**3GPP TSG-SA WG4 Meeting #129-e** **S4-241533**

**Online, 19 – 23 August 2024**

|  |
| --- |
| *CR-Form-v12.2* |
| **CHANGE REQUEST** |
|  |
|  | **TS 26.565** | **CR** | **0001** | **rev** |  | **Current version:** | **18.0.0** |  |
|  |
| *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:***  | Editorial Corrections to TS 26.565 |
|  |  |
| ***Source to WG:*** | Nokia |
| ***Source to TSG:*** | S4 |
|  |  |
| ***Work item code:*** | TEI18, SR\_MSE |  | ***Date:*** | 2024-08-13  |
|  |  |  |  |  |
| ***Category:*** | D |  | ***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-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:*** | Several editorial modifications and removal of an unresolved EN |
|  |  |
| ***Summary of change:*** | * Editorial improvements throughout the specification
 |
|  |  |
| ***Consequences if not approved:*** | The specification has editorial bugs and an unresolved EN. |
|  |  |
| ***Clauses affected:*** | 2, 5.1.2, 5.1.3, 5.2.1.1, 7.2, 8.5, 9.2, 9.3.2.1, 9.3.5.1, A.1, C.1.2.3.2, C.1.2.3.3, C.2.4, C.2.4.2, C.2.4.4, C.2.7 |
|  |  |
|  | **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:*** |  |

**CHANGE 1**

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non‑specific.

- For a specific reference, subsequent revisions do not apply.

- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 26.857: "5G Media Service Enablers".

[2] ISO/IEC 12113:2022, Information technology, Runtime 3D asset delivery format, Khronos glTF 2.0

[3] ISO/IEC 23090-14: Information technology — Coded representation of immersive media — Part 14: Scene Description for MPEG Media.

[4] 3GPP TS 26.119, Media Capabilities for Augmented Reality

[5] 3GPP TS 26.506, 5G Real-time Media Communication Architecture (Stage 2)

[6] 3GPP TS 26.113, Real-Time Media Communication; Protocols and APIs

[7] 3GPP TS 26.512, 5G Media Streaming (5GMS); Protocols

[8] 3GPP TS 26.522, 5G Real-time Media Transport Protocol Configurations

[9] 3GPP TS 26.510, Media Delivery: interactions and APIs for provisioning and media session handling

[10] Khronos, The OpenXR API, <https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html>

[11] W3C, WebXR Device API, [WebXR Device API (immersive-web.github.io)](https://immersive-web.github.io/webxr/)

[12] Khronos, WebGL Specification 1.0, [WebGL Specification (khronos.org)](https://registry.khronos.org/webgl/specs/latest/1.0/)

[13] W3C, Web Audio API, [Web Audio API (w3.org)](https://www.w3.org/TR/webaudio/)

[14] 3GPP TS23.501, System architecture for the 5G System (5GS).

[15] 3GPP TS23.503, 5G; Policy and charging control framework for the 5G System (5GS).

[16] 3GPP TS26.857, 5G Media Service Enablers.

[17] 3GPP TS 26.247: "Transparent end-to-end Packet-switched Streaming Services (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)".

[18] 3GPP TS 26.501, 5G Media Streaming (5GMS); General description and architecture.

**CHANGE 2**

### 5.1.2 Client Architecture

The client architectural breakdown is based on the client architecture in TS 26.119 [4] clause 5.1. The figure depicting the client architecture is replicated here as Figure 5.1.2-1 for convenience.



Figure 5.1.2-1 - XR Baseline terminal architecture

The split rendering client consists of the following components:

- The Media Access Functions: allows 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 SRS. 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 SRS.

- XR Runtime: Set of functions provided by  XR Device to the  XR Application to create  XR experiences.

**CHANGE 3**

5.1.3 End-to-End Architecture

 ****

**Figure 5.1.3-1 – Split management architecture**

As shown in Figure 5.1.3-1:

1. The Media Application Provider (MAP) provisions the split-rendering through M1.

2. In the use cases in which the MAP is involved in the media delivery, the M2 interface is used for this purpose.

3. The communication between Media AF and SRS is through M3. This interface is out of the scope of this document. This interface may for instance include the EDGE-3 interface.

4. The signaling as well as the media delivery between SRC and SRS is though M4.

5. The Media AF may provide SR-related information to the Media Session Handler (MSH) through the M5 interface, as defined in TS 26.510 [9].

6. The SRC in the UE discovers the application through M11 and handles the XR runtime.

7. The SRC discovers the client media capabilities through the M7 interface. This interface is out of the scope of this document.

8. The 5G Application and MAP interact through M8. This interface is out of the scope of this document.

**CHANGE 4**

#### 5.2.1.1 Call flow for edge server and split rendering session setup

Figure 5.2.1.1-1 demonstrates a general call flow for split-rendering.



Figure 5.2.1-1: High-level call flow for split-rendering

Steps:

1. In this optional step, the Media Application Provider requests and sets up the edge server(s) used for the split-rendering as described in TS 26.506 [5] clauses 6.1 or 6.2. The Media Application Provider may use any other method to allocation edge servers, or leave it to the MNO to set up appropriate edge servers to run the split-rendering process.

2. The Media Application Provider provisions the split-rendering session using M1 and M3, as defined in call flow of clauses 5.2.1.1. If the edge servers were provisioned in step 1, the edge servers ids are provided in this session to employ them for split-rendering.

NOTE: In the case of the client-driven edge management (TS 26.501 [18], clause 8.1), only the client-driven split-rendering (5.2.1.1) is applicable.

3. The split-rendering session is set up according to clause 5.2.2.

**CHANGE 5**

7.2 Provisioning

A Media Application Provider that wishes to offer applications using split rendering shall use the procedures and data models defined in TS 26.510 [9] clauses 6 and 8 to create a Provisioning Session with the Media AF.

The ProvisioningSessionType shall be set to “BIDIRECTIONAL”.

The aspId shall be configured and shall be a unique identifier for the Media Application Provider that offers split rendering.

The externalApplicationId shall be a URN that uniquely identifies the application and shall be terminated by the sub-string “+3gpp-sr”. An example is as follows: “urn:com:example:game+3gpp-sr”.

**CHANGE 6**

8.5 Split Rendering Transport Protocols

Split Rendering shall use WebRTC for the real-time transport of the rendered media. The RTP restrictions for WebRTC as specified in RFC8834 shall apply. The usage of the WebRTC data channel shall be in accordance with RFC8831.

The RTP Header Extension for PDU Set Marking specified in clause 4.2 of TS 26.522 [8] and RTP Header Extension for XR Pose specified in clause 4.3 of TS 26.522 [8] should be used for transmitting PDU set marking and XR pose information respectively.

**CHANGE 7**

9.2 Client API

As described in clause 5.1.3, the SRC exposes an API over M7 interface to the application. The SRC defines the following interface:

**Table 9.2-1 Split Rendering Client API**

| **Method** | **Parameters** | **State after Success** | **Description** |
| --- | --- | --- | --- |
| **in** | **out** |
| SplitRenderer() | - appId- aspId- externalServiceId- preferences | - srSessionId | STATE\_PROCESSING | Creates a SplitRenderer instance, representing the SRC, which can subsequently be used to connect to an SRS and perform split rendering. |
| getState() | - srSessionId | - state | N/A | Returns the current state of the SRC. Possible states are: STATE\_IDLE, STATE\_PROCESSING, STATE\_READY, STATE\_RUNNING, STATE\_STOPPED. |
| getConfiguration() | - srSessionId | - configuration | N/A | Allows the application to query the current configuration of the split rendering session.  |
| start() | - srSessionId | - boolean | STATE\_RUNNING | Instructs the SRC to discover and connect to an SRS.If current state is not STATE\_READY, the connection will fail. |
| stop() | - srSessionId- reason |  | STATE\_STOPPED | Terminates the connection to the SRS. |
| release() | - srSessionId |  | STATE\_IDLE | Releases all resources associated with the Split Rendering session. |
| getMetrics() | - srSessionId- 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 Preferences 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.

The parameters are defined as follows:

- aspId: a string that holds an identifier of the application service provider. The value is provisioned by the application service provider as defined in TS 26.510 [9].

- appId: a string that holds an identifier of the application. This value is provisioned by the application service provider as defined in TS 26.510 [9].

- externalServiceId: An identifier assigned by the Service Application Provider and shared with the SRC over M8 as defined in TS 26.510 [9].

- preferences: the preferences object carries parameters about the user’s current preferences. These include the preferred display configuration, e.g. 2D display, HMD, etc. It may also include information about quality versus latency preferences.

- configuration: the configuration object stores information about the currently active configuration for the session. It carries the same information as described in clause 8.4.2.2.

- srSessionId: the srSessionId is a unique identifier of the split rendering session at the SRC.

- reason: the reason for termination of the current session between SRS and SRC.

- metrics: the metrics and metrics report objects provide the current status of a selected set of metrics that pertain to the current split rendering session. The format should follow clause 7.5.

**CHANGE 8**

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 9.3.22-1.

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 format for transmitting the QoE timing information is specified in TS 26.522 [8]. SDP signalling required for negotiating the transmission of QoE metrics between the UE and the SRS is documentedin TS 26.522 [8].

The latency metrics that use the timing information defined in Table 9.3.2-1 are detailed in the clause 9.3.2.2.

**CHANGE 9**

9.3.5.1 General

The quality metrics report follows the XML-based report format defined in clause 9.3.5.2.

The MIME type of an XML-formatted QoE report shall be "application/3gprtc-qoe-report+xml".

The metrics reporting protocol is as defined in clause 9.5.3 of TS 26.510 [9]. Split rendering UEs shall use the above MIME content type. The metrics report format is defined in the following sub clause.

**CHANGE 10**

A.1 Guidelines for Application Developers

Application developers may use the SR\_MSE enabler as an SDK for developing applications that benefit from Split Rendering.

The SDK may be accessible through an API that conforms to the API definition in clause 9.2.

Application developers should implement monitoring of the split rendering session quality in their applications and always be aware that not all functionality described in this specification is always available for all split rendering sessions. It is then up to the application to decide whether the usage of split rendering is acceptable or not.

**CHANGE 11**

C.1.2.3.2 Video encoding

The SRS shall at least be able to support the encoding of video that complies to the capabilities in clause 10.4.3 of TS 26.119 [4].

C.1.2.3.3 Audio and Speech encoding

The SRS shall at least be able to support the encoding of audio that complies to the capabilities in clause 10.4.4 of TS 26.119 [4].

 **CHANGE 12**

C.2.4 SRC Capabilities

The adaptive split rendering profile may be used in monoscopic mode or stereoscopic mode. In monoscopic mode, the SRC receives video corresponding to a single view. This mode supports split rendering to 2D screens, devices of type 3 in TS 26.119 [4].

In stereoscopic mode, the SRC receives video corresponding to two views, one for each eye. This mode supports split rendering to stereoscopic screens, devices of type 1, 2, 4 in TS 26.119 [4].

**CHANGE 13**

C.2.4.2 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 prediction shall be formatted according to clause 8.3.2.2.

**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 8.3.2.3.

**XR-ObjId-Cap 1**: the SRC shall be able to receive, retrieve and collect identifiers of objects in a scene being rendered by the SRC in a split rendering session during an identified time interval. The state information shall be formatted according to clause C.2.3.2

 **XR-ObjState-Cap 1:** the SRC shall be able to receive, retrieve and collect state changes in identified objects in a scene being rendered in a split rendering session during an identified time interval. The state information shall be formatted according to clause C.2.3.3

**CHANGE 14**

C.2.4.4 Scene Processing and Rendering Capabilities

The SRC shall have the following minimum scene processing capabilities:

- the ***SD-Rendering-gltf-core*** scene processing capabilities defined in clause 9.2 of TS 26.119 [4].

**SD-Rendering-gltf-core** enables basic compatibility of an SRC with the adaptive split rendering profile for simple use cases, where the SRC does minimal local rendering and adaptability of rendering split is minimal. An example of such a limited scenario may be a cloud gaming use case where the application provider isolates a small subgraph of the complex game scene to be rendered by the SRC and shares the subgraph with the SRC. The subgraph may contain only the assets (mesh and textures) related to a user’s character and controller to allow the SRC to render these objects locally to mask motion to photon to render latency. More advanced use cases of adaptive split rendering place higher scene processing capabilities on the SRC.

The SRC should have the following scene processing capabilities:

- the ***SD-Rendering-gltf-Ext1*** scene processing capabilities defined in clause 9.2 of TS 26.119 [4].

In addition to the above specified scene processing capabilities, depending on the device type, the SRC shall have scene capabilities defined for each device type in clause 10 of TS 26.119 [4].

**CHANGE 15**

C.2.7 Extension to Client API Functions

The SRC should perform adaptive split management which may be based on metrics reports of an ongoing split rendering session, scene being rendered and UE operating conditions. For adaptive split rendering, the SRC exposes functions to load and update scene description resources. The SRC may also expose functions to an application to allow application developers to deploy custom logic for split management.

| **Method** | **Parameters** | **State after Success** | **Description** |
| --- | --- | --- | --- |
| **in** | **out** |  |  |
| setScene() | - srSessionId -scene description resource | -status | N/A | The application requests the SRC to load a scene description resource for rendering in the split rendering session. |
| updateScene() | - srSessionId -scene description resource | -status | N/A | The application requests the SRC to update a scene description resource being rendered in the split rendering session. |

**End of Changes**