ETSI TS 126 567 V19.0.0 (2025-10)

TECHNICAL SPECIFICATION

5G;
Split rendering over IMS
(3GPP TS 26.567 version 19.0.0 Release 19)

Reference
DTS/TSGS-0426567vj00

Keywords
5G

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Legal Notice

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

The present document may refer to technical specifications or reports using their 3GPP identities. These shall be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between 3GPP and ETSI identities can be found at 3GPP to ETSI numbering cross-referencing.

Modal verbs terminology

In the present document "**shall**", "**shall not**", "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Contents

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

The present document specifies functional entities, reference points and protocols for IMS-based split rendering. It also specifies codecs for delivery of split-rendered media and metadata for split-rendered content for both uplink and downlink. The present document also specifies procedures for split rendering session establishment, session management, adaption, as well as other procedures to support split rendering process based on network support functions.

Key use cases enabled by this specification include XR services incorporating real-time and non-real-time media in industrial (e.g., for monitoring, maintenance, collaboration, and tele-operation), enterprise and educational environments; entertainment use-cases, including cloud-gaming, and shared and collaborative entertainment and productivity XR services over IMS.

1 Scope

The present document specifies functional entities, reference points and protocols for IMS-based split rendering. It also specifies codecs for delivery of split-rendered media and metadata for split-rendered content for both uplink and downlink. The present document also specifies procedures for split rendering session establishment, session management, adaptation, as well as other procedures to support split rendering process based on network support functions.

Key use cases enabled by this specification include XR services incorporating real-time and non-real-time media in industrial (e.g., for monitoring, maintenance, collaboration, and tele-operation), enterprise and educational environments; entertainment use-cases, including cloud-gaming, and shared and collaborative entertainment and productivity XR services over IMS.

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 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 23.228: "IP Multimedia Subsystem (IMS); Stage 2".

[3] 3GPP TS 26.264: "IMS-based AR Real-Time Communication".

[4] 3GPP TS 23.501: "System architecture for the 5G System (5GS); Stage 2".

[5] 3GPP TS 26.565: "Split Rendering Media Service Enabler".

[6] 3GPP TS 26.119: "Device Media Capabilities for Augmented Reality Services".

[7] 3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction".

[8] OMA-ERELD-DM-V1_2-20070209-A: "Enabler Release Definition for OMA Device Management, Approved Version 1.2".

[9] 3GPP TS 28.405; "Management of Quality of Experience (QoE) measurement collection; Control and configuration"

[10] IETF RFC 3550 (2003): "RTP: A Transport Protocol for Real-Time Applications".

[11] IETF RFC 4960 (2007): "Stream Control Transmission Protocol".

[12] IETF RFC 8261 (2017): "Datagram Transport Layer Security (DTLS) Encapsulation of SCTP Packets".

[13] IETF RFC 8831 (2021): "WebRTC Data Channels".

[14] 3GPP TS 29.510: “Network function repository services; Stage 3".

[15] IETF RFC 2326 (1998): "Real Time Streaming Protocol (RTSP)".

[16] Microsoft MSFT_lod extension of Khronos glTF 2.0: https://github.com/KhronosGroup/glTF/tree/main/extensions/2.0/Vendor/MSFT_lod.

[17] LOD node of X3D™ Standard by Web3D™: https://www.web3d.org/documents/specifications/19775-1/V3.3/Part01/components/navigation.html#LOD.

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].

SR-DCMTSI Client: a split rendering capable DCMTSI Client.

Split rendering session: a media session running the split rendering process between two or more entities. A split rendering session may include a data channel, consisting of application data channel, and one or more RTP streams for delivering split rendered media.

3.2 Symbols

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

void

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].

A2P	Application to Person
API	Application Programming Interface
AR	Augmented Reality
AS	Application Server
DC	Data Channel
DC AS	Data Channel Application Server
DCMTSI	Data Channel Multimedia Telephony Service for IMS
DCSF	Data Channel Signalling Function
IMS	IP Multimedia Core Network Subsystem
LoD	Level of Detail
MF	Media Function
MTSI	Multimedia Telephony Service for IMS
NEF	Network Exposure Function
NRF	Network Repository Function
P2A	Person to Application
P2P	Person to Person
QMC	QoE Measurement Collection
QoE	Quality of Experience
SR	Split Rendering
UE	User Equipment

4 System description

4.1 Overview

The IMS stage-2 service description is specified in TS 23.228 [2]. The IMS architecture enhancements to support data channel services are specified in TS 23.228 Annex AC [2], which supports separation of signalling function and media function supporting data channel services.

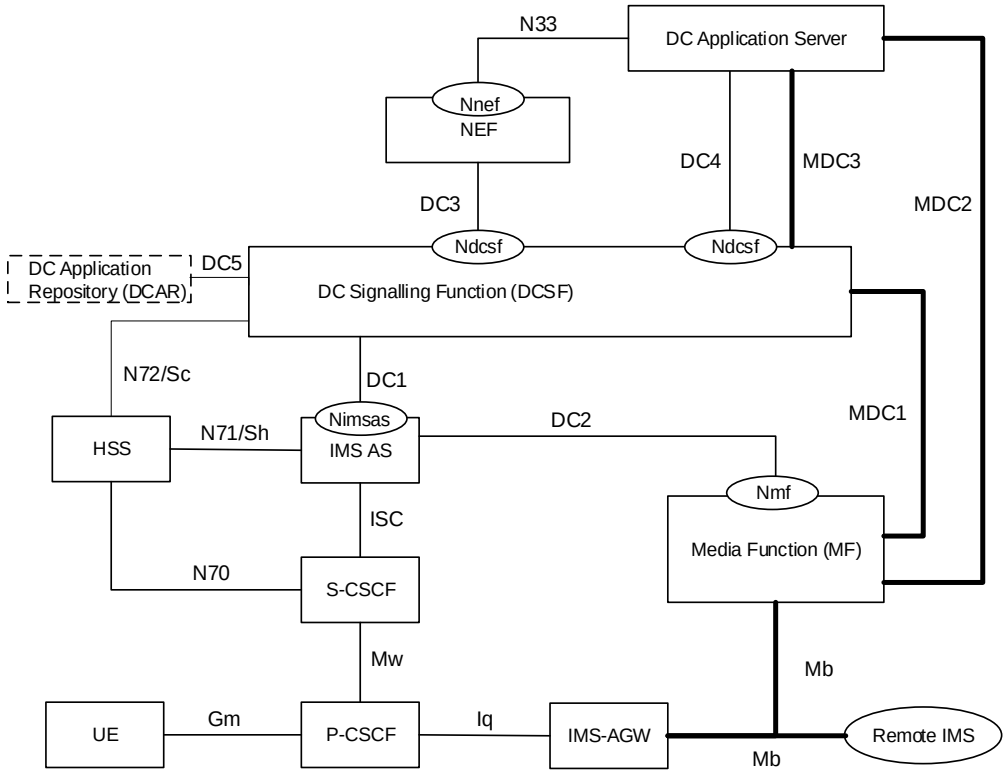

Figure 4.1.1: Architecture of IMS supporting DC usage with MF

Based on the architecture in Figure 4.1.1, TS 26.264 [3] provides the IMS-based conversational AR (Augmented reality) services, including the generalized end-to-end architecture to support AR communication over IMS DC, as shown in Figure 4.1.2.

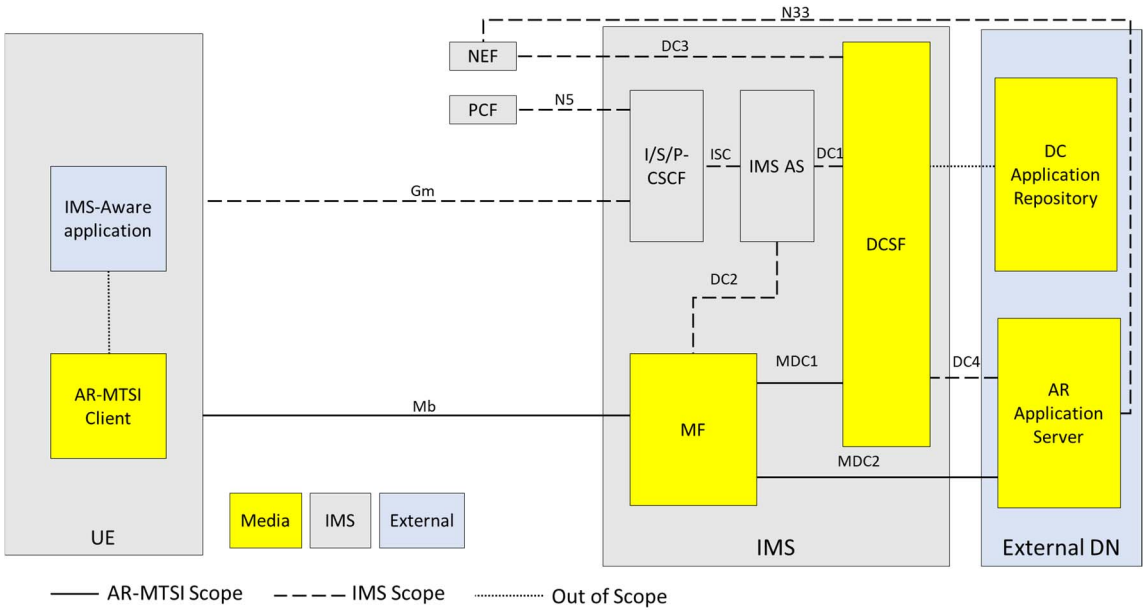

Figure 4.1.2: Generalized IMS DC Architecture to support AR communication (TS 26.264)

Accordingly, AR Application Server (AR AS) is responsible for AR service control related to AR communication, including AR session media control and AR media capability negotiation with the UE.

The DCSF receives event reports from the IMS AS and decides whether AR communication service is allowed to be provided during the IMS session. Additionally, the DCSF interacts with the AR AS for DC resource control.

MF supports AR conversational service by providing transcoding for terminals with limited capabilities. Additionally, the MF may collect spatial and media descriptions from UEs and create scene descriptions for symmetrical AR call experiences. MF also provide remote rendering for AR-MTSI clients in terminals with limited capabilities based on rendering negotiation. For remote rendering the AR-MTSI client provides AR metadata.

The IMS AS receives the media control instructions from the DCSF and accordingly interacts with the UE for connecting the UE's audio/video media termination to the MF and interacts with MF for data channel media resource management for AR media processing.

According to the architectures above, the present document introduces a mapping to the IMS architecture for IMS-based split rendering.

4.2 Reference Architecture

The generalized IMS DC architecture to support split rendering is shown in Figure 4.2.1.

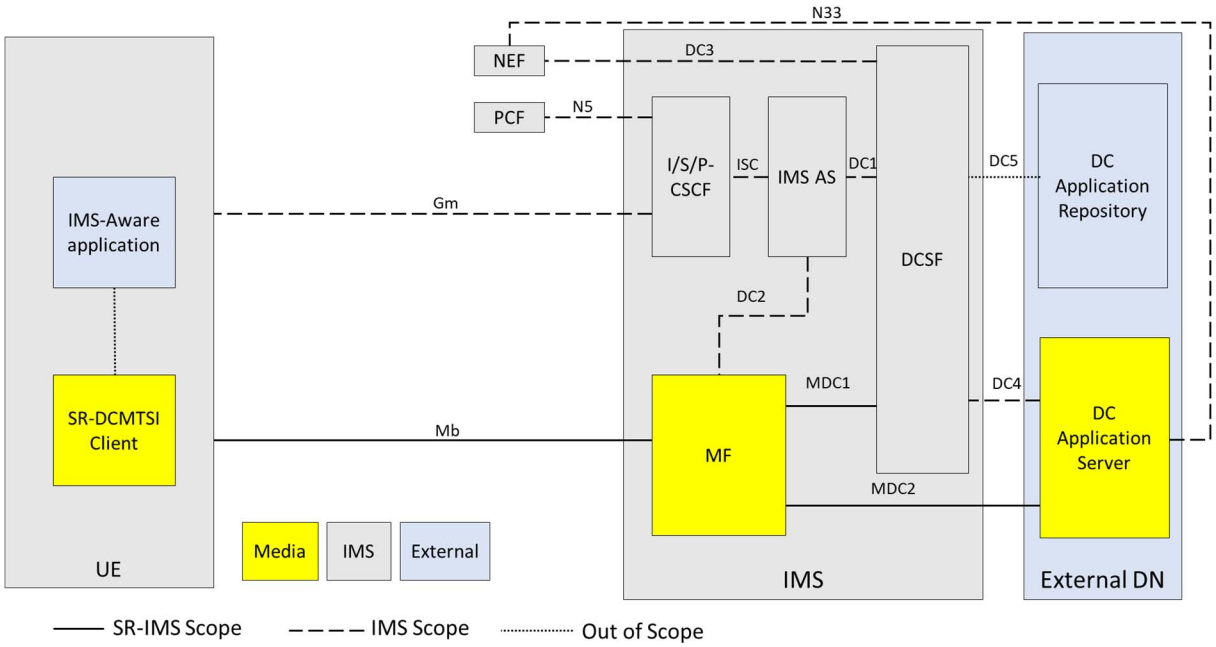

Figure 4.2.1: Generalized IMS DC Architecture to support split rendering

Split-Rendering DCMTSI Client (SR-DCMTSI Client): A DCMTSI Client, that is responsible for acquiring the UE media capabilities and interacting with the MF during the split-rendering process.

Media Function (MF): A Media Function, that is responsible for interacting with SR-DCMTSI Client during split-rendering process, monitoring resource usage, and managing/running the split rendering process, etc.

Data Channel Application Server (DC AS): A DC Application Server, that is responsible for service control related to split-rendering, including session media control and media capability negotiation with the UE via MF, etc.

4.3 Reference Points

As shown in Figure 4.2.1, the following reference points are used to enable split rendering over IMS:

Mb: Reference point to enable split rendering between UE and MF, as specified in TS 23.228 [2].

Gm: Reference point to support communication between UE and IMS, as specified in TS 23.228 [2].

MDC1: Reference point for transport of data channel media between data channel media function and DCSF. MF terminates the bootstrap data channel from the UE and forward HTTP traffic between UE and DCSF via MDC1, as specified in TS 23.228 [2].

MDC2: Reference point for transport of data channel media between data channel media function and DC Application Server, for split rendering functions between application server to MF. MF relay traffic on A2P/P2A application data channels between the UE and the DC Application Server via MDC2, as specified in TS 23.228 [2].

The following reference points are also used to support split rendering related procedures:

DC1: Reference point between the DCSF and the IMS AS, as specified in TS 23.228 [2].

DC2: Reference point between the IMS AS and MF, for split rendering related data channel media resource management, as specified in TS 23.228 [2].

DC3: Reference point between the DCSF and NEF, as specified in TS 23.228 [2].

DC4: Reference point between the DCSF and DC Application Server, as specified in TS 23.228 [2].

N33: Reference point between NEF and DC Application server, network exposure to enable split rendering related applications, as specified in TS 23.501 [4].

NOTE: DC5, ISC, N5, and the reference point between IMS-aware application and SR-DCMTSI are out of the scope of this specification.

4.4 Split Rendering DCMTSI Client (SR-DCMTSI)

An SR-DCMTSI is a DCMTSI client in terminal defined as per clause 3.1 of TS 26.114 [7] which supports split rendering. It is responsible for acquiring media capabilities and interacting with IMS functions during a split rendering session. It is responsible for the signalling with DCSF via IMS AS to set up a split rendering session and negotiating with the MF and DCAS about the parameters of the split rendering session, for example, the scene to be rendered and the split of rendering operations.

The SR-DCMTSI Client performs the following functions:

- Initiate a request of split rendering session establishment,
- Establish split rendering session, including signalling with DCSF via IMS AS to set up the data channel, and MF and DC AS to set up the split rendering,
- Operate the rendering loop on the UE,
- Update the rendering loop according to the network parameters and media capabilities with IMS functionalities,
- Stop the split rendering session or switch from a split rendering session to a local rendering session.

4.5 Media Function (MF)

4.5.1 Media Functions

MF is a Media Function which supports split rendering. It is responsible for interacting with SR-DCMTSI client and the DC AS during split-rendering process, monitoring resource usage, managing and running the split rendering process, etc.

The MF performs the following functions:

- Negotiate and interact with SR-DCMTSI Client (via IMS AS) and DC AS to establish the split rendering session,
- Operate the rendering loop, including, receiving meta-data from the SR-DCMTSI client, rendering frames, and forwarding requests to DC AS based on the split rendering logic,
- Expose its capabilities, capacity and media processing and rendering capabilities to the DC AS,
- Update the rendering loop according to the split rendering logic and requests from UE. If supported, the update can be based on the network parameters and interactions with other IMS functionalities,
- Stop/Pause/Resume the split rendering session according to the requests from UE.

4.5.2 Media Function (MF) Capabilities

4.5.2.1 MF Capabilities for SR

An MF providing split rendering services specified in this document shall comply with NRF registration and discovery procedures specified in AC.7.4.2 of TS 23.228 [2].

The capabilities required for an MF to support split rendering are specified as SR service profiles each identified by a unique URN. To support split rendering specified in this specification, an MF shall support the SR service profile Basic as defined in clause 4.5.2.2 and may support other SR service profiles for minimum interoperability. Each SR service profile shall be uniquely identified by a urn. An MF shall list the urn of each SR service profile it supports as a

MediaCapability in an MfInfo object as respectively defined in clauses 6.1.6.2.2 and 6.1.6.2.119 of TS 29.510 [14] in its NF profile when registering to an NRF.

4.5.2.2 MF Service Profiles for SR

4.5.2.2.1 Profile Basic

MF shall support:

- Scene description processing capabilities as specified in SD-Rendering-glTF-Core in TS 26.119 [6].
- Video encoding capabilities required to encode video complying with the capabilities specified in Annex Y.3 in TS 26.114 [7].
- Audio and speech encoding capabilities required to encode audio and speech complying with the capabilities specified in clause Y.4 of TS 26.114 [7].
- The type **urn:3gpp:split-rendering:mf:profile:Basic** shall be included in the Split Rendering Configuration defined in clause A.1.3 when the MF signals the SR-DCMTSI client in terminal.

4.5.2.2.2 Profile Advanced

MF shall support the basic profile and the following media processing capabilities:

- Scene description processing capabilities SD-Rendering-gltf-ext1, SD-Rendering-gltf-ext2 and SD-Rendering-gltf-interactive as specified in clause 10.4.5 of TS 26.119 [6].
- Video encoding capabilities required to encode video complying with the capabilities specified in clause 10.4.3 of TS 26.119 [6].
- Audio and speech encoding capabilities required to encode audio and speech complying with the capabilities specified in clause 10.4.4 of TS 26.119 [6].
- The type **urn:3gpp:split-rendering:mf:profile:Advanced** shall be included in the Split Rendering Configuration defined in clause A.1.3 when the MF signals the SR-DCMTSI client in terminal.

4.5.2.2.3 Application-Specific Profile

An MF that is capable of running and rendering external applications shall support the following media processing capabilities:

- Video encoding capabilities required to encode video complying with the capabilities specified in clause 10.4.3 of TS 26.119 [6].
- Audio and speech encoding capabilities required to encode audio and speech complying with the capabilities specified in clause 10.4.4 of TS 26.119 [6].
- The type **urn:3gpp:split-rendering:mf:profile:app-specific** shall be included in the Split Rendering Configuration defined in clause A.1.3 when the MF signals the SR-DCMTSI client in terminal.

In addition, by declaring applications among the supportedFeatures as defined in TS 29.510 [14], the MF declares that it fulfills all requirements necessary for running these declared applications and features.

4.5.2.3 MF API

The MF shall provide a RESTful API to the DC AS over MDC2 once an application DC for SR is established. The API shall be:

Table 4.5.2.3-1 MF APIs

Operation name	Allowed HTTP method(s)	Description
Get MF Profiles	GET	List of MF service profiles supported for SR by the MF based on its current operating conditions. MF service profiles for SR are defined as specified in clause 4.5.2.2. Each profile identified by a unique URN. The MF shall return a list of URNs of the profiles it can support based on its current operating conditions.
Get Capabilities(MF Profile)	GET	Enumerate the capabilities of the MF. This information may detail the rendering capacity of the MF. For example, GPU type, driver version/type, graphics runtimes and engines, VRAM, Scene description processing capabilities etc.
Select MF Profile(MF Profile)	SET	Select an MF profile whose URN is provided as argument to the SET operation. The MF profile whose URN is provided as argument shall be selected from the list of MF profiles returned by the Get MF Profiles operation.

4.6 DC Application Server (DC AS)

DC Application Server (DC AS) is responsible for service control related to split-rendering, including session media control and, session setup and media capability negotiation with the SR-DCMTSI client via MF and DCSF where applicable. The DC AS may provide split rendering application source data to the MF. The DC AS may be in the media path via MDC2, for example in Person to Application (P2A), Application to Person (A2P) or P2A2P scenarios as defined in clause AC.7.2.2 and AC.7.2.3 of TS 23.228 [2]. The DC AS may perform split rendering when it is in the media path.

The DC AS performs the following functions:

- Provide the DC application service, including the IMS DC application, to the DCSF.
- Negotiate and interact with SR-DCMTSI Client via MF and DCSF to establish the split rendering session,
- Query MF capabilities and capacity.
- Provide application source data for split rendering to the MF and to the SR-DCMTSI client
- If the DC AS is performing split rendering,
- Operate the rendering loop, including receiving meta-data from the SR-DCMTSI client via the MF, rendering frames based on the split rendering logic.
- Update the rendering loop according to the requests from UE via the MF
- Stop/Pause/Resume the split rendering session according to the requests from the SR-DCMTSI client.

5 Media codecs, configuration, and data transport

5.1 General

The SR-DCMTSI Client shall support the user plane protocol stack defined in Figure 4.3 of TS 26.114 [7]. All media components except data media components are transported over RTP with each respective payload format mapped onto RTP (RFC3550 [10]) streams. The data media components are transported over the data channels using SCTP (RFC4960 [11]) over DTLS (RFC8261 [12]), as specified for WebRTC data channels (RFC8831 [13]).

5.2 Media codecs

An SR-DCMTSI client that supports audio shall support the codec requirements for MTSI clients as specified in clause 5.2.1 of TS 26.114 [7].

An SR-DCMTSI client that supports video shall support the codecs requirements for MTSI clients as specified in clause 5.2.2 of TS 26.114 [7].

An SR-DCMTSI client that supports real-time text shall support the codec requirements for MTSI clients as specified in clause 5.2.3 of TS 26.114 [7].

An SR-DCMTSI client that supports still images shall support the codec requirements for MTSI clients as specified in clause 5.2.4 of TS 26.114 [7].

SR DCMTSI clients conforming to device types defined in clause 10 of TS 26.119 [6] should conform to media capabilities recommended for the respective device type in clause 10 of TS 26.119 [6].

5.3 Media configuration

SR-DCMTSI clients shall support media configuration requirements specified in clause 6 of TS 26.114 [7].

5.4 Data transport

5.4.1 General

An SR-DCMTSI client shall support data channel media and support procedures in clause 6.2.10 of TS 26.114 [7].

An SR-DCMTSI client shall support the data transport requirements specified in clause 7 of TS 26.114 [7].

Application data channels over which meta-data is transported shall support the IMS DC requirements in TS 23.228 [2], and requirements specified in clause 5.4.2.

5.4.2 Metadata Formats

5.4.2.1 General

SR-DCMTSI client and Media Function shall support the usage of the IMS data channel for the exchange of split rendering metadata with the MF. The 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 of TS 26.565 [5].

5.4.2.2 Pose Format

For XR services, the pose information format that is used for IMS-based split rendering shall comply with the format defined in clause 12.2 of TS 26.119 [6]. The pose information shall be carried as part of the data channel messaging mechanism. The metadata data channel message format is as defined in clause A.1.2. The message type shall be “urn:3gpp:split-rendering:v1:pose”.

5.4.2.3 Action Format

The action information format that is used for IMS-based split rendering shall comply with the format defined in clause 12.3 of TS 26.119 [6]. The action information shall be carried as part of the data channel messaging mechanism. The metadata data channel message format is as defined in clause A.1.2. The message type shall be “urn:3gpp:split-rendering:v1:action”.

5.4.2.4 Split Rendering Configuration Format

The SR-DCMTSI client and Media Function shall support the split rendering session configuration defined in clause Annex A.1.3.

If DC AS is in the media path performing split rendering, it shall support the split rendering session configuration defined in clause Annex A.1.3.

The split rendering configuration message shall be identified as “urn:3gpp:split-rendering:v2:sr-configuration”.

5.4.3 Metadata Data Channel Message Format

For the carriage of metadata defined in clause 5.4.2, such as pose and action information, the SRDCMTSI client and MF shall use an IMS application data channel. The data channel sub-protocol shall be identified as “3gpp-sr”, which shall be included in the dcmap attribute of the SDP.

The transmission order for the data channel shall be set to in-order and the transmission reliability shall be set to reliable.

The split rendering metadata message format shall be set to text-based and the messages shall be UTF-8 encoded JSON messages.

A data channel message may carry one or more split rendering messages as defined in Clause 8.3.3 of TS 26.565 [5] and reproduced in Table A.1.2-1.

Each split rendering message shall follow the format specified in Clause 8.3.3 of TS 26.565 [5] and reproduced in Table A.1.2-2.

6 Split Rendering Metrics

6.1 Metrics definition and formats

SR-DCMTSI client and Media Function supporting the QoE metrics feature shall support the collection and reporting of the metrics in clause 16.2 of TS 26.114 [7], which include corruption duration metric, successive loss of RTP packets, frame rate, jitter duration, sync loss duration, round-trip time, average codec bitrate, codec information, call setup time.

SR-DCMTSI client and Media Function supporting the QoE metrics feature shall also support the collection and reporting of the split-rendering latency metrics defined in clause 9.3.4 of TS 26.565 [5] according to the QoE metrics configuration, which include pose-to-render-to-photon, render-to-photon, round-trip interaction delay, user interaction delay, age of content, scene update delay, metadata delay, and data frame delay metrics. A Media Function may use the “QoE timing information” defined in clause 9.3.2 of TS 26.565 [5] to transmit the timing information required for measuring the QoE latency metrics to an SR-DCMTSI client.

The metrics listed above are valid for one or more media types such as speech/audio, video, and text, and are calculated for each measurement resolution interval "Measure-Resolution". They are reported to the OAM [9] or QoE server as shown in figure 16.5.1-1 of TS 26.114 [7] according to the measurement reporting interval "Sending-Rate" and after the end of the session.

6.2 Metrics Configuration

SR-DCMTSI client and Media Function supporting the QoE metrics feature shall support the usage of an OMA-DM solution [8] for configuration of QoE metrics and their activation. As an alternative, metrics configuration can be specified by the QoE Measurement Collection (QMC) functionality.

6.3 Metrics Reporting

6.3.1 General

The metrics reporting procedure specified in clause 16.4 of TS 26.114 [7] allows the SR-DCMTSI client to send QoE metrics reports to the QoE server.

An SR-DCMTSI Client shall report QoE metrics specified in clause 6.2 for the real-time media it has received using the protocol specified in clause 16.4 of TS 26.114 [7] according to the QoE metrics reporting configuration obtained in a 3GPP MTSIQOE (MTSI QoE metrics) management object (see clause 16.3.1 of TS 26.114 [7]) or in an RRC message (see clause 16.5.1 of TS 26.114 [7]).

The quality metrics report follows the XML-based report format defined in clause 6.3.3.

SR-DCMTSI Clients shall use the MIME type "`application/3gprtc-qoe-report+xml`" for an XML-formatted QoE report. The metrics report format is defined in clause 6.3.3.

6.3.2 QoE metric reporting configuration

The syntax of the "3GPP-QoE-Metrics" attribute specified in clause 16.3.2 of TS 26.114 [7] is extended as follows:

- QoE-Metrics = "3GPP-QoE-Metrics:" att-measure-spec *("," att-measure-spec)) CRLF
- att-measure-spec = Metrics ";" Sending-rate [";" Measure-Range] [";" Measure-Resolution] *([";" Parameter-Ext])
- Metrics = "metrics" "=" "{"Metrics-Name *("|" Metrics-Name) " }"
- Metrics-Name = 1*((%x21-2b) / (%x2d-3a) / (%x3c-7a) / %x7e) [":" Positive-Threshold ":" Negative-Threshold ":" Target] ;VCHAR except ";", ",", "{" or "}"
- Positive-Threshold = "positive=" (1*DIGIT ["." 1*DIGIT]) ; positive crossing threshold
- Negative-Threshold = "negative=" (1*DIGIT ["." 1*DIGIT]) ; negative crossing threshold
- Target = "target=" (1*DIGIT ["." 1*DIGIT]) ; target value
- Sending-Rate = "rate" "=" 1*DIGIT / "End"
- Measure-Resolution = "resolution" "=" 1*DIGIT ; in seconds
- Measure-Range = "range" ":" Ranges-Specifier
- Parameter-Ext = (1*DIGIT ["." 1*DIGIT]) / (1*((%x21-2b) / (%x2d-3a) / (%x3c-7a) / %x7c / %x7e))
- Ranges-Specifier = as defined in RFC 2326 [15].

The "Metrics", "Sending-Rate", "Measure-Resolution" and "Measure-Range" fields are defined in clause 16.3.2 of TS 26.114 [7].

The optional "Positive-Threshold" field, if used, shall define the positive crossing threshold of a QoE metric. When present, the QoE metric shall be reported once when its value exceeds the threshold value indicated in the "Positive-Threshold" property and shall not be reported again until it falls below that threshold and subsequently exceeds it.

The optional "Negative-Threshold" field, if used, shall define the negative crossing threshold of a QoE metric. When present, the QoE metric shall be reported once when its value falls below the threshold value indicated in the "Negative-Threshold" property and shall not be reported again until it exceeds that threshold and subsequently falls below it.

The optional "Target" field, if used, shall define the target value of a QoE metric.

An example for a QoE metrics reporting configuration is as shown below:

3GPP-QoE-Metrics:metrics={Round_Trip_Time:positive=80:negative=20:target=50};rate=5;resolution=1

6.3.3 Report format

The QoE report is formatted as an XML document that complies with the XML schema in listing 16.4.1 of TS 26.114 [7].

The schema in listing 6.3.3-1 allows the SR-DCMTSI client to report QoE metrics using the metrics reporting mechanism specified in clause 16.4 of TS 26.114 [7].

The filename of this schema is "TS26567_SR_IMSQoEMetrics.xsd".

Listing 6.3.3-1: SR_IMS QoE Metrics XML schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="urn:3gpp:metadata:2024:RTC:SR_IMSQoEMetrics"
    xmlns="urn:3gpp:metadata:2024:RTC:SR_IMSQoEMetrics"
    xmlns:sv="urn:3gpp:metadata:2017:MTSI:schemaVersion"    elementFormDefault="qualified">

    <xs:element name="QoeReport" type="QoeReportType"/>

    <xs:complexType name="QoeReportType">
    <xs:sequence>
        <xs:element name="statisticalReport" type="starType" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:any namespace="##other" processContents="skip" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="starType">
    <xs:sequence>
        <xs:element name="mediaLevelQoeMetrics" type="mediaLevelQoeMetricsType" minOccurs="1"
        maxOccurs="unbounded"/>
        <xs:element ref="sv:delimiter"/>
        <xs:element name="latencyQoeMetric" type="QoeMetricType" minOccurs="0"
        maxOccurs="unbounded"/>
        <xs:element ref="sv:delimiter"/>
    </xs:sequence>
    <xs:attribute name="startTime" type="xs:unsignedLong" use="required"/>
    <xs:attribute name="stopTime" type="xs:unsignedLong" use="required"/>
    <xs:attribute name="callId" type="xs:string" use="required"/>
    <xs:attribute name="clientId" type="xs:string" use="required"/>
    <xs:attribute name="qoeReferenceId" type="xs:hexBinary" use="optional"/>
    <xs:attribute name="recordingSessionId" type="xs:hexBinary" use="optional"/>
    <xs:attribute name="dnn" type="xs:string" use="optional"/>
    <xs:attribute name="snssai" type="xs:unsignedLong" use=”optional"/>
    <xs:anyAttribute processContents="skip"/>
    </xs:complexType>

    <xs:complexType name="mediaLevelQoeMetricsType">
    <xs:sequence>
        <xs:any namespace="##other" processContents="skip" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="mediaId" type="xs:integer" use="required"/>
    <xs:attribute name="totalCorruptionDuration" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="numberOfCorruptionEvents" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="corruptionAlternative" type="xs:string" use="optional"/>
    <xs:attribute name="totalNumberofSuccessivePacketLoss" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="numberOfSuccessiveLossEvents" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="numberOfReceivedPackets" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="framerate" type="doubleVectorType" use="optional"/>
    <xs:attribute name="totalJitterDuration" type="doubleVectorType" use="optional"/>
    <xs:attribute name="numberOfJitterEvents" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="totalSyncLossDuration" type="doubleVectorType" use="optional"/>
    <xs:attribute name="numberOfSyncLossEvents" type="unsignedLongVectorType"
        use="optional"/>
    <xs:attribute name="networkRTT" type="unsignedLongVectorType" use="optional"/>
    <xs:attribute name="internalRTT" type="unsignedLongVectorType" use="optional"/>
```

```
    <xs:attribute name="codecInfo" type="stringVectorType" use="optional"/>
    <xs:attribute name="codecProfileLevel" type="stringVectorType" use="optional"/>
    <xs:attribute name="codecImageSize" type="stringVectorType" use="optional"/>
    <xs:attribute name="averageCodecBitrate" type="doubleVectorType" use="optional"/>
    <xs:attribute name="callSetupTime" type="xs:unsignedLong" use="optional"/>
    <xs:anyAttribute processContents="skip"/>
    </xs:complexType>


    <xs:element name="latencyQoeMetric" type="QoeMetricType"/>
    <xs:complexType name="QoeMetricType">
        <xs:sequence>
            <xs:choice>
                <xs:element name="poseToRenderToPhoton" type="PoseToRenderToPhotonType"/>
                <xs:element name="renderToPhoton" type="RenderToPhotonType"/>
                <xs:elementname="roundTripInteractionDelay"
type="RoundTripInteractionDelayType"/>
                <xs:element name="userInteractionDelay" type="UserInteractionDelayType"/>
                <xs:element name="ageOfContent" type="AgeOfContentType"/>
                <xs:element name="sceneUpdateDelay" type="SceneUpdateDelayType"/>
                <xs:element name="metadataDelay" type="MetadataDelayType"/>
                <xs:element name="dataFrameDelay" type="DataFrameDelayType"/>
            </xs:choice>
            <xs:element ref="sv:delimiter"/>
            <xs:any namespace="##other" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
<xs:complexType name="PoseToRenderToPhotonType">
        <xs:attribute name="avgPoseToRenderToPhoton" type="doubleVectorType" use="optional"/>
        <xs:attribute name="minPoseToRenderToPhoton" type="unsignedIntVectorType" use="
optional"/>
        <xs:attribute name="maxPoseToRenderToPhoton" type="unsignedIntVectorType"
use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="RenderToPhotonType">
        <xs:attribute name="avgPoseToRenderToPhoton" type="doubleVectorType" use="optional"/>
        <xs:attribute name="minRenderToPhoton" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="maxRenderToPhoton" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="RoundTripInteractionDelayType">
        <xs:attribute name="avgRoundTripInteractionDelay" type="doubleVectorType"
use="optional"/>
        <xs:attribute name="numberOfUserActions" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="minRoundTripInteractionDelay" type="unsignedIntVectorType"
use="optional"/>
        <xs:attribute name="minActionIds" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="maxRoundTripInteractionDelay" type="UnsignedIntVectorType"
use="optional"/>
        <xs:attribute name="maxActionIds" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="UserInteractionDelayType">
        <xs:attribute name="avgUserInetractionDelay" type="doubleVectorType" use="optional"/>
        <xs:attribute name="numberOfUserActions" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="minUserInetractionDelay" type="unsignedIntVectorType"
use="optional"/>
        <xs:attribute name="minActionIds" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="maxUserInteractionDelay" type="unsignedIntVectorType"
use="optional"/>
        <xs:attribute name="maxActionIds" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="AgeOfContentType">
        <xs:attribute name="avgAgeOfContent" type="doubleVectorType" use="optional"/>
        <xs:attribute name="mumberOfSceneEvents" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="minageOfContent" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="maxAgeOfContent" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="SceneUpdateDelayType">
        <xs:attribute name="avgSceneUpdateDelay" type="doubleVectorType" use="optional"/>
        <xs:attribute name="numberOfSceneUpdates" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="minsceneUpdateDelay" type="unsignedIntVectorType" use="optional"/>
```

```
        <xs:attribute name="maxsceneUpdateDelay" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
    <xs:complexType name="MetadataDelayType">
        <xs:attribute name="avgMetadataDelay" type="doubleVectorType" use="optional"/>
        <xs:attribute name="numberOfMetadataMessages" type="unsignedIntVectorType"
use="optional"/>
        <xs:attribute name="minMetadataDelay" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="maxMetadataDelay" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>
<xs:complexType name="DataFrameDelayType">
        <xs:attribute name="avgDataFrameDelay" type="doubleVectorType" use="optional"/>
        <xs:attribute name="numberOfDataFrames" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="minDataFrameDelay" type="unsignedIntVectorType" use="optional"/>
        <xs:attribute name="maxDataFrameDelay" type="unsignedIntVectorType" use="optional"/>
        <xs:anyAttribute processContents="skip"/>
    </xs:complexType>



    <xs:simpleType name="doubleVectorType">
    <xs:list itemType="xs:double"/>
    </xs:simpleType>

    <xs:simpleType name="stringVectorType">
    <xs:list itemType="xs:string"/>
    </xs:simpleType>

    <xs:simpleType name="unsignedLongVectorType">
    <xs:list itemType="xs:unsignedLong"/>
    </xs:simpleType>
</xs:schema>
```

7 Procedures

7.1 Procedures for session establishment

7.1.1 General procedures

The figure 7.1.1-1 indicates the general procedures of split rendering session establishment including session setup and negotiation.

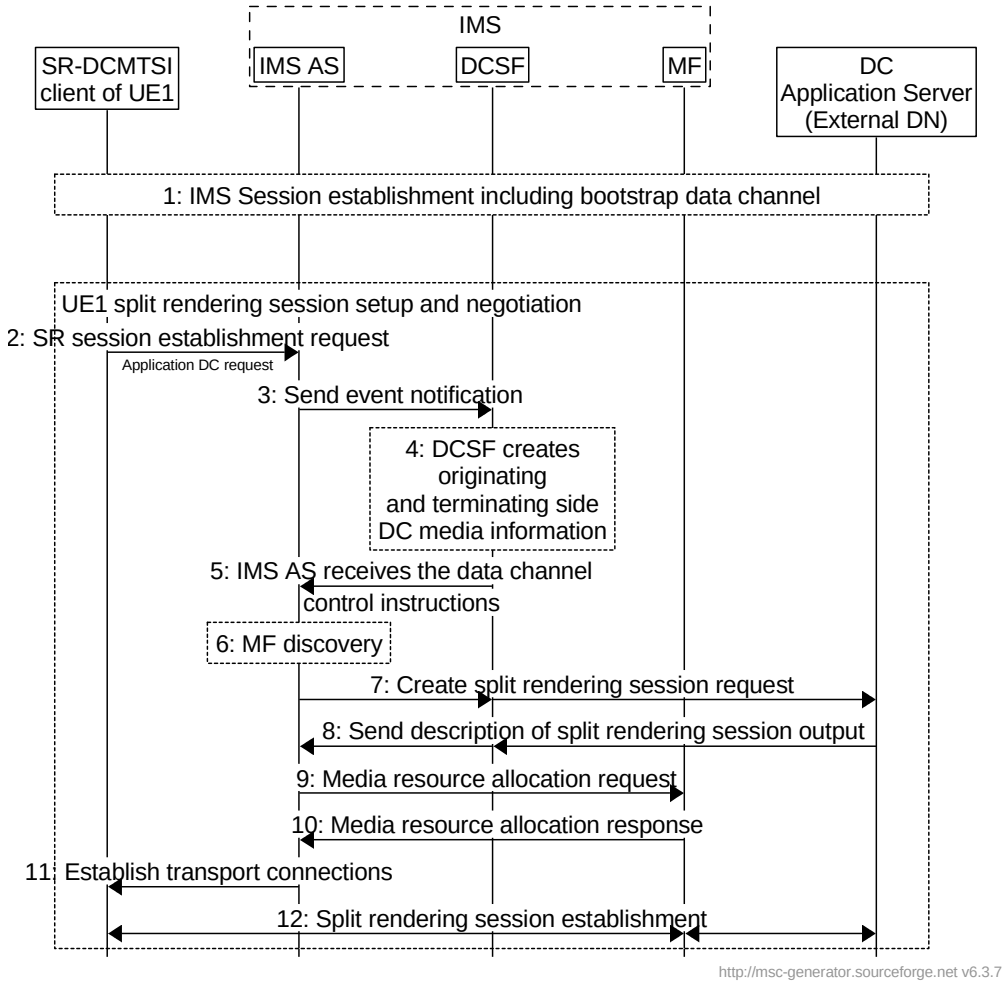

Figure7.1.1-1: General procedures for split rendering session establishment

The steps are as follows:

1: The UE1 initiates an IMS session including bootstrap data channel establishment with the originating MF.

2: UE1 sends a request to IMS AS to establish an application data channel for a split rendering session:

The UE1 can initiate a request of split rendering session establishment if its media capabilities cannot meet the related media rendering requirements. Then the UE1 calculates which objects can be rendered by itself based on its status and decides which part of the objects to be rendered in the UE1 and the others to be rendered in the IMS network.

3: The IMS AS sends DCSF the event notifications including split rendering related information.

4: The DCSF receives event notifications from the IMS AS and processes the session establishment request based on the information in the notification (i.e. associated split rendering related information). The DCSF manages (if applicable) application data channel resources at the MF which meet the split rendering request, to instruct IMS AS to terminate the media flow of the UE1 to the MF.

5 and 6: The IMS AS receives the data channel control instructions from the DCSF and accordingly interacts with the MF via DC2.

7: The IMS AS sends a split rendering session establishment request to the DC AS via the DCSF, , the request may include the information of the objects to be rendered in IMS network.

8: The DC AS sends a description of the split rendering output to the IMS AS via the DCSF.

9: The IMS AS sends the media resource allocation request to the MF, to reserve XR media rendering resource for the UE1.

10: When the resources are allocated successfully, the MF returns a successful response to the IMS AS.

11: The IMS AS returns a successful response to the UE1.

12: Successful split rendering session is established between UE1 and MF through the application data channel.

7.1.2 Procedures for P2P session establishment

The figure 7.1.2-1 indicates the procedures of split rendering session establishment for Person to Person (P2P) scenarios.

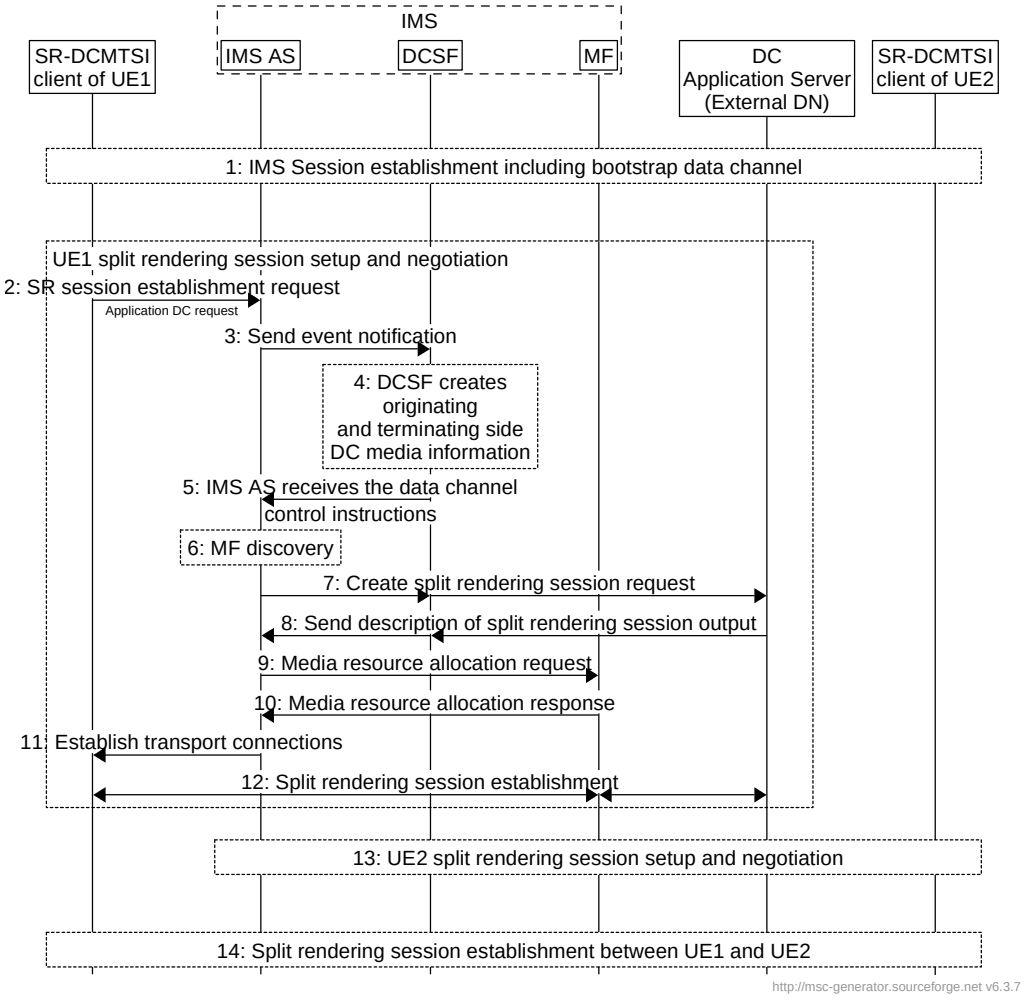

Figure7.1.2-1: Procedures for P2P split rendering session establishment

The steps are as follows:

1: IMS session establishment between UE1 and UE2, including bootstrap session establishment by UE1 and UE2 with originating MF.

2-12: Split rendering session establishment between UE1 and originating MF according to the steps 2-12 of clause 7.1.1.

13: Split rendering session established for UE2 between the originating MF and UE2. Similarly, seps 2 to 12 are followed for UE2. Step 2 may be a reply to the application data channel establishment (re)INVITE from UE1.

14: Successful split rendering session is established between UE1 and UE2 through the application data channel anchored by the MF.

7.1.3 Procedures for P2A(/2P) session establishment

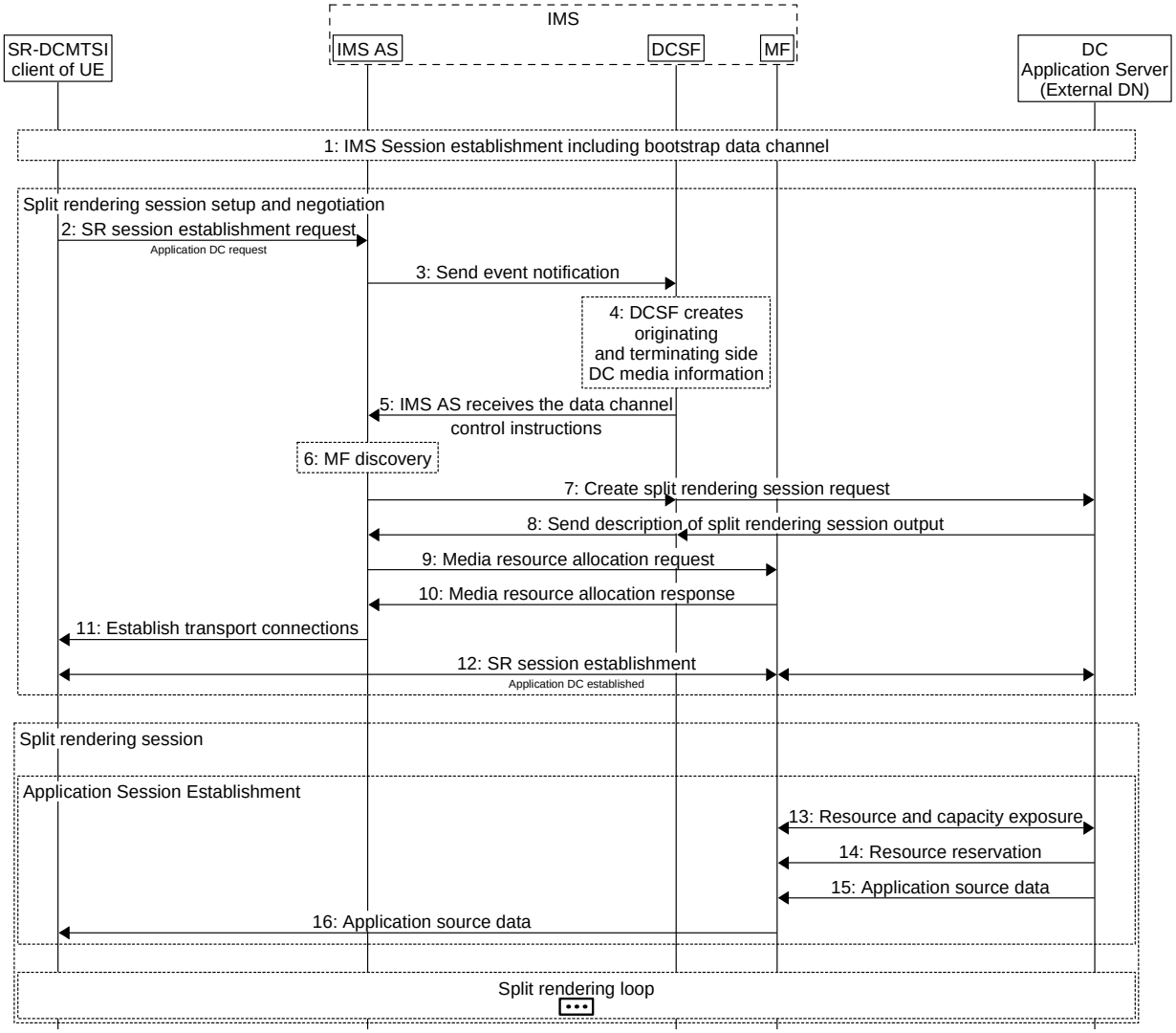

Figure7.1.3-1: Procedures for P2A(/2P) split rendering session establishment

The steps are as follows:

1: IMS session establishment between UE1 and MF, including bootstrap session establishment by UE1 and UE2 with originating MF.

2-12: Split rendering session establishment between UE1 and originating MF, Same as in steps 2-12 of clause 7.1.2.

13: The MF exposes its current capabilities and resources to the DC AS.

The information exposed may be the hardware and software stack and the resources currently available at the MF.

14: DC AS selects resources at the MF and asks MF to reserve these resources.

15: DC AS transfers application source data, for example scripts and scene graph, graphical assets and split logic to the MF.

16: UE relevant application source data like scene graph, application and split logic is sent to the UE.

7.2 Procedures for session modification

7.2.1 General procedures

A UE may trigger split rendering session modification during an established split rendering session e.g., if the current session no longer meets the media rendering requirements. If the existing MF does not meet the new requirements, the session is migrated to a new MF. The session modification procedure in such a case is shown in Figure 7.2.1-1.

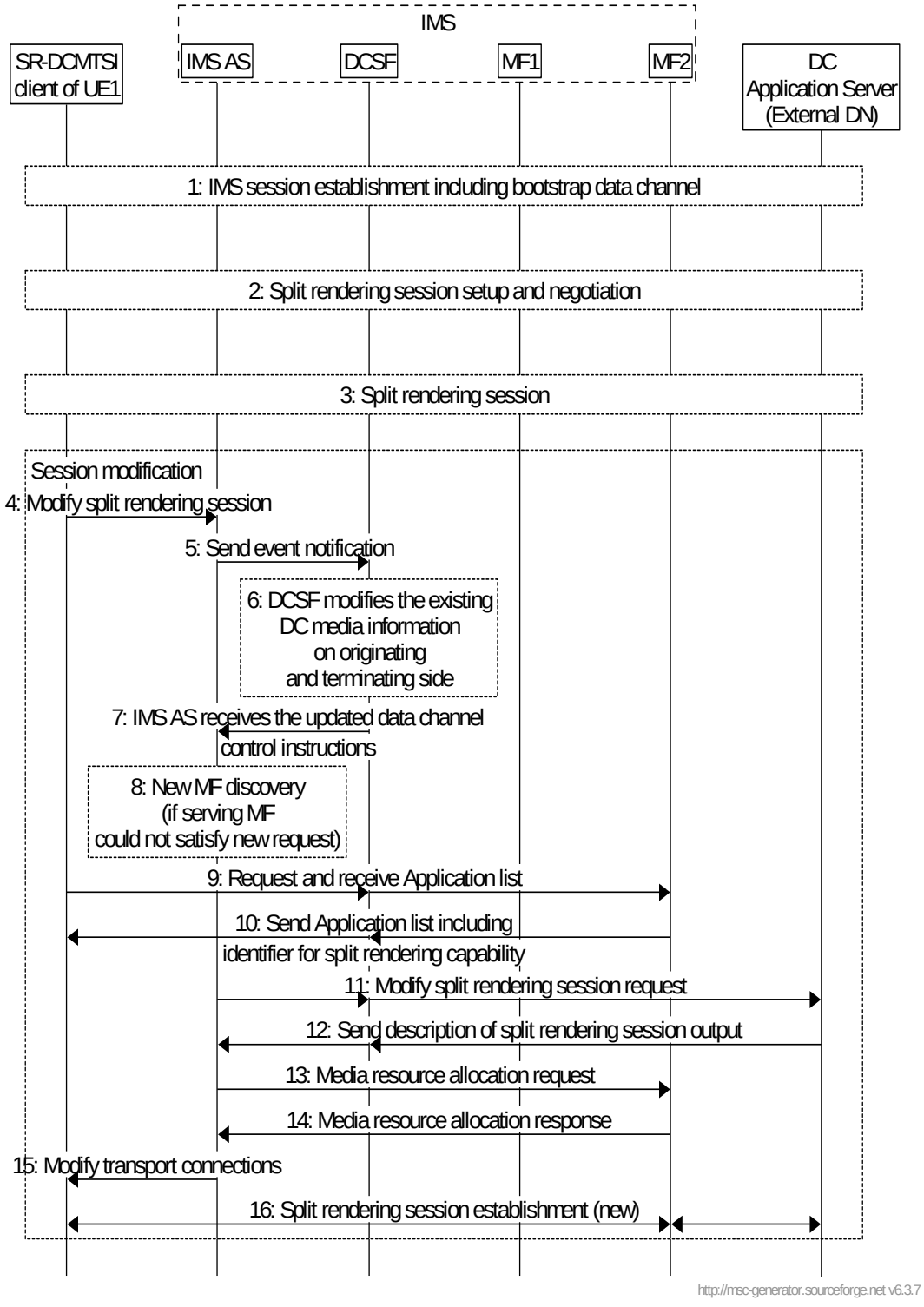

Figure7.2.1-1: General procedures for session modification

The steps are as follows:

1: IMS session establishment including bootstrap data channel.

2: Split rendering session establishment between UE1 and the MF1 (details see clause 7.1.1).

3: Split rendering session is running.

4: During a running split rendering session, when the UE1 discovers that its media capabilities cannot meet the related media rendering requirements, the UE1; can send a request to modify the split rendering session to the IMS AS.

5: The IMS AS sends updated event notifications to DCSF to modify the current split rendering session.

6: The DCSF receives event notification from the IMS AS and process the session modification request based on the information in the received notification (i.e. associated split rendering related information). The DCSF manages (if applicable) application data channel resources at the MF which meet the new split rendering request; to instruct IMS AS to terminate the media flow of the UE1 to the new MF(s).

7: The IMS AS receives the updated data channel control instructions from the DCSF.

NOTE: Here it is assumed that there are more than one MF in the network. It is also assumed that the serving MF does not satisfy the new request (due to unavailability).

8: The (new) MF2 is discovered (if the existing serving MF1 does not satisfy the UE1 request for session modification).

9: UE1 sends application request to the newly discovered MF2 to request an application list.

10: Accordingly, the MF2 responds with application list including an identifier that if the application is capable to be split rendered. DCSF receives the list and forward to the UE1.

11: The IMS AS sends a split rendering session modification request to the DC AS via DCSF through the established application data channel, the request may include the information of the objects to be rendered in IMS network.

12: The DC AS sends a description of the split rendering output to the IMS AS via DCSF.

13: The IMS AS sends the media resource allocation request to the MF2, to reserve media rendering resource for the UE1.

14: When the resources are allocated successfully, the MF2 returns a successful response to the IMS AS.

15: The IMS AS returns a successful response to the UE1.

16: The modified split rendering session is established between UE1 and MF2 through the data channel.

NOTE: MF1 serves the split rendering session until MF2 takes over.

7.2.2 Procedures for P2P session modification

The figure 7.2.2-1 indicates the procedures of split rendering session modification for P2P scenarios.

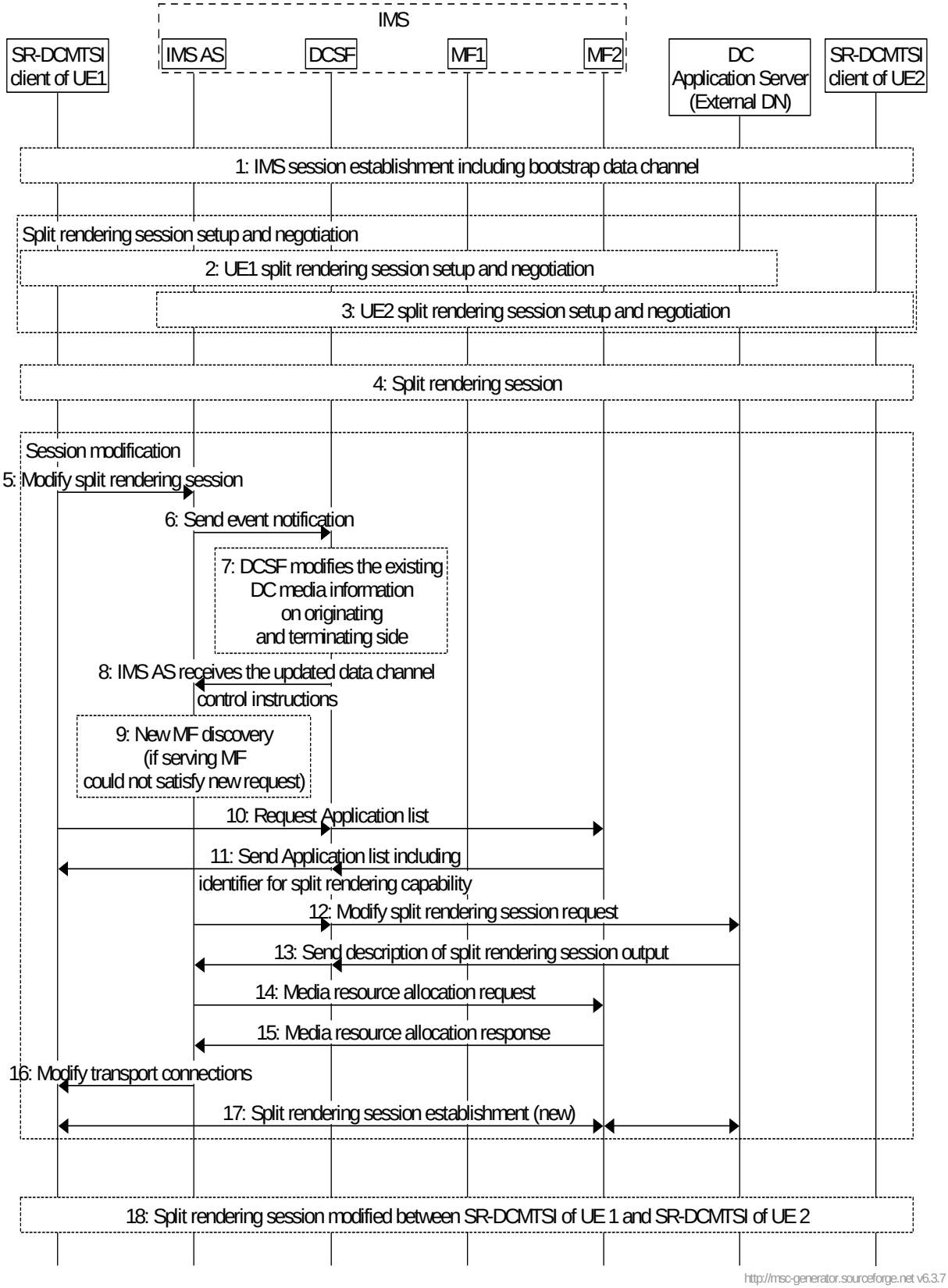

Figure 7.2.2-1: Procedures for P2P split rendering session modification

The steps are as follows:

1: IMS session establishment between UE1 and UE2, including bootstrap session establishment by UE1 and UE2 with originating MF.

2 to 4: Split rendering session establishment between UE1 and UE2 anchored by MF1 (details see clause 7.1.2).

5 to 17: The split rendering session is modified; the new session is established between UE1 and MF2 through the data channel.(details see clause 7.2.1).

18: The new split rendering session is established between UE1 and UE2 anchored by MF2.

7.3 Network support procedures

7.3.1 General procedures

An SR-DCMTSI client or an MF may trigger further procedures during a split rendering, one such procedure may be to adapt the split of rendering operations between the SR-DCMTSI client and the MF during a split rendering session. This split may be due to change in operating conditions of the split rendering session, for example operating conditions of the UE, the MF or changes in the application or scene being rendered, for example changes in the scene description. Split adaptation may include data exchange, for example, exchange of adaptation messages, application state information and assets needed for the split rendering of an DC application. The following generic procedure shall apply, while the exact details may depend on the DC-application being rendered.

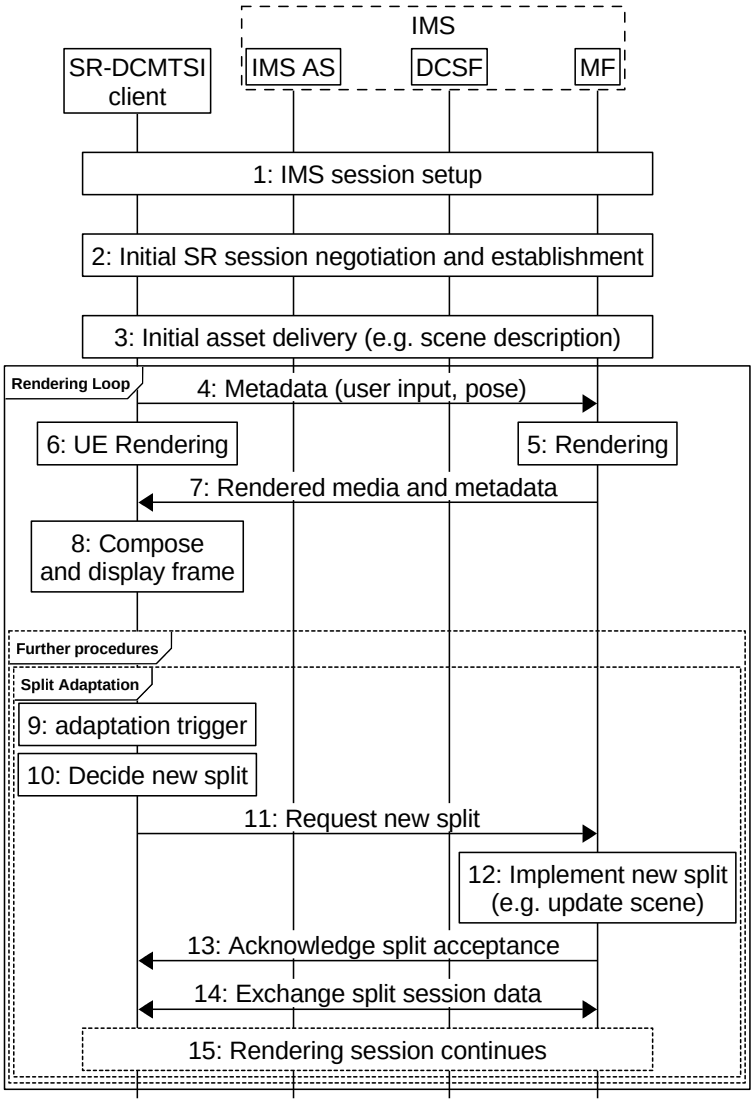

Figure 7.3.1-1: General procedures for adaptation of split rendering

The steps are as follows:

1: The IMS session is established between the SR-DCMTSI client in terminal and a terminating SR-DCMTSI Client which may be in a terminal. For P2P calls, procedures in clause 7.1 are followed.

2: A split rendering session is set up between the SR-DCMTSI client and a serving MF.

3: Assets related to the application being split rendered may be delivered to participants of the split rendering session. The asset delivery may include javascript assets, scene descriptions and graphical objects needed for the session.

Rendering Loop:

The rendering loop is executed continuously during the duration of the split rendering session, for each frame.

4: The SR-DCMTSI client in terminal sends metadata required for rendering to the MF. The metadata may include pose, pose predictions, user inputs etc.

5 and 6: The SR-DCMTSI client in terminal and the MF render the frame.

7: The frame rendered by the MF is transmitted to the SR-DCMTSI client in terminal as well as possible metadata.

8: The SR-DCMTSI client in terminal composes a display frame from the received rendered media and media rendered locally.

NOTE: Steps 5,6,7 although ordered above, may occur in any order. Step 8 may include pose-correction. Step 8 and step 6 may be executed as a single step.

Further Procedures:

Split Adaptation:

NOTE 1: Split Adaptation refers to adaptation of split rendering operations in an ongoing split rendering session between the SR-DCMTSI client and MF, without impacting the MF resources provisioned by IMS AS in step 9 of clause 7.1.1.

NOTE 2: Adaptation of split rendering may be used for interactive objects that react to user actions, pose, eye gaze, eye status, etc. The metadata (e.g. level of interaction, and actions as well as status of the user) may change during the lifetime of the session, the rendering is appropriately adapted according to the updated metadata. An SR-DCMTSI client that supports the adaptive split rendering based on eye status information supports the message format defined in Annex A.2.6.

9: A trigger to adapt the split occurs at the SR-DCMTSI client in terminal; the trigger may be, for example, a change in available UE resources (e.g. battery, compute) or updated metadata, changes in QoE of the split rendering session, changes in the scene/application being rendered.

10: The SR-DCMTSI client in terminal decides if a new split of the rendering operations is needed and determines the new split.

11: The SR-DCMTSI client in terminal sends a request to the MF to adapt the split to the new split.

12: The MF actuates the new split of the rendering operations.

13: The MF sends an acknowledgment of the new split to the SR-DCMTSI client in terminal.

14: The MF and UE may exchange messages and data to support the new split of operations. This may include exchange of messages, for example, for synchronization of the state of the scene being split rendered or exchange of assets, for example, those in Step 3.

The meta-data messages exchanged shall follow the formats specified in clause 5.4.

15: The rendering loop (steps 4 through 8) continues.

NOTE: Split adaptation is shown to be initiated by the SR-DCMTSI client in terminal for clarity, the procedure may be triggered by the MF. Further, other procedures to actuate the new split may be executed during the split rendering session.

7.3.2 Processing Delay adaptation based on QoE metrics

An SR-DCMTSI client or an MF may trigger further procedures during a split rendering. An additional procedure may include adjusting various round-trip delays between the SR-DCMTSI client and the MF during a split rendering session. The delay adaptation information may be sent from the SR-DCMTSI client to the MF periodically when the measured QoE metrics (e.g., poseToRenderToPhoton, roundtripInteractionDelay) goes out of the target delay range. Upon receiving the notification, an MF may adjust the delay involved in some of the processing tasks. For example, an MF may change the Level of Detail (LoD) of the objects in an XR scene which impacts the processing complexity of the XR scene. Delay adaptation procedure may include data exchange, for example, exchange of information messages for delay adaptation. The following generic procedure may apply, while the exact details may depend on the DC-application being rendered.

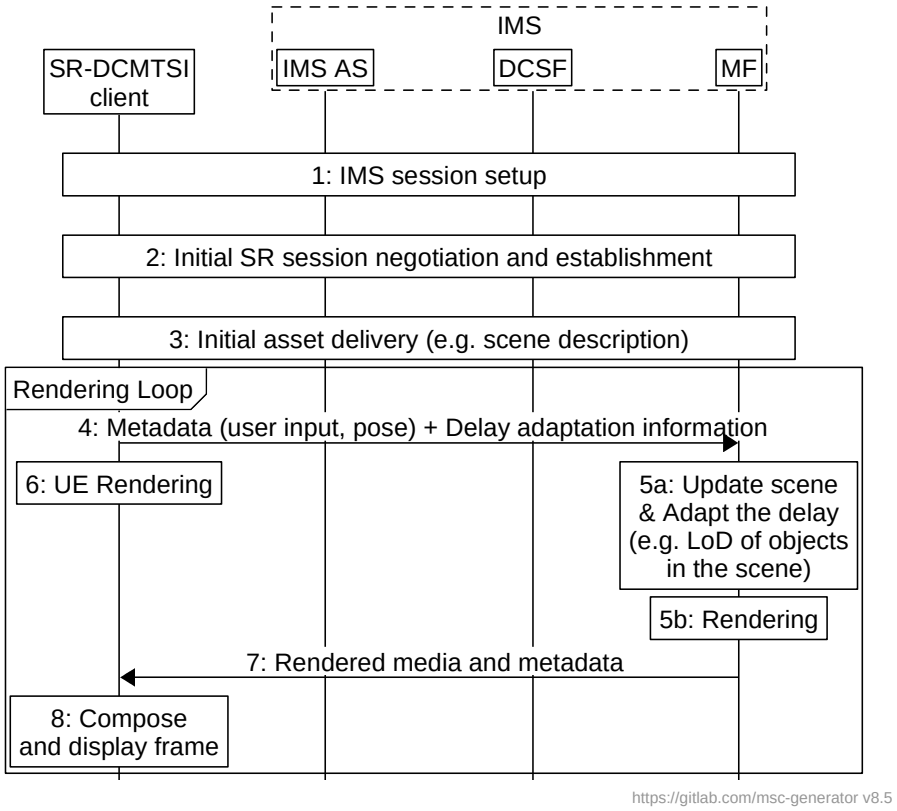

Figure 7.3.2-1: General procedures for processing delay adaptation based on QoE metrics information

The steps are as follows:

1 to 3: As described in clause 7.3.1.

The steps for processing delay adaptation based on QoE metrics information during the rendering loop are as follows:

The SR-DCMTSI client measures and collects the QoE metrics negotiated in the delay adaptation configuration message. The QoE metrics considered for delay adaptation may contain all or a subset of the QoE latency metrics (e.g., poseToRenderToPhoton, roundtripInteractionDelay) negotiated in the metrics configuration message in clause 6.3.1.

4: The SR-DCMTSI client in terminal sends metadata required for rendering to the MF. In addition to the pose, pose predictions and user input metadata, the metadata may include a delay adaptation information message to the MF. The trigger to send the delay adaptation information message is based on the configured periodicity to inform the MF that the measured delay of a QoE latency metric is out of the target delay range. For example, the measured poseToRenderToPhoton QoE latency metric goes out of the target delay range due to new network conditions.

5a: The MF may adjust the processing delay based on the delay adaptation information message.

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

6 and 5b: The SR-DCMTSI client in terminal and the MF render the frame.

7 and 8: As described in clause 7.3.1.

When processing delay adaptation procedure is used, the SR-DCMTSI client in terminal may render some objects of the scene and an MF may render other objects. The SR-DCMTSI client in terminal, may determine and change the LoD of the locally rendered objects to adjust the processing delay.

The SR-DCMTSI client in terminal may use the asset request message defined in clause A.2.7 to request the asset with one or more corresponding LoDs. The asset request message for one or more LoDs can include a list of nodes of a scene graph corresponding to the Level-of-Details of that asset (object).

7.3.3 Asset delivery

An IMS DC application is defined in TS 23.228 [2] as a webpage with content including HTML and javascript and possibly images and style sheets. An IMS DC application is downloaded from the network to the UE through the bootstrap data channel. As such when a split rendering session is established, the split rendering application downloaded to the UE does not have a scene graph corresponding to the scene to be split rendered. However, the HTML page of a split rendering DC application may have a reference to the scene graph or scene description resource. The procedure below in Figure 7.3.3-1 shall be followed for asset delivery.

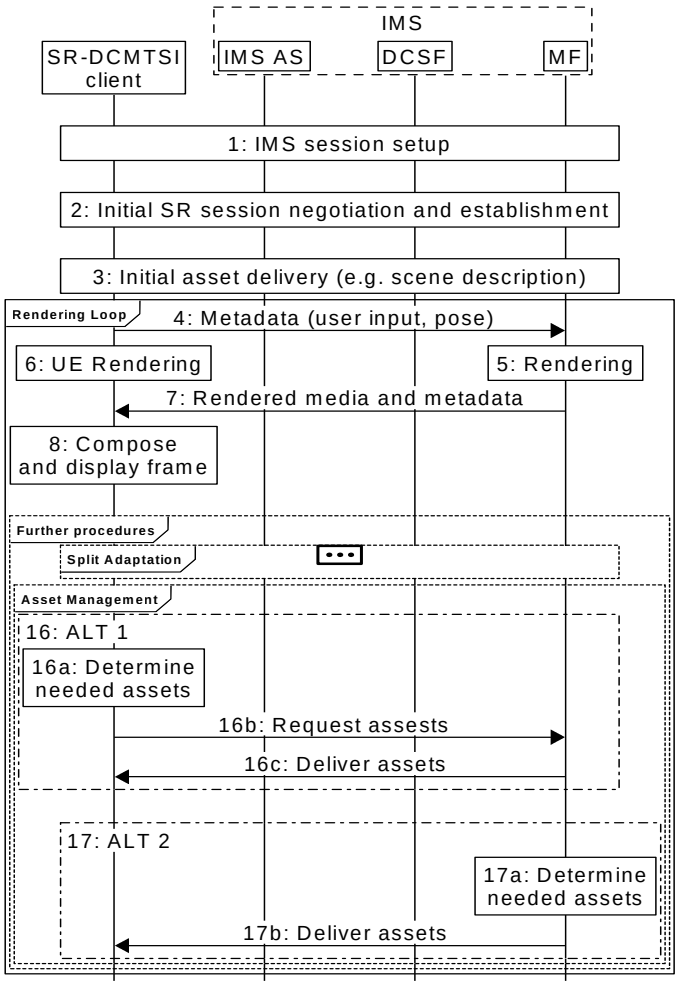

Figure 7.3.3-1 Asset Delivery

The steps are as follows:

1 and 2: As described in 7.3.1.

3: The UE sends a request to the MF for initial delivery of assets based on the agreed split. The request shall conform to the message and payload format defined in Annex A.2.7.

4-15: As described in 7.3.1.

During a split rendering session, the rendering loop at the UE may reach a state where objects to be rendered locally by the UE are not available.

Parallel to the frame rendering loop the IMS DC application being split rendered manages the assets needed for coherent rendering of frames.

Asset management:

16. UE Centric Alternative:

16a. UE1 determines which assets are needed by the UE in near future for local rendering, for example, based on an agreed adaption split and locally available assets.

16b. The UE requests the identified assets from the MF

16c. The MF delivers the requested assets.

17. Network Centric Alternative:

17a. The MF determines which assets are needed by UE1 in near future for local rendering at the UE1. The MF may do this based on a record of assets delivered to UE1 and an agreed adaptation split.

17b. The MF delivers the requested assets.

Annex A (normative): Metadata Formats and Message Types

A.1 General

A.1.1 Overview of Metadata Formats and Message Types

This annex defines the metadata and metadata message types supported by this specification. Metadata formats and meta data channel message types and formats supported in this specification re-use or modify formats and message types defined in other 3GPP specifications as defined in Table A.1.1-1.

Table A.1.1-1 Formats and relationship with 3GPP specifications

Format	Source specification	Clause in source specification	Modified for this specification	Clause in this specification	urn
Metadata Data Channel Message	TS 26.565 [5]	8.3.3	No	5.4.3, A.1.2	N/A
Split Rendering Configuration	TS 26.565 [5]	8.4.2	Yes	A.1.3	urn:3gpp:split-rendering:v2:sr-configuration
Pose	TS 26.119 [6]	12.2	No	5.4.2.2	urn:3gpp:split-rendering:v1:pose
Action	TS 26.119 [6]	12.3	No	5.4.2.3	urn:3gpp:split-rendering:v1:action
Split Adaptation Configuration	TS 26.565 [5]	C.2.3.1	Yes	A.2.3.1	N/A
Split Adaptation Message	TS 26.565 [5]	C.2.3.2	Yes	A.2.3.2	urn:3gpp:split-rendering:v2:asrp:sr-split
State Synchronization Message	TS 26.565 [5]	C.2.3.3	Yes	A.2.3.3	urn:3gpp:split-rendering:v2:sr-state
Seamless Adaptive Split	TS 26.565 [5]	C.2.3.2	Yes	A.2.4	urn:3gpp:split-rendering:v1:asrp:sr-split-seamless
Processing Delay Adaptation Configuration	N/A	N/A	N/A	A.2.5	urn:3gpp:split-rendering:v1:daqoe:configuration
Processing Delay Adaptation Information	N/A	N/A	N/A	A.2.5	urn:3gpp:split-rendering:v1:daqoe:information
Adaptive split rendering with eye status information	N/A	N/A	N/A	A.2.6	urn:3gpp:split-rendering:v1:sr-split-eyeinfo
Asset Request	N/A	N/A	N/A	A.2.7	urn:3gpp:split-rendering:v1:asrp:sr-asset

NOTE: The metadata formats specified in Table A.1.1-1, if modified from the source specification for this specification may not be interoperable with the source specification.

A.1.2 Metadata Message Format

The data channel messages shall conform to the format in Table A.1.2-1.

Table A.1.2-1 Split Rendering Metadata Messages Format

Name	Type	Cardinality	Description
messages	Array(Message)	1..n	A list of split rendering metadata messages. Each message shall be formatted according to the Message data type as defined in Table

Each message shall conform to the format in Table A.1.2-2.

Table A.1.2-2 Split Rendering Metadata Message Data Type

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
Type	string	1..1	A urn that identifies the message type.
Message	object	1..1	The message content depends on the message type.
sendingAtTime	number	0..1	The time when the split rendering metadata message is transmitted from the split rendering client to the split rendering server.

A.1.3 Split Rendering Configuration

The SR-DCMTSI client shall send a split rendering session configuration information to the MF and if applicable, to the DC AS after successful establishment of a split rendering session and before starting the rendering loop. The session configuration shall be in JSON format and shall follow the format in Table A.1.3-1.

Table A.1.3-1 Split Rendering Configuration Format

Name	Type	Cardinality	Description
renderingFlags	Array(SR_CONFIG_FLAGS)	0..1	Provides a set of flags to activate/deactivate selected rendering functions. The defined SR_CONFIG_FLAGS are: FLAG_ALPHA_BLENDING FLAG_DEPTH_COMPOSITION FLAG_EYE_GAZE_TRACKING
splitRenderingMFProfile	array(URN)	0..1	A list of supported split-rendering service profile identifiers for the MF. The profile identifiers are listed in clause 4.5.2.2 for each profile.
splitRenderingFeatures	array(SR_FEATURE_FLAGS)	0..1	A list of split-rendering features supported by the SR-DCMTSI client. The supported features flags are: ADAPTIVE SEAMLESS_ADAPTIVE DELAY_ADAPTIVE
deviceCapabilities	Object	0..1	If the SR-DCMTSI client is implemented by a device defined in TS 26.119 [4], clause 6.2.5., device capabilities may be listed here.
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.
extraConfigurations	Object	0..1	A placeholder for addition configuration information.

A.2 Message Types

A.2.1 Pose

The pose messages shall be conformant to the pose message format identified by “urn:3gpp:split-rendering:v1:pose” as defined in TS 26.119 [6].

A.2.2 Action

The action messages shall be conformant to the action message format identified by “urn:3gpp:split-rendering:v1:action” as defined in TS 26.119 [6].

A.2.3 Split Adaptation

A.2.3.1 Configuration format

The configuration format defined in Annex A.1.3 with the additional fields defined below in Table A.2.3.1-1 shall be used for split rendering configuration exchange for adaptive split rendering.

Table A.2.3.1-1 Adaptive Split Rendering Configuration Format

Name	Type	Cardinality	Description
renderingSplit	Object	1..1	An object identifying objects to be rendered and where they are to be rendered (MF or UE). The message shall be a dictionary object. with keys “MF” and “UE”, and values corresponding to a key shall be a list of named nodes from the scene description being rendered in the SR session. The keys shall indicate where the objects named in the corresponding value list are rendered.
synchronizedStatesInit	Object	1..1	An object identifying states to be synchronized between the MF and UE and their initial state
states	Object	1..1	A list of state identifiers, their current values
state	String/number	1..n	Identifier of a state
initVal	String	1..n	Initial value of the state
stateVals	Array	1..1	An array of values possible for the state

The synchronizedStatesInit object is used during split rendering session establishment to configure which states are to be synchronized between the MF and the UE. In a scene being split rendered, there may be multiple state machines and for adaptive split rendering only a subset of the states may be affected by the split operations.

A.2.3.2 Split Adaptation Message Format

An SR-DCMTSI client that supports adaptive split rendering shall support the split adaptation message as defined in Table A.2.3.2-1 below based on the split adaptation message defined clause C.2.3.2 of TS 26.565 [5].

Table A.2.3.2-1 Message format for split adaptation messages

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp:split-rendering:v1:asrp:sr-split
message	Object	1..1	Message content
subtype	string	1..1	An identifier of the subtype of the message, it may be a request (REQ) for new split or acknowledgement (ACK), acceptance (OK) or rejection of a request (NOK).
renderingSplitId	string	1..1	An identifier of the rendering split unique within the scope of the SR session
renderingSplit	Object	0..1	A object identifying objects to be rendered and where they are to be rendered (MF or UE). The message shall be a dictionary object. with keys “MF” and “UE”, and values corresponding to a key shall be a list of named nodes from the scene description being rendered in the SR session. The keys shall indicate where the objects named in the corresponding value list are rendered.

Split adaptation messages indicating acceptance, acknowledgment or rejection of a split adaptation request may not include the renderingSplit Object.

A.2.3.3 State Synchronization Message Format

During a split rendering session, various states associated with the scene being rendered may transition. Depending on the nature of the application being executed, a transition may occur at the UE, at the MF or at both the UE and MF. For the application execution to be consistent, some state transitions need to be synchronized between the MF and UE. The UE and MF may agree on which states to synchronize during session setup. To synchronize state transitions during a split rendering session the MF and UE shall exchange messages of the type “urn:3gpp:split-rendering:v2:asrp:sr-state” . The same message type shall be used to send a state synchronization update, acknowledge a state synchronization update or simultaneously send and acknowledge a state synchronization update. The state synchronization update messages shall be conformant with the meta-data message format defined in A.1.1 and the message content shall be formatted as shown in Table A.2.3.3-1.

Table A.2.3.3-1 Message format for state synchronization messages

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp:split-rendering:v2:sr-state
message	Object	1..1	Message content
subtype	string	1..n	An identifier of the subtype of the message, it may be a state synchronization update (SYNC), acknowledgment (ACK) or both (SYNC_ACK)
syncUpdateId	string	1..1	An identifier of the synchronization update unique within the scope of the SR session
synchronizedStates	Object	1..1	An object identifying states that are synchronized between the MF and UE and their current state. Only states that have transitioned may be exchanged
states	Object	1..1	A list of state identifiers, their current values and last change time
identifier	String/number	1..n	Identifier of a state
val	Object/String/number	1..n	Value of the state
lastChangeTime	number	1..1	The timestamp of the last change in state

Split adaptation messages indicating an acknowledgment of a state update may not include the synchronizedStates Object.

A.2.4 Seamless Adaptive Split

An SR-DCMTSI client that supports the adaptive split rendering with seamless adaptation shall support the seamless adaptive split message format defined below.

Table A.2.4-1 Message format for seamless adaptive split messages

Name	Type	Cardinality	Description
Id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp: split-rendering: v1: asrp:sr-split-seamless
message	Object	1..1	Message content
subtype	string	1..1	An identifier of the subtype of the message, it may be a request (REQ) for new split or acknowledgement (ACK), acceptance (OK) or rejection of a request (NOK).
renderingSplitId	string	1..1	An identifier of the rendering split unique within the scope of the SR session
renderingSplit	Object	0..1	An object identifying objects to be rendered and where they are to be rendered (MF or UE). The message shall be a dictionary object. with keys “MF” and “UE”, and values corresponding to a key shall be a list of named nodes from the scene description being rendered in the SR session. The key ‘UE is used for objects that are to be rendered by the UE and key ‘MF is used for objects that are to be rendered by the MF, when seamlessSplit conditions are not met.
seamlessSplit	object	0..1	An object that if present indicates a seamless adaptation of the rendering process when possible.
radius	number	1..1	A distance in meters that defines a sphere centered at the UE, such that preferential rendering is used for objects that lie within this sphere. An object lies within the preferential rendering sphere if for all or some points defining the collider associated with the object the distance from the UE is less than the radius.
type	string	1..1	A string that indicates the type of preferential rendering to be used for the objects defined by the key ‘UE’ in renderingSplit. The following values are supported: “Local”: The objects are rendered at the UE when they are within the sphere define by R and rendered by the MF when they are outside of it. “LOD”: The MF renders the objects when they are within the radius R at a high fidelity. For example, 3D models with a higher Level-of-detail may be rendered.

NOTE: Deterministic calculation of objects to be rendered by the UE, by the MF, and by the DC AS needs to be ensured. The floating point representation used by the UE, MF and the DC AS for physics calculation and latency between the UE, MF and the DC AS may impact determinism of such calculations.

A.2.5 Processing Delay Adaptation based on QoE metrics

A.2.5.1 Configuration format

An SR-DCMTSI client that supports the processing delay adaptation shall support the processing delay adaptation configuration message format as below.

The MF shall share the configuration information of the delay adaptation procedure with the SR-DCMTSI client in terminal during the split rendering session negotiation and establishment processes. The configuration may be updated during the rendering loop. The configuration information of the delay adaptation procedure shall be in JSON format according to the Metadata Data Channel Message Format defined in clause 5.4.3. The message type shall be "urn:3gpp:split-rendering:v1:daqoe:configuration".

Table A.2.5.1-1 – Configuration message format for Processing Delay adaptation based on QoE metrics

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp:split-rendering:v1:daqoe:configuration
message	Object	1..1	Message content
qoeMetrics	array	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 configuration message in clause 6.3.1.
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 defined in clause 6.3.1.
periodicity	string	1..1	The periodicity of the delay adaptation information for that QoE metric. It may be expressed as a multiple of the "Measure-Resolution" defined in clause 16.3.2 of TS 26.114 [7]. Whenever a delay adaptation information message is sent, the SR-DCMTSI client in terminal shall reset its timer to the value of the periodicity property and it shall begin countdown of the timer again.
flexibleObjects	array	0..1	An array of objects for which several LoDs are available for the delay adaptation.
flexibleObjectId	string	1..1	A unique identifier of an object within the scope of the split rendering session. For example, the index of the node of the object in the scene description.
levelIds	Array(string)	1..1	An array of node identifiers in a scene description corresponding to the LoDs of the object identified by the flexibleObjectId.
criteria	string	0..1	An information to guide the application for determining the object and corresponding LoD available in the XR scene. This information can be either: - "VIEWING_DISTANCE": for increasing the LoD (i.e., by increasing the processing delay) for close object(s) if the measured delay is below the target delay and by decreasing the LoD (i.e., by decreasing the processing delay) for far object(s) if the measured delay is above the target delay, - "FIELD_OF_VIEW": for increasing the LoD (i.e., by increasing the processing delay) for object(s) in the center of the FoV if the measured delay is below the target delay and by decreasing the LoD (i.e., by decreasing the processing delay) for object(s) located at the borders of the FoV if the measured delay is above the target delay, - "SCREEN_COVERAGE": for increasing the LoD (i.e., by increasing the processing delay) for object(s) having larger screen coverage if the measured delay is below the target delay and by decreasing the LoD (i.e., by decreasing the processing delay) for object(s) having smaller screen coverage if the measured delay is above the target delay.

NOTE 1: The target delay range for a QoE metric is delimited by a minimum threshold and a maximum threshold delay value. The thresholds and the target delay, for each QoE metric in the qoeMetrics array, can be provided by the DC Application Server to the SR-DCMTSI client in terminal.

NOTE 2: The MF can get the object LoDs information from the scene description. For example, using the MSFT_lod [16] vendor extension to the SD-Rendering-glTF-Core, or the LOD node from the X3D by Web3D [17].

A.2.5.2 Metadata format

During a IMS-based split rendering session, the operating environment of the serving MF, the resources of SR-DCMTSI client or the network conditions may change. Consequently, the roundtrip delay may need to be adjusted to deliver a consistent QoE.

When delay adaptation procedure is enabled, the SR-DCMTSI client in terminal 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 SR-DCMTSI client in terminal may report the measured delay based on the configured periodicity.

An SR-DCMTSI client that supports the processing delay adaptation shall support the processing delay adaptation information message format as below.

The delay adaptation information message format that is used for IMS-based split rendering shall comply with the format defined in Table A.2.5.2-1. The delay adaptation information message shall be carried as part of the data channel messaging mechanism. The metadata data channel message shall be in JSON format according to the Metadata Data Channel Message Format defined in clause 5.4.3. The message type shall be “urn:3gpp:split-rendering:v1:daqoe:information”.

Table A.2.5.2-1 Metadata format for Processing Delay Adaptation information message based on 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 for which delay adaptation is needed. This qoeMetrics array may contain all or a subset of the QoE metrics negotiated in the configuration message in clause A.2.5.1.
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.

A.2.6 Adaptive split rendering with eye status information

If an SR-DCMTSI client that supports the adaptive split rendering with eyes status information, it shall support the message format defined in Table A.2.6-1. The eye status information shared by SR-DCMTSI client to MF during the adaptation procedure shall be in JSON format according to the Metadata Data Channel Message Format defined in clause 5.4.3. The message type shall be “urn:3gpp:split-rendering:v1:sr-split-eyeinfo”.

Table A.2.6-1 Message format for eyes status information

Name	Type	Cardinality	Description
eyesInfo	Object	1..1	An array of eye information objects corresponding to past and current eyes status for the viewer.
eyesStatus	number	1..1	The current eyes status, e.g.: 0: Eyes are open 1: Eyes are closing 2: Eyes are closed 3: Eyes are opening
eyesStatistics	Object	1..1	The current viewer eyes statistics
averageDuration	number	1…3	Array of average eyes duration for the viewer, e.g.: [1] represents average closing time, [2] represents average closed time [3] represents average opening time
averageInterval	number	1..1	Average interval between two eye blinking.
elapsedTime	Number	0…n	Time expressed in milliseconds since the last eyes open status.

A.2.7 Asset Request

An SR-DCMTSI client that supports the split rendering shall support the asset request messages below.

Table A.2.7-1 Message format for asset requests

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp:split-rendering:v1:asrp:sr-asset
message	Object	1..1	Message content
request	string	1..1	A request for assets, identifying the assets requested, for example as a list of nodes of a scene graph or a list of URIs which reference assets for nodes in the scene graph.

A.2.8 Foveated optimizations

A.2.8.1 Introduction

Gaze based optimizations like foveated rendering and foveated encoding reduce resource usage and improve user experience for a given resource budget. If gaze data is available from the SR-DCMTSI client, for example, as gaze predictions , the SR-DCMTSI client and MF or DC AS may use gaze data for gaze-based optimizations in rendering and encoding. For gaze-based optimizations in rendering and encoding, the SR-DCMTSI client and the MF agree on an optimization profile during session negotiation from a list of profiles. An optimization profile provides importance maps based on the gaze position for foveated rendering and encoding. An importance map provides quality information for different regions of a frame, their size and location with reference to a gaze point.

During a split rendering session, the optimization profile being used might need to be adapted or a switch to a different profile might be desired based on, for example, user preference, network conditions, monitored QoE, etc.

A.2.8.2 Configuration format

To use gaze-based optimizations of rendering and encoding, the split rendering configuration shall indicate the gaze-based optimization profile used in the “extraConfigurations” field of the split rendering configuration format specified in Annex A.1.3. The configuration shall be JSON formatted and conform to the format in Table A.2.8.2-1. A gaze-based optimization profile contains importance maps for rendering and encoding a frame according to varying qualities

based on the reported gaze location, which may be a gaze prediction. . During a split rendering session, for each frame, the MF calculates importance maps for rendering and encoding based on the gaze-based optimization profile and the gaze data received from the SR-DCMTSI client, centered around the gaze predicted for the current frame The gaze data received from the SR-DCMTSI client may contain confidence values of the predicted gaze, which may be used in the importance map calculations by the MF.

Table A.2.8.2-1 Configuration format for gaze-based optimization profile.

Name	Type	Cardinality	Description
gazeOptProfile	Object	1..N	An object corresponding to a gaze-based profile. It may be only an identifier such as a platform dependant name of a preset or level, a URI/N or it may comprise all information needed to use the profile, including quality regions, their relative sizes and their assigned quality.
renderingMap	Object	0..1	An object containing quality regions for rendering and their relative size around a gaze location. It may also contain an indication of actual rendering quality to be used for the different quality regions, for example, sampling rates.
name	String	1..1	A session wide unique identifier of the rendering map, for example, “Level 1”, “Level 2”, “Level 3” or “Wide”, “Balanced”, Narrow”. The name may identify a platform dependant preset which has fixed values for size of different regions of the frame and the desired rendering quality.
qualityRegion	Object	0..N	A descriptor of the parameters of a region of a frame to be rendered with a particular importance or quality
name	String	1..1	An identifier of the region, for example, “high quality”, “medium quality”, “low quality”.
size	Float	0..1	A value of the size of the region as normalized radius of a circular region centred at the gaze point.
quality	Float	0..1	A value of the relative importance or quality of the region as a normalized numeric value, where a value of 1 refers to the highest quality
extras	Object	0..N	Additional information about the renderingMap
encodingMap	Object	0..1	An object containing quality regions for encoding and their relative size around a gaze location. It may also contain information about desired relative encoding quality to be used for the different quality regions, for example, QP offsets or importance values to be used by the encoder for rate control.
name	String	1..1	A session wide unique identifier of the encoding map, for example, “Level 1”, “Level 2”, “Level 3” or “Wide”, “Balanced”, Narrow”. The name may identify a preset which has fixed values for size of different regions of the frame and their desired quality.
qualityRegion	Object	0..N	A descriptor of the parameters of a region of a frame to be encoded with a particular relative importance.
name	String	1..1	An identifier of the region, for example, “high quality”, “medium quality”, “low quality”.
size	Float	0..1	A descriptor of the size of the region,as a normalized radius of a circular region centred at the gaze point.
quality	Float	0..1	A value of the relative quality of the region as a normalized numeric value, where a value of 1 refers to the highest quality
extras	Object	0..N	Additional information about the encodingMap

A.2.8.3 Metadata format

Adapting gaze-based optimization being in use in a split rendering session shall follow the general network procedures specified in clause 7.3.1. The metadata message to adapt the gaze-based optimization profile shall conform to the format specified in clause 5.4.3 and shall have the type indicated as “urn:3gpp:split-rendering:v1:asrp:gaze_opt_adapt”. Depending on the implementation and the adaptation needed, the payload may indicate a switch to a new gaze optimization profile or modification of parameters of the current profile. The message shall conform to the format in Table A.2.8.3-1.

Table A.2.8.3-1 Metadata message format for gaze-based optimization adaptation

Name	Type	Cardinality	Description
id	string	1..1	A unique identifier of the message in the scope of the data channel session.
type	string	1..1	urn:3gpp:split-rendering:v1:asrp::asrp:gaze_opt_adapt
message	Object	1..1	Message content
subtype	string	1..1	An identifier of the subtype of the message, it may indicate a switch in optimization profile (SwitchOptProf) or a change in parameters of the optimization profile (ModOptProf)
gazeOptProfile	Object	1..1	An object corresponding to a gaze-based profile. It may be only an identifier such as a URI/N or it may comprise all information needed to use the profile. If the message subtype is to switch a profile, this contains or points to the gazeOptProf to switch to. If the message subtype is ModOptProfile, this contains or points to a modified version of the current optimization profile.

Annex B (informative): Change history

Change history							
Date	**Meeting**	**TDoc**	**CR**	**Rev**	**Cat**	**Subject/Comment**	**New version**
04-2024	127b-e	S4-240645				[SR_IMS]TS 26.567 Skeleton	0.0.1
04-2024	127b-e	S4-240790				[SR_IMS] version agreed during SA4#127bis-e (including S4-240583 and S4-240651)	0.1.0
05-2024	128	S4-241272				[SR_IMS] version agreed during SA4#128 (including S4-241160, 241161, 241274 and S4-241213)	0.2.0
08-2024	129-e	S4-241739				[SR_IMS] version agreed during SA4#129 (including S4-241550, 241731, 241735 and S4-241728)	0.3.0
11-2024	130	S4-241953				[SR_IMS] version included agreement during Post SA4#129 RTC SWG Telco (S4aR240081, S4aR240063)	0.3.1
11-2024	130	S4-242172				[SR_IMS] version agreed during SA4#130 (including S4-241947, 241962, 241963, 241972, 242041, 242043 and S4-242092)	0.4.0
02-2025	131	S4-250089				[SR_IMS] version included agreement during Post SA4#130 RTC SWG Telco (S4aR250009 with editor's note in meeting minutes, S4aR250056, S4aR250057, S4aR250061, S4aR250062, S4aR250069)	0.4.1
02-2025	131	S4-250300				[SR_IMS] version agreed during SA4#131 (including S4-250268, S4-250200, S4-250304, S4-250305, S4-250306, S4-250199) with editorial corrections.	0.5.0
02-2025	131	S4-250404				[SR_IMS] editorial corrections	0.5.1
03-2025	SA#107					Version 1.0.0 created by MCC for presentation to TSG SA	1.0.0
04-2025	131b-e	S4-250455				[SR_IMS] version included agreement during Post SA4#131 RTC SWG Telco (S4aR250082).	1.0.1
04-2025	131b-e	S4-250674				[SR_IMS] version agreed during SA4#131b-e (including S4-250499, S4-250520, S4-250641, S4-250666, S4-250733) with editorial corrections.	1.1.0
05-2025	132	S4-251091				[SR_IMS] version agreed during SA4#132 (including S4-250893, S4-251086, S4-251087, S4-251110, S4-251140) with editorial corrections.	1.2.0
06-2025	RTC SWG post 132	S4aR250115				[SR_IMS] updated version with editorial corrections.	1.2.1
07-2025	133	S4-251469				[SR_IMS] updated version (including S4-251359) with editorial corrections.	1.3.0
09-2025		SP-250933				Version 2.0.0 created by MCC to be sent to TSG SA for approval	2.0.0
09-2025	SA#109					Version 19.0.0 created by MCC upon approval by TSG SA#109	19.0.0

History

Document history		
V19.0.0	October 2025	Publication