|  |
| --- |
| 3GPP TS 26.565 V0.4.0 (2023-04) |
| Technical Specification |
| 3rd Generation Partnership Project;Technical Specification Group Services and System Aspects;Split Rendering Media Service Enabler;(Release 18) |
|   |
|  |  |
|  |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP.The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented.This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification.Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. |

|  |
| --- |
|  |
| ***3GPP***Postal address3GPP support office address650 Route des Lucioles - Sophia AntipolisValbonne - FRANCETel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16Internethttp://www.3gpp.org |
| ***Copyright Notification***No part may be reproduced except as authorized by written permission.The copyright and the foregoing restriction extend to reproduction in all media.© 2022, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).All rights reserved.UMTS™ is a Trade Mark of ETSI registered for the benefit of its members3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational PartnersLTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational PartnersGSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 5

Introduction 6

1 Scope 7

2 References 7

3 Definitions of terms, symbols and abbreviations 7

3.1 Terms 7

3.2 Symbols 7

3.3 Abbreviations 7

4 General 8

4.1 Overview 8

4.2 Typical Use Cases 8

5 Reference Architecture and Procedures 8

5.1 Reference Architecture 8

5.1.1 Introduction 8

5.1.2 Client Architecture 8

5.1.3 End-to-End Architecture 9

5.1.5 User Plane Architecture 10

5.2 Procedures and Call Flows 10

5.2.1 Call flow for Split Rendering instance discovery 10

5.2.1.1 Call flow for edge server and split rendering session setup 10

5.2.1.2 Client-driven procedures and call flows 11

5.2.1.3 Network-driven procedures and call flows 12

5.2.2 Call flow for Split Rendering session setup 13

6 Prerequisites 14

6.1 Requirements on 5G System 14

6.2 Requirements on Device APIs and Functionality 14

7 Split Rendering Application Function 14

7.1 Functionality 14

7.2 RESTful APIs 14

7.2.1 Split-Rendering Provisioning procedures (SR-1) 14

7.2.1.1 General 14

7.2.1.2 Create Split-Rendering Configuration 14

7.2.1.3 Read Split-Rendering Configuration properties 14

7.2.1.4 Update Split-Rendering Configuration properties 14

7.2.1.5 Destroy Split-Rendering Configuration 15

7.2.2 Split-Rendering Provisioning API 15

7.2.2.1 Overview 15

7.2.2.2 Resource structure 15

7.2.2.3 Data model 15

7.2.2.3.1 SplitRenderingConfiguration resource 15

7.2.2.4 Configuration Guidelines for Split Rendering 16

7.2.2.4.1 Guidelines on Provisioning Session 16

7.2.2.4.2 Guidelines on Edge Resource Configuration 16

7.2.2.4.3 Guidelines on Policy Template 16

7.2.3 Dynamic Policy API for Split Rendering 16

8 Split Rendering User Plane 16

8.1 Split Rendering Signalling Protocols 16

8.2 Split Rendering Formats 17

8.2.1 General 17

8.2.2 Pixel Streaming Profile 17

8.2.2.1 Overview 17

8.2.2.2 Downlink Formats 17

8.2.2.3 Uplink Formats 17

8.3 Split Rendering Transport Protocols 18

8.4 Split Rendering Formats for Session Setup and Negotiation 18

8.4.1 General 18

8.4.2 Split Rendering Configuration Format 18

8.4.2.1 Introduction 18

8.4.2.2 Split Rendering Configuration Format 18

9 Split Rendering Client 19

9.1 Functionality 19

9.2 Client API 19

10 Security and Privacy Aspects 20

10.1 Security 20

10.2 Privacy 20

Annex A (Informative): Implementation Guidelines 21

Annex A.1 Guidelines for Application Developers 21

Annex A.2 Guidelines for Split Rendering MSE Implementers 21

Annex A.3 Conformance Testing 21

Annex B (normative): IDL Definition of Client API 22

Annex X (informative): Change history 23

# Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# Introduction

This specification defines a media service enabler for split rendering in the 5G system.

# 1 Scope

The present document defines a Media Service Enabler for Split Rendering according to the guidelines of TR26.857 [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".

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms given in TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in TR 21.905 [1].

**example:** text used to clarify abstract rules by applying them literally.

## 3.2 Symbols

For the purposes of the present document, the following symbols apply:

<symbol> <Explanation>

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in TR 21.905 [1].

AF Application Function

AS Application Server

MAF Media Access Function

MSH Media Session Handler

5G-RTC 5G Real-Time Communication

SR Split Rendering

SRC Split Rendering Client

SRS Split Rendering Server

UE User Equipment

# 4 General

## 4.1 Overview

The Split Rendering Media Service Enabler collects a set of 5G media functions to build a media service enabler that targets application developers, network operators, and application service providers, to enable the realization of split rendered applications.

The interfaces, formats, protocols, and APIs are either referenced or defined in this specification. This will allow for interoperability between multiple vendor implementations.

This specification targets primarily XR applications. However, it is not limited to XR applications and may be used for rendering for 2D displays.

## 4.2 Typical Use Cases

A typical use case for the split rendering MSE is immersive gaming. In this use case, the UE benefits from invoking split rendering by avoiding the download of the game to the phone and getting high quality graphics from edge rendering.

Another use case that can benefit from split rendering is immersive communication, where users gather in a shared space and interact with each other and with the environment. Users may be represented by sophisticated Avatars and as the number of users increases the rendering will become more complex.

# 5 Reference Architecture and Procedures

## 5.1 Reference Architecture

### 5.1.1 Introduction

In this clause, different variants of the reference architecture for the split rendering MSE are defined, each representing a different perspective and level of details.

The following functions are introduced:

* Split-Rendering Client (SRC): This function is responsible to acquire the UE media capabilities and negotiates with the RTC AS to agree on the split-rendering process at the RTC AS.
* Split-Rendering Server (SRS): This function is responsible for negotiation of SR session with SRC, monitoring the server’s edge resource usage, and managing/running the split rendering process.
* Application Function (AF): responsible for provisioning, QoS allocation, and edge resource discovery.
* Application Service Provider: The application provider that offers the service.
* Application: The application running on UE
* Media Session Handler (MSH): is the entity on UE that is responsible for the control plane communication with the AF.

### 5.1.2 Client Architecture

The client architectural breakdown is based on the client architecture in TS26.119 clause X.

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.

### 5.1.3 End-to-End Architecture



Figure 5.1-3 – Split management architecture

As shown in Figure 5.1.3:

1. The 5G Application Providers (AP) provisions the split-rendering through RTC-1.

2. In the use cases in which the AP is involved in the media delivery, the RTC-2 interface is used for this purpose.

3.The communication between AF and SRS is through RTC-3. 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 RTC-4.

5. The AF may provide the split-rendering information to the Media Session Handler defined by RTC-5, defined in TS26.506.

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

7. The 5G Application and AP interact through RTC-8-8. This interface is out of the scope of this document.

### 5.1.5 User Plane Architecture

Figure 5.1.5-1 depicts the user plane architecture for split rendering.

 

The SR interfaces are considered to be specializations of their parent RTC interfaces as defined in TS26.506.

In the context of split rendering, the SR-4 interface is further classified as SR-4s and SR-4m sub-interfaces. The SR-4s interface covers all user-plane signaling, including WebRTC and ICE signaling. The SR-4m serves for media and metadata exchange between the split rendering client and the split rendering server.

## 5.2 Procedures and Call Flows

### 5.2.1 Call flow for Split Rendering instance discovery

#### 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 Application Provider requests and sets up the edge server(s) used for the split-rendering as described in TS 26.506 clauses 6.1 or 6.2. The 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 Application Provider provisions the split-rendering session using SR-1 and SR-3, 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 8.1), only the client-driven split-rendering (5.2.1.1) is applicable.

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

#### 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 establishes the WebRTC session.

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

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

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

### 5.2.2 Call flow for Split Rendering session setup

The split rendering operation can be described by the as depicted in the call flow in the Figure 5.2.2-1.

![Msc-generator~|version=8.2.0~|lang=signalling~|size=857x485~|text=#This is the default signalling chart.~n#Edit and press F2 to see the result.~n#You can change the default chart~n#with the leftmost button on the Preferences pane of the ribbon.~n~nnumbering=yes;~nhscale=auto;~n~nsrc: Split REndering Client (SRC) {~nxr[label=~qXR\nRuntime~q], sm[label=~qScene\nManager\n(thin PE)~q], xrsrc[label=~qXR source\nManagement~q], maf[label=~qMedia\nAccess\nFunction~q];~n};~nsrs[label=~qSplit\nRendering\nServer~q];~n~nsm-~gsrs: create a split rendering session;~nsrs-~gsm: send description of split rendering output;~nsm-~gmaf-srs: establish transport connections\ne.g. WebRTC session;~nxr..srs:Rendering Loop [number=no] {~n~2xr~l-~gxrsrc: receive pose information and user actions;~n~2xrsrc-~gsrs: transmit pose infomration and user actions;~n~2srs..srs: perform rendering\nfor requested posse;~n~2srs-~gmaf: send next buffer frame(s);~n~2maf..maf: decode and process\nbuffer frame;~n~2maf-~gsm-xr: pass raw buffer frames for display;~n~2xr..xr: compose and \nrender frame;~n};~|]()

Figure 5.2.2- 1 High-level call flow for split rendering session setup and operation

The steps are:

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.
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.
5. The Source Manager shares the pose predictions and user input actions with the split rendering server.
6. The split rendering server uses that information to render the frame.
7. The rendered frame is encoded and streamed down to the MAF.

# 6 Prerequisites

## 6.1 Requirements on 5G System

## 6.2 Requirements on Device APIs and Functionality

*MSE-7*

# 7 Split Rendering Application Function

## 7.1 Functionality

The SR application function is an RTC AF that supports the provisioning and configuration of split rendering sessions. The SR AF shall support the procedures for the RTC AF as defined in TS26.506.

## 7.2 RESTful APIs

### 7.2.1 Split-Rendering Provisioning procedures (RTC-1)

#### 7.2.1.1 General

These procedures are used by the Application Provider and the AF on SR-1 to provision for the split-rendering process.

#### 7.2.1.2 Create Split-Rendering Configuration

This procedure is used by the AP to create a new Split-Rendering Configuration. The AP shall use the HTTP POST method for this purpose and the request message body shall include a SplitRenderingConfiguration resource.

If the procedure is successful, the AF shall generate a resource identifier representing the new Split-Rendering Configuration. In this case, the AF shall respond with a 201 (Created) HTTP response message and shall provide the URL to the newly created resource in the Location header field. The response message body may include a SplitRenderingConfiguration resource that represents the current state of the Split Rendering Configuration, including any fields set by the AF.

If the procedure is not successful, the AF shall provide a response code as defined in clause 7.3.

#### 7.2.1.3 Read Split-Rendering Configuration properties

This procedure is used by the AP to obtain the properties of an existing Split-Rendering resource from theAF. The HTTP GET method shall be used for this purpose.

If the procedure is successful, the AF shall respond with a 200 (OK) response message that includes the SplitRenderingConfiguration resource in the response message body.

If the procedure is not successful, the AF shall provide a response code as defined in clause 7.3.

#### 7.2.1.4 Update Split-Rendering Configuration properties

The update operation is invoked by the AP to modify the properties of an existing SplitRenderingConfiguration resource. All writeable properties except domainNameAlias may be updated. The HTTP PATCH or HTTP PUT methods shall be used for the update operation.

If the procedure is successful, the AF shall respond with a 200 (OK) and provide the content of the resource in the response, confirming the successful update operation.

If the procedure is not successful, the AF shall provide a response code as defined in clause 7.3.

#### 7.2.1.5 Destroy Split-Rendering Configuration

This operation is used by AP to destroy a Split-Rendering Configuration resource and to terminate the related distribution. The HTTP DELETE method shall be used for this purpose. As a result, the AF will release any associated network resources, purge any cached content, and delete any corresponding configurations.

If the procedure is successful, the AF shall respond with a 200 (OK) response message.

If the procedure is not successful, the A.F shall provide a response code as defined in clause 7.3.

### 7.2.2 Split-Rendering Provisioning API

### 7.2.2.1 Overview

This clause specifies the API that the AP uses at interface SR-1 to create and manage Provisioning sessions for media services that use split rendering. Each split rendering configuration is represented by a SplitRenderingConfiguration, for which the resource structure is specified in 7.2.2.2 and the data model is specified in clause 7.2.2.3.

### 7.2.2.2 Resource structure

The Split-Rendering Provisioning API is accessible through this URL base path:

{apiRoot}/3gpp-rtc1*/*{apiVersion}*/*provisioning-sessions/{provisioningSessionId}/

Table 7.2.2-1 below specifies the operations and the corresponding HTTP methods that are supported by this API. In each case, the Provisioning Session identifier shall be substituted into {provisioningSessionId} in the above URL template and the sub-resource path specified in the second column shall be appended to the URL base path.

Table 7.2.2‑1: Operations supported by the Split-Rendering Provisioning API

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Sub‑resource path | Allowed HTTP method(s) | Description |
| Create Split-Rendering Configuration | split-rendering-configuration | POST | Used to create a Split-Rendering Configuration resource. |
| Retrieve Split-Rendering Configuration | GET | Used to retrieve an existing Split-Rendering Configuration. |
| Update Split-Rendering Configuration | PUT,PATCH | Used to modify an existing Split-Rendering Configuration. |
| Delete Split-Rendering Configuration | DELETE | Used to delete an existing Split-Rendering Configuration. |
|  |  |  |  |

### 7.2.2.3 Data model

#### 7.2.2.3.1 SplitRenderingConfiguration resource

The data model for the SplitRenderingConfiguration resource is specified in table 7.2.2-2 below:

Table 7.2.2-2: Definition of SplitRenderingConfiguration resource

| Property name | Data Type | Cardinality | Description |
| --- | --- | --- | --- |
| Name | String | 1..1 | A name for this Split Rendering Configuration. |
|  Status | Boolean | 1..1 | Indicates whether to this split rendering configuration is active |
|  edgeResourceConfigurationId | ResourceId | 0..1 | The identifier of the edge resource configuration that will be used for sessions of this split rendering configuration. |
|  policyTemplateId | ResourceId | 1..1 | The identifier of the policy template that will be applied to the sessions of this split rendering configuration. |
|  Configuration | Object | 1..1 | Describes the split-rendering configuration currently used by the SRS. Editor’s Note: The syntax and semantics of this element are TBD. |

### 7.2.2.4 Configuration Guidelines for Split Rendering

#### 7.2.2.4.1 Guidelines on Provisioning Session

The ProvisioningSessionType shall be set to “BIDIRECTIONAL”.

The aspId shall be configured and shall be a unique identifier for the Application Service 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 examples is as follows: “urn:com:example:game+3gpp-sr”.

#### 7.2.2.4.2 Guidelines on Edge Resource Configuration

A split rendering application may use the procedures defined in TS26.512 clause 7.10 to define an edge resource configuration to be used for split-rendering session. 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.
* The easRelocationRequirements shall indicate “RELOCATION\_INTOLERANT” in the tolerance field.

#### 7.2.2.4.3 Guidelines on Policy Template

A policy template for the split rendering shall be associated with the split rendering configuration. The QoS specification should include at least the following sub-streams:

* 2 configurations for left and right eye buffer streams
* 1 optional configuration for a depth buffer stream
* 1 configuration for an audio stream

### Dynamic Policy API for Split Rendering

The WebRTC Signaling Server (which potentially maybe part of the Split Rendering Server) shall support the dynamic policy API as defined in X.

Furthermore, the Split Rendering Server 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.

## 7.3 HTTP response codes

Guidelines for error responses to the invocation of APIs of NF services are specified in clause 4.8 of TS 29.501 [22]. API-specific error responses are specified in the respective technical specifications.

# 8 Split Rendering User Plane

## 8.1 Split Rendering Signalling Protocols

## 8.2 Split Rendering Formats

### 8.2.1 General

### 8.2.2 Pixel Streaming Profile

#### 8.2.2.1 Overview

The full-prerendering profile is restricted to the support of 2D content exclusively. The capabilities of the receiving UE are shared with the split rendering server prior to the start of the split rendering session. These capabilities and configurations would indicate the audio-visual output setup on the UE. For example, it would indicate that the output device is an HMD that supports 2 views and stereo audio.

#### 8.2.2.2 Downlink Formats

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.

Editor’s Note: Referencing the downlink formats in MeCAR TS 26.119, potentially to OpenXR formats, is TBD.

#### 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.

Editor’s Note: Referencing the uplink formats in MeCAR TS 26.119, potentially to OpenXR formats, is TBD.

## 8.3 Split Rendering Transport Protocols

## 8.4 Split Rendering Formats for Session Setup and Negotiation

### 8.4.1 General

In Figure 5.2.1-1 and 5.2.1-2, step 5 defines the negotiation between the SRC and SRS for the split-rendering configuration. In most simple case, the SRC provides SRS the capabilities of the device and if SRS can accommodate the split-rendering processing that addresses the device, it confirms the configuration. In such scheme, the SRS is responsible to make the decision and no back-and-forth negotiation occurs.

### 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:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Cardinality** | **Description** |
| 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. |
| 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. |
| extraConfigurations | Object | 0..1 | A placeholder for addition configuration information. |

# 9 Split Rendering Client

## 9.1 Functionality

*Reference pre-requisites, user plane functionality, control plane*

## 9.2 Client API

*MSE-6*

# 10 Security and Privacy Aspects

## 10.1 Security

Signaling for session establishment and exchange of application-specific messages shall use a secure transport channel based on WebSockets as defined in TS26.113.

Media transport shall be secured by the usage of WebRTC.

## 10.2 Privacy

Editor’s Note: Privacy considerations are FFS.

# Annex A (Informative):Implementation Guidelines

## Annex A.1 Guidelines for Application Developers

## Annex A.2 Guidelines for Split Rendering MSE Implementers

## Annex A.3 Conformance Testing

Annex B (normative):
IDL Definition of Client API

Annex X (informative):
Change history

|  |
| --- |
| Change history |
| Date | Meeting | TDoc | CR | Rev | Cat | Subject/Comment | New version |
| April 2023 | 123-e | S4-230726 |  |  |  | Improvements and Corrections to edge and dynamic policy procedures in SR |  |