**3GPP TSG- Meeting #**

**Fukuoka, Japan, 19th May 2025 - 23rd May 2025 revision of S4-250523**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *CR-Form-v12.2* | | | | | | | | |
| **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 | **X** | Radio Access Network |  | Core Network | **X** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | |
| ***Title:*** |  | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Source to WG:*** |  | | | | | | | | | |
| ***Source to TSG:*** |  | | | | | | | | | |
|  |  | | | | | | | | | |
| ***Work item code:*** |  | | | | |  | ***Date:*** | | | ss |
|  |  | | | |  | |  | | |  |
| ***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-16 (Release 16) Rel-17 (Release 17) Rel-18 (Release 18) Rel-19 (Release 19)* | |
|  |  | | | | | | | | | |
| ***Reason for change:*** | | SR\_IMS has defined split rendering profiles that complement those defined in SR\_MSE and corresponding functions and procedures. The DELAY\_ADAPTIVE, feature is also relevant in SR\_MSE. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Summary of change:*** | | Add the processing delay adpatation feature:   1. Add call flow 2. Add information message format 3. Add extra configuration 4. Add metadata capability in the SR profiles 5. Add enabler to use the processing delay adpatation feature with Adaptive Split rendering profile. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Consequences if not approved:*** | | Missing features from SR\_IMS in SR\_MSE. | | | | | | | | |
|  | |  | | | | | | | | |
| ***Clauses affected:*** | | 5.2.3, 8.3.2, 8.4.2, C.2.1, and C.2.2 | | | | | | | | |
|  | |  | | | | | | | | |
|  | | **Y** | **N** |  | | | |  | | |
| ***Other specs*** | |  | **X** | Other core specifications | | | | TS/TR ... CR ... | | |
| ***affected:*** | |  | **X** | Test specifications | | | | TS/TR ... CR ... | | |
| ***(show related CRs)*** | |  | **X** | O&M Specifications | | | | TS/TR ... CR ... | | |
|  | |  | | | | | | | | |
| ***Other comments:*** | |  | | | | | | | | |
|  | |  | | | | | | | | |
| ***This CR's revision history:*** | |  | | | | | | | | |

FIRST change

### 5.2.3 Call flow for Processing Delay Adaptation

For processing delay adaptation, the general procedures and call flows in clause 5.2.2 applies with the following additions and modifications.

- The SRS and SRC agree on the QoE metrics considered for delay adaptation. That may contain all or a subset of the QoE latency metrics (e.g., poseToRenderToPhoton, roundtripInteractionDelay) negotiated in the configuration message in clause 8.4.2.3.

- In the rendering loop, The SRC measures and collects the QoE metrics considered for delay adaptation. The SRC may send periodically the in-band QoE metrics to the SRS when the measured QoE metrics goes out of the target delay range.

Figure 5.2.3-1 illustrates a high-level call flow set up and operation for a split rendering session which supports the processing delay adaptation profile.



Figure 5.2.3-1: High level call flows for Processing Delay Adaptation Profile

The steps are:

Steps 0 to 4 are as described in clause 5.2.2.

5. The Source Manager shares the pose predictions and user input actions. In addition, the metadata may include a in-band QoE metrics message to the SRS.

6a The SRS may adjust the processing delay based on the in-band QoE metrics message.

NOTE: For example, the SRS may change the LoD of the objects that are part of the scene for the delay adaptation.

6b The SRS renders the frame.

7 to 10: As described in clause 5.2.2.

Next 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.

#### 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 TS 26.119 [4] clause 12.2. 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 TS 26.119 [4] clause 12.3. 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”.

#### 8.3.2.4 In-band QoE Metrics Format

When processing delay adaptation procedure is enabled, the SRC checks for the QoE metrics being monitored whether the measured delays are within the target delay range or not. When a measured delay is outside the target delay range for a QoE metric, the SRC reports the measured QoE metrics in-band based on the configured periodicity.

NOTE: When the target range values for a QoE metric is delimited by the positiveCrossingThresholds and the negativeCrossingThresholds properties provided to the SRC in the clientMetricsReporting‌Configurations resource of ServiceAccessInformation as defined in clause 9.2.3.1 of TS 26.510 [9].

The in-band QoE metrics format shall comply with the format defined in Table 8.3.2.4-1. The in-band QoE metrics 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:daqoe:information”.

Table 8.3.2.4-1 Message format for In-band QoE Metrics

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Cardinality | Description |
| id | string | 1..1 | A unique identifier of the message in the scope of the IMS-based split rendering session. |
| type | string | 1..1 | urn:3gpp:split-rendering:v1:daqoe:information |
| message | object | 1..1 | Message content |
| qoeMetrics | array | 1..1 | An array of the QoE metrics measurements. This qoeMetrics array may contain all or a subset of the QoE metrics negotiated in the configuration message in clause 8.4.2.3. |
| qoeMetricId | string | 1..1 | A unique identifier of the QoE metric within the scope of the split rendering session. |
| delayValue | number | 1..1 | The measured delay value of that QoE metric. |

Next change

### 8.4.2 Split Rendering Configuration Format

#### 8.4.2.1 Introduction

The Split Rendering client establishes an XR session locally based on the device configuration and user selection. The SR client defines the view configuration (e.g. mono or stereo views), the projection format (such as projection, equirectangular, quad, or cubemap), the swap chain image configuration, etc.

In addition, XR space and action configurations are negotiated between the SR client and server. This includes defining common XR spaces and defining and selecting actions and action sets.

The format is extensible to support the exchange of additional/future configuration information.

#### 8.4.2.2 Split Rendering Configuration Format

The session configuration information shall be in JSON format. It shall have the following format:

Table 8.4.2.2-1 Split Rendering Configuration Format

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Cardinality** | **Description** |
| splitRenderingFlags | array(SR\_CONFIG\_FLAGS) | 0..1 | Provides a set of flags to activate/deactivate selected split rendering configurations/functions. The defined SR\_CONFIG\_FLAGS are:  - FLAG\_ALPHA\_BLENDING  - FLAG\_DEPTH\_COMPOSITION  - FLAG\_EYE\_GAZE\_TRACKING  - FLAG\_DELAY\_ADAPTATION |
| splitRenderingProfile | array(URI) | 0..1 | A list of supported split-rendering profile identifiers on the UE. The profile identifiers are listed in Annex C for each profile. |
| deviceCapabilities | Object | 0..1 | Device capabilities as defined in TS 26.119 [4], clause 6.1. |
| spaceConfiguration | Object | 0..1 | The space configuration is typically sent by the split rendering server to the split rendering client. Upon reception of this information, the SR client uses this information to create the reference and action spaces as well as to agree on common identifiers for the XR spaces. |
| referenceSpaces | Array | 0..1 | An array of reference spaces and their identifiers. |
| id | number | 1..1 | A unique identifier of the XR space in the context of the split rendering session. |
| refSpace | enum | 1..1 | One of the defined reference spaces in OpenXR. These may be: XR\_REFERENCE\_SPACE\_TYPE\_VIEW, XR\_REFERENCE\_SPACE\_TYPE\_LOCAL, or XR\_REFERENCE\_SPACE\_TYPE\_STAGE. |
| actionSpaces | Array | 0..1 | An array of action spaces that need to be defined by the split rendering client in the XR session. |
| id | number | 1..1 | A unique identifier of the XR space in the context of the split rendering session. |
| actionId | number | 1..1 | Provides the unique identifier of the action. |
| subactionPath | string | 1..1 | The subaction path identifies the action, which can then be mapped by the XR runtime to user input modalities. |
| initialPose | Pose | 0..1 | Provides the initial pose of the new XR space’s origin. |
| viewConfiguration | Object | 0..1 | Conveys the view configuration that is configured for the XR session. |
| type | Enum | 1..1 | The type indicates the view configuration. Defined values are MONO and STEREO. Other values may be added. |
| width | number | 1..1 | The recommended width of the swapchain image. |
| height | number | 1..1 | The recommended height of the swapchain image. |
| compositionLayer | string | 1..1 | An identifier of the selected composition layer. |
| minPoseInterval | number | 0..1 | The minimum time interval between two consecutive pose information instances sent to the network, in milliseconds. |
| fovs | Array | 0..1 | An array that provides a list of the field of views (FoV) associated with each view. |
| fov | Object | 1..n | Indicates the four sides of the field of view used for the projection of the corresponding XR view. The number of views n is determined by the *type* enum of the *viewConfiguration*. Both the *viewPoses* in the Pose Format and the *fovs* arrays shall be ordered in a consistent way (i.e., a same index can be used to retrieve the view pose and the related FoV information). |
| angleLeft | number | 1..1 | The angle of the left side of the field of view. For a symmetric field of view this value is negative. |
| angleRight | number | 1..1 | The angle of the right side of the field of view. |
| angleUp | number | 1..1 | The angle of the top part of the field of view. |
| angleDown | number | 1..1 | The angle of the bottom part of the field of view. For a symmetric field of view this value is negative. |
| environmentBlendMode | enum | 1..1 | The type indicates the environment blend mode configuration. Defined values are OPAQUE, ADDITIVE and ALPHA\_BLEND. Other values may be added. |
| actionConfiguration | Array | 0..1 | This contains a list of the actions that are to be defined by the SR client. |
| action | Object | 1..n | A definition of a single action object. |
| id | number | 1..1 | A unique identifier of the action. |
| actionType | enum | 1..1 | The type of the action state. This can be a Boolean, float, vector2, pose, vibration output, etc. |
| subactionPaths | string | 1..n | An array of subaction paths associated with this action. The split rendering client will provide the state of all defined sub-action paths. |
| inbandReporting | object | 0..1 | An object containing the configuration for the requested in-band QoE reporting. When inbandReporting object is present the SRC shall support in-band reporting of QoE metrics as specified in clause 8.3.2.4. |
| qoeMetrics | array(object) | 1..1 | An array of the QoE metrics for which delay adaptation is considered. This qoeMetrics array may contain all or a subset of the QoE latency metrics negotiated in the Metrics reporting configuration defined in clause 8.11.3 of TS 26.510 [9]. |
| qoeMetricId | string | 1..1 | A unique identifier of the QoE metric within the scope of the split rendering session. The name of that QoE metric is chosen as unique ID, this name should be consistent with the name provided in the metrics reporting configuration as defined in clause C.1 of TS 26.113 [6]. |
| periodicity | string | 1..1 | The periodicity of the in-band metric reporting for that QoE metric. This periodicity value shall be less than or equal to the samplingPeriod value of the MetricsReportingConfiguration defined in clause 8.11.3 of TS 26.510 [9]. |
| extraConfigurations | Object | 0..1 | A placeholder for addition configuration information. |

Next change

# C.2 Adaptive Split Rendering Profile

## C.2.1 Introduction

This profile defines procedures and requirements for SRS and SRC to support split rendering features beyond a remote rendering paradigm.

Adaptive split rendering profile allows the SRC to render some objects of a scene locally and the rendering split can be adapted between the SRS and SRC during a session. The adaptation of the rendering split may be triggered either by the SRS or the SRC to maintain a consistent QoE of the SR session or to accommodate changes in operating conditions. The triggers may be, for example, channel conditions, SRC or SRS conditions or defined by the application provider.

To successfully render two parts of a scene separately in a split fashion, additional aspects of the rendering process need to be considered. Two basic requirements are maintaining a coherent state of the scene between the SRS and SRC and awareness of rendering split. Another requirement is seamless composition and display of the media rendered by the SRS and SRC into a frame to be displayed.

The processing delay adaptation procedure can be used with the Adaptive split rendering profile. The processing delay adaptation procedure may include adjusting various round-trip delays between the SRC and the SRS during a split rendering session. The delay adaptation allows the SRS to adjust the rendering task delay to maintain a consistent round-trip delay (e.g., pose-to-render-to-photon or roundtrip interaction delay).

Next change

## C.2.2 Procedures and Call Flows

For adaptive split rendering, the general procedures and call flows in clause 5.2 are followed with the following additions and modifications.

- The SRS and SRC should share a scene description. The implementation details may vary. The application provider may decide whether to provide identical scene descriptions to the SRS and the SRC or whether to provide a truncated version of the scene description to the SRC.

Note: The Application Service Provider may provide the scene description resource to the SRS and SRC, for example, via M8 to SRC and via M2 to SRS.

- The SRS and SRC agree on an initial rendering split during session negotiation and the states to be synchronized, for example, in Step 5 of the procedure in clause 5.2.1.2.

- The initial rendering split and states to be synchronized are indicated in the SR configuration.

- In the rendering loop, exchange of split adaptation messages and state synchronization messages between the SRS and SRC is supported.

- When the processing delay adaptation procedure is used, the SRC shall send in-band QoE metrics to the SRS.

Figure C.2.2-1 illustrates a high level call flow set up and operation for a split rendering session which supports the adaptive split rendering profile.



Figure C.2.2.‑1: High level call flows for Adaptive Split Rendering Profile

The steps are:

0. In this optional step the SRC and the SRS acquire scene description of the scene to be rendered during the split rendering session. The actual implementation of delivery of the scene description by to the SRC and SRS is up to the application provider.

1. The Presentation Engine discovers the split rendering server and sets up a connection to it. It provides information about its rendering capabilities and the XR runtime configuration, e.g the OpenXR configuration may be used for this purpose. States to be synchronized and the initial rendering split is negotiated during this step.

2. In response, the split rendering server 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 server to stream pose and retrieve split rendering buffers.

4. The Source Manager retrieves pose and user input from the XR runtime and state changes in negotiated states and possible requests from the Scene Manager.

5. The Source Manager shares the pose predictions and user input actions , state changes and possible split adaptation messages with the split rendering server.

6. a. The split rendering server uses that information to, update states, render the frame and possibly update the split.

b. The Scene Manager update states, renders a frame and possibly updates the split.

7. a. The rendered frame is encoded and streamed to the MAF.

b. Possible split adaptation and state change messages are shared with the presentation engine,

8. The received media frames decoded and processed,

9. The raw buffer frames are passed to the Scene Manager, this includes the frames received from the SRS and the frames rendered locally by the PE,

10. The scene manager prepares composition layers and their corresponding swapchain images.

11. The swapchain images are forwarded to the XR runtime for composition and rendering,12. The frames are composed and displayed.

The final composition of a frame from media received from the SRS and locally rendered objects depends on the application logic. Implementation guidelines in C.2.7 provide a simple example.

End of changes