | **Current version:** | **1.0.0** |  |
|  | | | | | | | | |
| *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 | **X** | Radio Access Network |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** | CR on suggested edits to the TS | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** | Qualcomm Inc. | | | | | | | | | |
| ***Source to TSG:*** | S4 | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** | SR\_MSE | | | | |  | ***Date:*** | | | 23rd January 2024 |
|  |  | | | |  | |  | | |  |
| ***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 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-16 (Release 16) Rel-17 (Release 17) Rel-18 (Release 18) Rel-19 (Release 19)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | This pCR proposes changes to TS26.565 to address a few issues and to replace redundant sections by references in the target specs. | | | | | | | | |
|  | |  | | | | | | | | |
| ***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:*** | |  | | | | | | | | |

|  |
| --- |
| **First Change** |

# 1 Scope

The present document defines a Media Service Enabler for Split Rendering according to the guidelines of TR26.857 [1]. The Split Rendering MSE covers functionality on the UE and on the Media AS. It also defines an API that is exposed to application developers on the UE to start and manage split rendering sessions.

|  |
| --- |
| **2nd Change** |

### 5.1.2 Client Architecture

The client architectural breakdown is based on the client architecture in TS26.119 [4] clause 5.1. The figure depicting the client architecture is replicated here as Figure 5.1.2-1 for convenience.

Une image contenant diagramme

Description générée automatiquement

Figure 5.1.2-1 - XR Baseline terminal architecture

The split rendering client consists of the following components:

* The Media Access Functions: allow for fetching and processing of the pre-rendered media in preparation of final display. The MAF is also responsible for the carriage of any metadata or local media to the split rendering server.
* The scene manager and thin Presentation Engine: is responsible for the negotiation of the split rendering session and the parsing of the description of the rendered media as provided by the SR server. It is also responsible for setting up and managing the XR session with the XR runtime.
* The XR source management is responsible for gathering timed metadata such as pose and action information and sending it to the SR server.

|  |
| --- |
| **3rd Change** |

#### 5.2.1.2 Client-driven procedures and call flows

Figure 5.2.1.2-1 demonstrates a call flow for setting up the split rendering by the client.



Figure 5.2.1.2-1: High-level call flow for initiating a split

Steps:

1. The Application Service Provider requests the SRF the provisioning a split management session.
2. The split management session is announced to the Application as part of the Service Access Information.
3. The Application requests a split of the client media functions from the SRC.
4. The SRC inquires the Media Session Handler about the client’s media capabilities.
5. The SRC and SRS negotiate on the acceptable capabilities for the device and agree on the split option.
6. The SRS starts the split rendering process.
7. The SRC provides the session information via the RTC-6 interface and requests the application of dynamic policy and subscription to network assistance.
8. The SRC establishes the WebRTC session.

9. The SRC informs the application that the split-rendering on edge is running.

10. The SRC sends uplink metadata, such as pose and action information.

11. The SRS sends the rendered media to the SRC.

|  |
| --- |
| **4th Change** |

## 7.3 Dynamic Policy and Network Assistance

Dynamic policy and network assistance may be provisioned by the Application Provider with the RTC AF. The allowed dynamic policies for the split rendering sessions of the application provider are communicated to the MSH in the client using the Configuration procedure.

Upon the creation of a new split rendering session and upon eligibility, the MSH shall use the Dynamic Policy API to request the allocation of network resources and charging policy to the session based on the information in the corresponding Provisioning session.

A policy template that is provisioned for split rendering should be associated with the split rendering configuration.



The MSH And the WebRTC Signaling Server shall support the dynamic policy API as defined in TS26.113.

The Application Provider may provision support for PDU Set marking. The SRS shall support the PDU Set marking and should support the End of Burst marking for the RTP streams that are generated by the Split Rendering Server.

|  |
| --- |
| **5th Change** |

## 7.4 Edge Resources

A split rendering application may use the procedures defined in TS26.512 [7] clause 7.10 to define an edge resource configuration to be used for split-rendering sessions. In this case:

* The eligibilityCriteria shall be present and shall have appRequest set to true.
* The easRequirements shall indicate “SR” as the easType and shall include “3gpp-sr” among the easFeatures. The serviceKpi shall be present and indicate the SRS processing and networking capabilities and requirements.

|  |
| --- |
| **6th Change** |

# 8 Split Rendering User Plane

# 8.1 General

The user plane for split rendering covers all traffic between the SRC and SRS, or the SRC and any other RTC AS. The common formats for split rendering are defined in this clause. Split rendering profiles may define additional user plane formats.

This clause illustrates the protocol stack for the User plane transport related to the signalling as well as the media delivery between SRC and SRS though RTC-4.

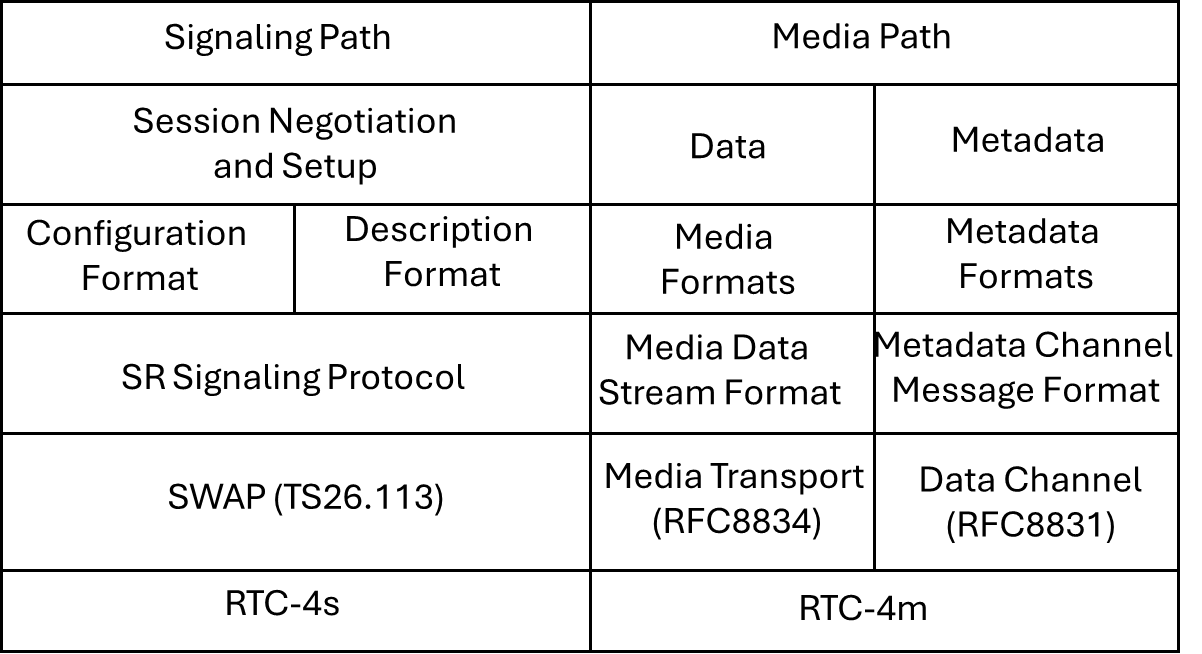


Figure 8.1-1 Split rendering protocol Stack

|  |
| --- |
| **7th Change** |

### 8.3.2 Metadata Formats

#### 8.3.2.1 General

Both SRC and SRS shall support the usage of the WebRTC data channel for the exchange of split rendering metadata. The WebRTC data channel shall declare “3gpp-sr” as the data channel sub-protocol. The message content format depends on the type of the message. The data channel sub-protocol is defined in clause 8.3.3.

Message types shall be unique identifiers in the URN format. This clause defines a set of message types and their formats. The messages are derived from the OpenXR API to ensure smooth operation with AR devices that support OpenXR. In case other XR APIs are used, mapping the message payload to the appropriate XR API structures shall be performed by the split rendering client.

Editor’s Note: This following sections will potentially reference the corresponding formats in 26.119.

#### 8.3.2.2 Pose Format



The pose format that is used by all split rendering profiles defined by this specification shall comply with the format defined in TS26.119 clause 6.2.2. The pose information shall be carried as part of the data channel messaging mechanism defined in clause 8.3.3 and shall be provided in JSON format. The message type shall be “urn:3gpp:split-rendering:v1:pose”.

#### 8.3.2.3 Action Format

The action information format that is used by all split rendering profiles defined by this specification shall comply with the format defined in TS26.119 clause 6.2.3. The action information shall be carried as part of the data channel messaging mechanism defined in clause 8.3.3 and shall be provided in JSON format. The message type shall be “urn:3gpp:split-rendering:v1:action”.



|  |
| --- |
| **8th Change** |

## 9.2 Client API

As described in clause 5.1.3, the SRC exposes an API over RTC-7 interface to the application. The SRC defines the following interface:

| **Method** | **Parameters** | | **State after Success** | **Description** |
| --- | --- | --- | --- | --- |
| **in** | **out** |
| SplitRenderer() | - appId  - aspId?  - settings? | - session handle | STATE\_READY | Creates a SplitRenderer object, which can subsequently be used to connect to an SRS and perform split rendering. |
| connect() | - settings?  - criteria? | - connection handler | STATE\_CONNECTED | Instructs the SRC to discover and connect to an SRS. |
| request() | - media configuration | -confirmation, notifications | N/A | Used by the client to request the application of dynamic policy to the split rendering media streams as described by the flow descriptors. It also requests the MSH to subscribe and relay any notifications about network assistance associated with this session. |
| disconnect() | - reason? |  | STATE\_DISCONNECTED | Terminates the connection to the SRS. |
| getMetrics() | - metrics | - metrics report | N/A | Retrieves a set of metric reports for the split rendering session that describe the quality of experience of the session. |

The application is able to subscribe to events related to the split rendering session by setting the corresponding event handler. The supported events are:

* State change: the state of the SR session has changed
* Error: an error has occurred during the split rendering session. The error is not severe enough to cause a state change to the STATE\_ERROR state.
* Quality change: the SRC has observed a change in the quality of the split rendering session. This may involve one or more SR metrics.

The Settings object shall contain the following information:

* Information about the desired rendering, e.g. choose to render on 2D device or on one of the available connected XR devices.

The criteria object may contain the following information:

* Requirements for latency and bitrate that are different from the ones in the provisioning,
* KPIs for the SRS instance, such as its graphics capabilities or current load.

|  |
| --- |
| **9th Change** |

### 9.3.1 General

This clause defines a set of metrics that are relevant to the operation of a split rendering session.

### 9.3.2 QoE Metrics Formats

#### 9.3.2.1 Timing Information Format

The timing information associated with the rendered frame is transmitted in the RTCP report block formats. This timing information is listed in the Table 8.3.Y.3-1.

The SRS may use the “QoE timing information” RTCP Extended Reports messages to transmit the timing information required for measuring the QoE metrics to an SRC. The RTCP report block formats for transmitting the QoE timing information is specified **in [TS 26.522].** SDP signalling required for negotiating the transmission of QoE metrics between the UE and the SRS is documented **in [TS 26.522]**.

The latency metrics that use the timing information defined in Table 8.3.Y.3-1 are detailed in the section 9.3.Table 8.3.Y.3-1: Timing information in the RTCP block formats.

|  |  |
| --- | --- |
| **Name** | **Description** |
| estimatedAtTime (ref. T1) | This time is defined in Table 8.2.2.2-1 - Pose Prediction Format.  This time is sent from the split rendering client.  This time is then received by the split rendering server and sent back to the split rendering client with the associated media frame. |
| sendingAtTime (ref. T1’) | This time is defined in Table 8.2.3-2 - Split Rendering Metadata Message Data Type  This time is sent from the split rendering client.  This time is then received by the split rendering server and sent back to the split rendering client with the associated media frame. |
| startToRenderAtTime (ref. T3) | The time when the renderer in the Split Rendering Server starts to render the associated media frame. |
| sceneUpdateTime (ref. T6) | The time when the Scene Manager starts to update the 3D scene graph according to the viewer pose and the user actions. |
| serverTransmitTime (ref. T5) | The time when the encoded rendered frame is transmitted from the split rendering server to the split rendering client. |

Editor’s note: the reference time used for the timing information is for FFS.

|  |
| --- |
| **10th Change** |

## 10.2 Privacy

Users of the split rendering MSE shall be aware that the application data and traffic are fully accessible to the SRS. The SRC shall ensure that the SRS used is trusted by the application provider, for example through the validation of the SRS’s X.509 certificates.

|  |
| --- |
| **11th Change** |

## C.1 Pixel Streaming Profiles

### C.1.1 Introduction

This Annex defines split rendering profiles to define requirements for SRC and SRS for different scenarios. At this stage the following two profiles are defined:

* 2D Pixel Streaming Profile in clause C.2 to support split rendering to 2D screens, devices of type 3 in TS 26.119 [4].
* 3D Pixel Streaming Profile in clause C.3 to support split rendering to MeCAR glasses to devices of type 1, 2, and 4 in TS 26.119 [4].

### C.1.2 2D Pixel Streaming Profile

#### C.1.2.1 Introduction

This profile defines required capabilities for UE-based SRC functionalities as network-side SRS capabilities to support split rendering to 2D screens.

#### C. 1.2.2 SRC Capabilities

#### C. 1.2.2.1 Overview

Requirements for UE-based SRC functionalities for following functions are defined in this clause:

- Media Decoding

- Media Encoding

- Metadata Formats

Editor’s Note: Additional Media Capabilities are for further study

The capabilities of the receiving UE are shared with the split rendering server prior to the start of the split rendering session.

Editor’s Note: Signaling of capabilities and configurations are for further study. For example, it would indicate that the output device is an HMD that supports 2 views and stereo audio.

#### C. 1.2.2.2 Media Capabilities

The SRC shall support the media capabilities of a device type 3 as defined in TS 26.119 [4], clause 10.4.

#### C. 1.2.2.3 Metadata Formats

**XR-Pose-Cap 1:** the SRC shall be able to retrieve one or more pose predictions for each view and for every frame to be rendered. The pose predication shall be formatted according to clause 8.2.2.2.

**XR-Pose-Cap 2:** the SRC shall be able to retrieve and collect the user actions that occurred during an identified time interval. The action information shall be formatted according to clause 8.2.2.3.

#### C.1.2.3 SRS Capabilities

##### C.1.2.3.1 Overview

Requirements for network-based SRS functionalities for following functions are defined in this clause:

- Media Encoding

- Media Decoding

- Metadata Formats

Editor’s Note: Additional Media Capabilities are for further study

The capabilities of the SRC are shared with the SRC prior to the start of the split rendering session.

Editor’s Note: Signaling of capabilities and configurations are for further study.

##### C.1.2.3.2 Video encoding

Editor’s Note: Video Encoding capabilities is for further study to match the SRC capabilities.

##### C.1.2.3.3 Audio and Speech encoding

Editor’s Note: Audio and Speech Encoding capabilities is for further study to match the SRC capabilities.

##### C.1.2.3.4 Video decoding

Editor’s Note: Video Decoding Capabilities are for further study

##### C.1.2.3.5 Audio and Speech decoding

Editor’s Note: Audio and Video Decoding Capabilities are for further study

##### C.1.2.3.6 Metadata Formats

Editor’s Note: Metadata capabilities are for further study to match the SRC capabilities.

### C.1.3 3D Pixel Streaming Profile

#### C.1.3.1 Introduction

This profile defines required capabilities for UE-based SRC functionalities as network-side SRS capabilities to support MeCAR devices.

#### C.1.3.2 SRC Capabilities

##### C.1.3.2.1 Overview

Requirements for UE-based SRC functionalities for following functions are defined in this clause:

- Media Decoding

- Media Encoding

- Metadata Formats

Editor’s Note: Additional Media Capabilities are for further study

The capabilities of the receiving UE are shared with the split rendering server prior to the start of the split rendering session.

Editor’s Note: Signaling of capabilities and configurations are for further study. For example, it would indicate that the output device is an HMD that supports 2 views and stereo audio.

##### C.1.3.2.2 Media Capabilities

The SRC shall support the media capabilities of a device type 1 as defined in TS 26.119 [4], clause 10.2.

If the device is a device type 2 as defined in TS 26.119 [4], clause 10.4, it shall also support the media capabilities of a device type 2 as defined in TS 26.119 [4], clause 10.3.

If the device is a device type 4 as defined in TS 26.119 [4], clause 10.5, it shall also support the media capabilities of a device type 2 as defined in TS 26.119 [4], clause 10.5.

##### C.1.3.2.3 Metadata Formats

**XR-Pose-Cap 1:** the SRC shall be able to retrieve one or more pose predictions for each view and for every frame to be rendered. The pose predication shall be formatted according to clause 8.2.2.2.

**XR-Pose-Cap 2:** the SRC shall be able to retrieve and collect the user actions that occurred during an identified time interval. The action information shall be formatted according to clause 8.2.2.3.

#### C.1.3.3 SRS Capabilities

##### C.1.3.3.1 Overview

Requirements for network-based SRS functionalities for following functions are defined in this clause:

- Media Encoding

- Media Decoding

- Metadata Formats

Editor’s Note: Additional Media Capabilities are for further study

The capabilities of the SRC are shared with the SRC prior to the start of the split rendering session.

Editor’s Note: Signaling of capabilities and configurations are for further study.

##### C.1.3.3.2 Video encoding

Editor’s Note: Video Encoding capabilities is for further study to match the SRC capabilities.

##### C.1.3.3.3 Audio and Speech encoding

Editor’s Note: Audio and Speech Encoding capabilities is for further study to match the SRC capabilities.

##### C.1.3.3.4 Video decoding

Editor’s Note: Video Decoding Capabilities are for further study

##### C.1.3.3.5 Audio and Speech decoding

Editor’s Note: Audio and Speech Decoding Capabilities are for further study

##### C.1.3.3.6 Metadata Formats

Editor’s Note: Metadata capabilities are for further study to match the SRC capabilities.

## C.1.4 Description of the Rendering Format for Pixel Streaming Profiles

#### C.1.4.1 General

In response to the Split Rendering Configuration message, the SRS shall reply with a description of the rendering format.

The rendering format description shall be a compliant glTF 2.0 [2] file. The file may include references to the buffer streams that contain the components of the rendered media.

Both SRS and SRC shall comply with the SD-Rendering-Ext1 capability as defined in TS26.119 [4].

In addition, both SRS and SRC shall support for referencing WebRTC RTP streams and data channels as described in [3].

An SRC that complies with the 3D Pixel Streaming profile shall support the 3GPP\_node\_prerendered extension as defined in C.4.2.

#### C.1.4.2 3D Pixel Streaming Profile-specific glTF Extension

The 3GPP\_node\_prerendered extension is an extension at the node level to describe that the corresponding node is accessible as a prerendered content. The 3GPP\_node\_prerendered extension should be associated with the root node of the scene. It constitutes an alternative representation of the node and all its children. As such, if present, if the client decides to use the pre-rendered representation, it shall completely ignore the mesh description of the node and its children nodes.

The 3GPP\_node\_prerendered supports multiple 2D video textures and audio sources that correspond to the rendered views and audio content.

The semantics of the 3GPP\_node\_prerendered are provided by the following table:

| **Name** | **Type** | **Usage** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| visual | Object | O | N/A | An object that describes the rendered visual components of the content. |
| audio | Object | O | N/A | An object that describes the rendered audio components of the content. |

The description of the visual object is provided in the following table:

| **Name** | **Type** | **Usage** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| visual\_configuration | enum | O | VIEW\_STEREO | An indication of the view configuration for the pre-rendered media. It can either be VIEW\_MONO or VIEW\_MONO. |
| Views | array(Object) | M |  | An array that describes the views of the prerendered content. |
| eye\_visibility | enum | M |  | The visibility of the current view. This can take one of the following values: “EYE\_LEFT”, EYE\_RIGHT”, “EYE\_BOTH”, or “EYE\_NONE”. EYE\_NONE is used for depth and transparency components. |
| composition\_layers | array(number) | M |  | An array of accessors identifiers that each corresponds to a composition layer of the parent view. |
| composition\_layer\_type | array(enum) | M |  | For each of the composition layers of the parent view, this indicates the type of that composition layer. The values should be provided in the same order as the composition\_layers. The allowed values are: “COMPOSITION\_LAYER\_PROJECTION”, “COMPOSITION\_LAYER\_QUAD”, “COMPOSITION\_LAYER\_EQUIRECTANGULAR”, “COMPOSITION\_LAYER\_CUBEMAP”, “COMPOSITION\_LAYER\_DEPTH”, and “COMPOSITION\_LAYER\_OCCUPANCY”. |

The description of the audio object in the prerendered media extension is provided in the following table:

| **Name** | **Type** | **Usage** | **Default** | **Description** |
| --- | --- | --- | --- | --- |
| type | enum | O | AUDIO\_STEREO | describes the format of the prerendered audio content. The type can take one of the following values: “AUDIO\_MONO”, “AUDIO\_STEREO”, and “AUDIO\_HOA”. |
| Components | array(number) | M |  | provides a list of the accessors that point to the media streams associated with rendered audio content. |

The JSON scheme for the 3GPP\_node\_prerendered is as follows:

|  |
| --- |
| {      "$schema" : "http://json-schema.org/draft-07/schema",      "title" : "3GPP\_node\_rendered",      "type" : "object",      "description": "glTF extension to described pre-rendered content",      "allOf": [ { "$ref": "glTFProperty.schema.json"} ],      "properties" : {          "visual": {              "$ref": "3GPP\_node\_rendered.visual.schema.json",              "description": "visual streamed buffers"          },          "audio": {              "$ref": "3GPP\_node\_rendered.audio.schema.json",              "description": "audio streamed buffers"          },          "extensions": {},          "extras": {}      },      "required": ["visual"]  }  {      "$schema" : "http://json-schema.org/draft-07/schema",      "title" : "3GPP\_node\_rendered.visual",      "type" : "object",      "description": "Object representing the visual rendered media",      "allOf": [ { "$ref": "glTFProperty.schema.json"} ],      "properties" : {          "view\_configuration": {              "type": "string",              "description": "the view configuration used for the session",              "gltf\_detailedDescription": "the view configuration used for the session",              "enum": ["VIEW\_MONO", "VIEW\_STEREO"]          },          "views": {              "type": "array",              "description": "array of layer view objects",              "gltf\_detailedDescription": "",              "items": {                  "$ref": "3GPP\_node\_rendered.visual.view.schema.json"              },              "minItems": 1          },          "extensions": {},          "extras": {}      },      "required": ["views"]  }  {      "$schema" : "http://json-schema.org/draft-07/schema",      "title" : "3GPP\_node\_rendered.visual.view",      "type" : "object",      "description": "A representation of a rendered view",      "allOf": [ { "$ref": "glTFProperty.schema.json"} ],      "properties" : {          "eye\_visibility": {              "type": "string",              "description": "the visibility of the current view",              "enum": ["EYE\_LEFT", "EYE\_RIGHT", "EYE\_BOTH", "EYE\_NONE"]          },          "composition\_layers": {              "type": "array",              "description": "array of timed accessors that carry the streamed buffers for each composition layer of the view",              "items": {                  "type": "integer"              },              "minItems": 1          },          "composition\_layer\_type": {              "type": "array",              "items": {                  "type": "string",                  "description": "the type of composition layer in the array of composition layers with the same array index",                  "gltf\_detailedDescription": "the type of composition layer in the array of composition layers with the same array index",                  "enum": ["COMPOSITION\_LAYER\_PROJECTION", "COMPOSITION\_LAYER\_QUAD", "COMPOSITION\_LAYER\_EQUIRECTANGULAR", "COMPOSITION\_LAYER\_CUBEMAP", "COMPOSITION\_LAYER\_DEPTH", "COMPOSITION\_LAYER\_OCCUPANCY"]              },              "minItems": 1          },          "extensions": {},          "extras": {}      },      "required": ["views"]  }  {      "$schema" : "http://json-schema.org/draft-07/schema",      "title" : "3GPP\_node\_rendered.audio",      "type" : "object",      "description": "Object representing the audio rendered media",      "allOf": [ { "$ref": "glTFProperty.schema.json"} ],      "properties" : {          "type": {              "type": "string",              "description": "the type of the rendered audio",              "gltf\_detailedDescription": "the type of the rendered audio",              "enum": ["AUDIO\_MONO", "AUDIO\_STEREO", "AUDIO\_HOA"],              "default": "AUDIO\_STEREO"          },          "components": {              "type": "array",              "description": "array of timed accessors to audio component buffers",              "items": {                  "type": "integer"              },              "minItems": 1          },          "extensions": {},          "extras": {}      },      "required": ["components"]  } |

|  |
| --- |
| **12th Change** |

### C.1.5 Profile Restrictions and Requirements

All Pixel Streaming profile are expected to be relocation intolerant and if using the 5G edge procedure shall set to

When the 2D Pixel Streaming profile is used, a policy template and a dynamic policy request may include the following QoS specifications, one for each of the components of the downlink streams:

* 1 QoS specification corresponding to the mono view.
* 1 QoS specification corresponding to one depth buffer stream associated with the mono view.
* 1 QoS specification corresponding to an occupancy/transparency buffer stream associated with the mono view.
* 1 QoS specification corresponding to an audio stream.

the 3D

* 2sone associated with the left and/or the right views
* 2 associated with the left and/or the right views