|  |
| --- |
| 3GPP TS 26.113 V1.2.0 (2024-04) |
| Technical Specification |
| 3rd Generation Partnership Project;Technical Specification Group Services and System Aspects;Real-Time Media Communication; Protocols and APIs(Release 18) |
|  |
|  |  |
|  |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP.The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented.This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification.Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. |

|  |
| --- |
|  |
| ***3GPP***Postal address3GPP support office address650 Route des Lucioles - Sophia AntipolisValbonne - FRANCETel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16Internethttps://www.3gpp.org |
| ***Copyright Notification***No part may be reproduced except as authorized by written permission.The copyright and the foregoing restriction extend to reproduction in all media.© 2023, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).All rights reserved.UMTS™ is a Trade Mark of ETSI registered for the benefit of its members3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational PartnersLTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational PartnersGSM® and the GSM logo are registered and owned by the GSM Association |

Contents

Foreword 6

Introduction 7

1 Scope 8

2 References 8

3 Definitions of terms, symbols and abbreviations 9

3.1 Terms 9

3.2 Symbols 9

3.3 Abbreviations 9

4 Procedures for real-time media communication 10

4.1 General 10

4.2 Procedures for media session handling 10

4.2.1 Provisioning (RTC-1) procedures 10

4.2.2 Network media session handling (RTC-3, RTC-5) procedures 10

4.2.3 UE media session handling (RTC-6) procedures 10

4.3 Procedures for media content and signalling transport 10

4.3.1 Media-centric transport (RTC-4) procedures 10

4.3.2 UE media delivery (RTC-7) procedures 11

5 General aspects of APIs 11

5.1 Usage of HTTP 11

5.1.1 HTTP protocol version 11

5.1.1.1 RTC AF 11

5.1.2 HTTP message bodies for API resources 11

5.1.3 Usage of HTTP headers 11

5.1.3.1 General 11

5.1.3.2 Media Session Handler identification 11

5.1.3.3 RTC AF identification 11

5.1.3.4 Support for conditional HTTP GET requests 11

5.1.3.5 Support for conditional HTTP POST, PUT, PATCH and DELETE requests 11

6 Provisioning interface (RTC-1) 12

6.1 General 12

6.2 Provisioning Sessions API 12

6.3 Configuration Provisioning API 12

6.4 Consumption Reporting Provisioning API 13

6.5 Edge Resources Provisioning API 13

6.6 Policy Templates Provisioning API 13

6.7 Metrics Reporting Provisioning API 13

7 Media hosting interface (RTC-2) 16

8 RTC AS to RTC AF interface interface (RTC-3) 16

9 Media-centric transport interface interface (RTC-4) 16

10 Control transport interface interface (RTC-5) 17

10.1 General 17

10.2 Service Access Information API 17

10.3 Configuration Information API 17

10.4 Dynamic Policies API 17

10.5 Network Assistance API 18

10.6 Metrics Reporting API 18

10.7 Consumption Reporting API 18

11 Client API (RTC-6) 18

12 Client interface (RTC-7) 18

13 Protocols of real-time media communication 18

13.1 General 18

13.2 WebRTC signalling protocol 19

13.2.1 General 19

13.2.2 Protocol and version identification 19

13.2.3 WebSocket URI structure 19

13.2.4 SWAP 20

13.2.4.1 Protocol and version identification 20

13.2.4.2 Transport 20

13.2.4.3 State machine 20

13.2.4.4 Message syntax and semantics 22

13.2.4.4.1 Common message fields 22

13.2.4.4.1.1 Source (source) 22

13.2.4.4.1.2 Message Identifier (messge\_id) 22

13.2.4.4.1.3 Message Type (message\_type) 23

13.2.4.4.2 Register message 23

13.2.4.4.2.1 Description 23

13.2.4.4.2.2 Parameters 23

13.2.4.4.3 Response message 24

13.2.4.4.3.1 Description 24

13.2.4.4.3.2 Parameters 24

13.2.4.4.4 Connect message 24

13.2.4.4.4.1 Description 24

13.2.4.4.4.2 Parameters 24

13.2.4.4.5 Accept message 24

13.2.4.4.5.1 Description 24

13.2.4.4.5.2 Parameters 24

13.2.4.4.6 Update message 24

13.2.4.4.6.1 Description 24

13.2.4.4.6.2 Parameters 25

13.2.4.4.7 Reject message 25

13.2.4.4.7.1 Description 25

13.2.4.4.7.2 Parameters 25

13.2.4.4.8 Close message 25

13.2.4.4.8.1 Description 25

13.2.4.4.8.2 Parameters 25

13.2.4.4.9 Application message 25

13.2.4.4.9.1 Description 25

13.2.4.4.9.2 Parameters 25

13.2.4.5 Integrity and security 25

13.2.4.6 JSON schema 26

13.2.4.7 Protocol operation 27

14 Packet-loss handling 28

14.1 Packet-loss handling mechanisms in WebRTC endpoints 28

14.1.1 Video 28

14.1.1.1 General 28

14.1.1.2 NACK messages 29

14.1.1.3 PLI message 29

14.1.1.4 SLI message 29

14.1.1.5 FIR message 29

14.1.1.6 Temporal-Spatial Trade-Off Request (TSTR) 29

14.1.1.7 Temporary Maximum Media Stream Bit Rate Request (TMMBR) 29

14.1.1.8 RTP retransmission 29

14.2 Packet-loss handling mechanisms supported in RTC endpoint 30

14.2.1 General 30

14.2.2 Video 30

14.2.2.1 General 30

14.2.2.2 NACK, PLI, SLI and FIR messages 30

14.2.2.3 TMMBR and TMMBN messages 31

14.2.2.4 RTP retransmission 31

15 RTC QoE metric reporting protocol 31

15.1 General 31

15.2 Quality of Experience metrics definition 31

15.2.1 Introduction 31

15.2.2 Corruption duration metric 32

15.2.3 Successive loss of RTP packets 32

15.2.4 Frame rate 33

15.2.5 Jitter duration 33

15.2.6 Sync loss duration 34

15.2.7 Round-trip time 34

15.2.8 Average codec bitrate 35

15.3 Quality metrics reporting protocol 35

15.3.1 General 35

15.3.2 Report format 35

15.3.3 Reporting protocol 37

Annex A (informative): RTC client in terminal 39

A.1 Overview of high-level RTC architecture 39

A.2 Reference RTC endpoint model 39

A.2.1 Audio 41

A.2.1.2 Microphone 41

A.2.1.2 Pre/post-processor 42

A.2.1.3 Codec 42

A.2.2 Video 42

A.2.2.1 Camera 42

A.2.2.2 Pre/post-processor 42

A.2.2.3 Codec 43

A.2.3 Sensor 43

A.2.3.1 General 43

A.2.3.2 Measure 43

Annex <X> (informative): Change history 44

# 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

[Editor’s note: Needs to be rephrased]

The immersive Real-Time Communication (iRTC) supports a set of features that enable a wide variety of immersive real-time media applications. For capturing media signals in more dimensions than 2D video or mono audio, outputs from multiple cameras and microphones, and the sensors are described. iRTC uses WebRTC with a modular protocol stack as transport, which is integrated into 5G systems, such that applications in need of QoS or other support can receive the necessary services from the network. 3GPP or other SDO’s specifications are referred when necessary.

# 1 Scope

The present document specifies the set of stage-3 procedures, APIs, and protocols for the reference points defined in Real-Time Media Communication (RTC) architecture. While TS 26.510 defines the common set of APIs and interactions, this document refers to TS 26.510 for the general aspects and primarily deals with RTC-specific aspects to support WebRTC-based real-time media transport over 5G.

# 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 26.506: "5G Real-time Media Communication Architecture (Stage 2)".

[3] 3GPP TS 26.510: "Media delivery; interactions and APIs for provisioning and media session handling".

[4] 3GPP TS 29.500: "5G System; Technical Realization of Service Based Architecture; Stage 3".

[5] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content".

[6] 3GPP TS 26.512: "5G Media Streaming (5GMS); Protocols".

[7] IETF RFC 8834 (2021): "Media Transport and Use of RTP in WebRTC".

[8] IETF RFC 8835 (2021): "Transports for WebRTC".

[9] 3GPP TS 23.003: "Numbering, addressing and identification".

[10] IETF RFC 8829 (2021): "JavaScript Session Establishment Protocol (JSEP)".

[11] IETF RFC 7807 (2016): "Problem Details for HTTP APIs".

[12] IETF RFC 8825 (2021): "Overview: Real-Time Protocols for Browser-Based Applications".

[13] IETF RFC 5124 (2008): "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF)".

[14] IETF RFC 7007 (2013): "Update to Remove DVI4 from the Recommended Codecs for the RTP Profile for Audio and Video Conferences with Minimal Control (RTP/AVP)".

[15] IETF RFC 3551 (2003): "RTP Profile for Audio and Video Conferences with Minimal Control".

[16] IETF RFC 4585 (2006): "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)".

[17] IETF RFC 3711 (2004): "The Secure Real-time Transport Protocol (SRTP)".

[18] IETF RFC 5104 (2008): "Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF)".

[19] IETF RFC 4588 (2006): "RTP Retransmission Payload Format".

[20] 3GPP TS 26.114: " IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction".

[21] IETF RFC 2616 (1999): "Hypertext Transfer Protocol -- HTTP/1.1".

[22] IETF RFC 7478 (2015): "Web Real-Time Communication Use Cases and Requirements".

[23] 3GPP TS 26.119: "Media Capabilities for Augmented Reality"

[24] 3GPP TS 38.331: "NR; Radio Resource Control (RRC); Protocol specification".

[25] Apple: "Getting Raw Accelerometer Events".

[26] Google: "Sensor Coordinate System".

[27] ITU-R Recommendation BT.601-7 (03/2011): "Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios".

[28] Microsoft: "Microphone Array Geometry Descriptor Format".

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

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

[31] W3C Recommendation: WebRTC: Real-Time Communication in Browsers, March 2023. <https://www.w3.org/TR/webrtc/>

[32] IETF RFC 7874 (2016): "WebRTC Audio Codec and Processing Requirements"

[33] IETF RFC 7742 (2016): "WebRTC Video Processing and Codec Requirements"

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

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

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

## 3.2 Symbols

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

<symbol> <Explanation>

## 3.3 Abbreviations

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

3DoF Three Degrees of Freedom

6DoF Six Degrees of Freedom

API Application Programming Interface

AR Augmented Reality

DRB Data Radio Bearer

DTLS Datagram Transport Layer Security

FFS For Further Study

FoV Field of View

HMD Head-Mounted Display

HTTP Hyper-Text Transfer Protocol

ICE Interactive Connectivity Establishment

IMU Inertial Measurement Unit

iRTC Immersive Real-Time Communication

LIDAR Light Detection and Ranging

MR Mixed Reality

MNO Mobile Network Operator

NAT Network Address Translation

OTT Over-The-Top

RGB Red-Green-Blue colour space

RGBD Red-Green-Blue-Depth

RTC Real-Time Communication

RTP Real-time Transport Protocol

SCTP Stream Control Transmission Protocol

SDO Standards Developing Organization

SLAM Simultaneous Localization And Mapping

SRTCP Secure Real-time Transport Control Protocol

SRTP Secure Real-time Transport Protocol

SSE Server-Sent Events

STUN Session Traversal Utilities for NAT

TLS Transport Layer Security

ToF Time of Flight

TURN Traversal Using Relays around NAT

WebRTC Web Real-Time Communication

XHR XMLHttpRequest

XR Extended Reality

# 4 Procedures for real-time media communication

## 4.1 General

This clause defines all procedures for real-time media communication using the different RTC reference points. Table 4.1-1 summarises the APIs used to provision and use RTC features specified in TS 26.506 [2].

Table 4.1‑1: Summary of APIs relevant to RTC features

|  |  |  |
| --- | --- | --- |
| RTCfeature | Abstract | Relevant APIs |
| Interface | API name | Clause |
| Content configuration | Content delivery is configured according to Configuration Provisioning associated with a Provisioning Session. | RTC-1 | Provisioning Sessions API | 6.2 |
| Configuration Provisioning API | 6.3 |
| RTC-5 | Configuration Information API | 10.3 |
| Service Access Information API | 10.2 |
| Metrics reporting | The RTC endpoint uploads metrics reports to the RTC AF according to a provisioned Metrics Reporting Configuration it obtains from the Service Access Information for its Provisioning Session. | RTC-1 | Provisioning Sessions API | 6.2 |
| Metrics Reporting Provisioning API | 6.7 |
| RTC-5 | Service Access Information API | 10.2 |
| Metrics Reporting API | 10.6 |
| Consumption reporting | The RTC endpoint provides feedback reports on currently consumed content according to a provisioned Consumption Reporting Configuration it obtains from the Service Access Information for its Provisioning Session. | RTC-1 | Provisioning Sessions API | 6.2 |
| Consumption Reporting Provisioning API | 6.4 |
| RTC-5 | Service Access Information API | 10.6 |
| Consumption Reporting API | 10.7 |
| Dynamic Policy invocation | The RTC endpoint activates different traffic treatment policies selected from a set of Policy Templates configured in its Provisioning Session. | RTC-1 | Provisioning Sessions API | 6.2 |
| Policy Templates Provisioning API | 6.6 |
| RTC-5 | Service Access Information API | 10.6 |
| Dynamic Policies API | 10.4 |
| Network Assistance | The RTC enpoint requests bit rate recommendations and delivery boosts from the RTC AF. | RTC-5 | Service Access Information API | 10.6 |
| Network Assistance API | 10.5 |
| Edge content processing | Edge resources are provisioned for processing content in RTC sessions. | RTC-1 | Provisioning Sessions API | 6.2 |
| Edge Resources Provisioning API | 6.5 |
| RTC-5 | Service Access Information API | 10.6 |

## 4.2 Procedures for media session handling

### 4.2.1 Provisioning (RTC-1) procedures

A RTC Application Provider may use the procedure in this clause to provision the network for WebRTC sessions that are operated by that RTC Application Provider. In order to configure ICE candidates, dynamic policies, and/or reporting, the RTC Application Provider shall create a new Provisioning session in the RTC AF and shall use the interactions specified in clause 5.2.2 of TS 26.510 [3] at reference point RTC-1 to create and subsequently manipulate Provisioning session in the RTC AF.

Throughout the Provisioning session established, reference point RTC-1 offers the following set of procedures:

- Discovery of ICE candidates: relays the configuration information for STUN, TURN, and SWAP servers in the trusted domain to RTC MSH in UE, at RTC-5, if required by the Provisioning session. The list of associated server information depends on the collaboration scenarios as identified in TS 26.506 [2].

- Configuration of dynamic policies: allows the configuration of Policy Templates at RTC-5 that can be applied to RTC-4m media sessions.

- Configuration of reporting: permits the MNO to collect, at RTC-5, QoE metrics and consumption reports about RTC-4m media sessions.

A RTC Application Provider may use any of these procedures, in any combination, to support its WebRTC sessions.

### 4.2.2 Network media session handling (RTC-3, RTC-5) procedures

The following operations at reference point RTC-5 are used by a RTC MSH in an UE to invoke services relating to WebRTC session on the RTC AF. Reference point RTC-3 may be involved to a subset of operations involved in the exchange of QoS flow information as well as QoE and consumption report.

- Service Access Information: It is the set of parameters and addresses needed by RTC endpoint to activate transmission and/or reception of WebRTC session. It additionally includes configuration information to invoke the subsequent procedures. The detailed procedure to acquire Service Access Information is specified in clause 5.3.2 of TS 26.510 [3].

- Configuration Information: It is the set of addresses needed by RTC endpoint to acquire the service URL. It may include the addresses of trusted STUN/TURN servers as well as trusted WebRTC signalling servers that supports the SWAP protocol. If it is activated by RTC Application Provider at reference point RTC-1, RTC MSH shall use the procedures and operations specified in clause 5.3.x of TS 26.510 [3].

- Dynamic policy invocation: It is used by RTC MSH to manage Dynamic Policy Instance resources in the RTC AF. RTC MSH shall use the interaction specified in clause 5.3.3 of TS 26.510 [3] to instantiate Policy Template in the RTC AF that are described in the Dynamic Policies API in clause 10.4.

- Metrics reporting: It is used to submit a QoE metrics report to the RTC AF by RTC MSH of RTC endpoint at reference point RTC-5 or by the RTC AS at reference point RTC-3, if metrics reporting is applied for a media streaming session. To determine whether and how to send metrics reports the RTC AF, the RTC MSH shall use the procedures and operations specified in clause 5.3.5 of TS 26.510 [3].

- Consumption reporting: It is used to submit a consumption report to the RTC AF by the RTC MSH of the RTC endpoint at reference point RTC-5 or by the RTC AS at reference point RTC-3, if consumption reporting is applied for WebRTC session. This is indicated by the presence of a Client Consumption Reporting Configuration in the Service Access Information. To determine whether and how to send consumption reports to the RTC AF, the RTC MSH shall use the procedures and operations specified in clause 5.3.6 of TS 26.510 [3].

Editor’s NOTE: Resources for consumption reporting is FFS.

- Network assistance: It is used by the RTC endpoint to request Network Assistance from one of the RTC AF instances listed in the Network Assistance Configuration of the Service Access Information. To do this, the RTC MSH shall use the procedures and operations specified in clause 5.3.4 of TS 26.510 [3].

### 4.2.3 UE media session handling (RTC-6, RTC-11) procedures

The reference point RTC-6 is used to exchange the report of media consumption as configured by Service Access Information. When consumption reporting is active for a particular WebRTC session, the RTC MSH shall use procedures and operations specified in clause 5.4.6 of TS 26.510 [3].

The reference point RTC-11 is used to exchange the QoE metric reporting as configured by Service Access Information. When metric reporting is active for a particular WebRTC session, the RTC MSH shall use procedures and operations specified in clause 15.

## 4.3 Procedures for media content and signalling transport

### 4.3.1 Media-centric transport (RTC-4) procedures

#### 4.3.1.1 General

Reference point RTC-4 interface may be further split into signalling part (RTC-4s) and media transport part (RTC-4m), depending on the collaboration scenario as specified in 3GPP TS 26.506 [2]. Table 4.3.1.1-1 describes the associated reference points for collaboration scenarios.

Table 4.3.1.1‑1: Associated reference point RTC-4s/4m for collaboration scenarios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reference point | Collaboration scenario 1 | Collaboration scenario 2 | Collaboration scenario 3 | Collaboration scenario 4 |
| RTC-4m | N/A | Required\* | Required | Required |
| RTC-4s | N/A | N/A | Required | Required |
| \* For the case when TURN server within ICE Function is involvedNOTE) N/A is meant that the corresponding reference point is not the scope of this specification |

#### 4.3.1.2 Signalling (RTC-4s) procedures

This reference point is used for the exchange of signalling messages related to the WebRTC session between two or more WebRTC endpoints. The RTC aware application (i.e., Native WebRTC app and Web app) send/receive signalling message to/from RTC AS (i.e., WebRTC Signalling function) RTC-4s. Signalling procedures for RTC-4s refer to the procedure specified in the signalling protocol for RTC in clause 13.2.

If trusted WebRTC signalling servers is provided, a RTC endpoint shall configure to one of the listed signalling servers (e.g., use Configuration Information provided at RTC-5). The configured signalling server information may be sent to WebRTC Framework at RTC-11. Using this information, Native WebRTC application and Web app communicate to the signalling server for media session set up (e.g., SDP negotiation) at RTC-4s.

#### 4.3.1.3 Media transport (RTC-4m) procedures

This reference point is used for transmission of media and other related data between two or more WebRTC endpoints. The WebRTC framework of the RTC endpoint send/receive the media data, application data and/or media related meta-data to/from RTC AS (e.g., trusted Media Function) or other RTC endpoint based on the input from the RTC aware application (e.g., Native WebRTC app and Web app).

[In the context of this specification for RTC endpoints, neither the requirements for RTC endpoints for audio codecs and processing as defined in IETF RFC 7874 [32] nor the requirements for RTC endpoints for video codecs and processing as defined in IETF RFC 7742 [33] apply.] For codecs support in RTC endpoints in the context of this specification, please refer to clause 16.

Editor’s note: The bracketed text is not agreed and needs to be revised for language.

Media transport at RTC-4m is established based on the collaboration scenario defined in TS 26.506 [2] and the signalling protocol applied for the media session establishment.

### 4.3.2 UE media delivery (RTC-7) procedures

This reference point RTC-7 is used to following purposes:

- To use WebRTC framework for media handling (e.g., gathering media capability information of the UE, controlling media transport). The functionalities provided on this interface are equivalent to WebRTC API defined in W3C [31].

# 5 General aspects of APIs

## 5.1 Usage of HTTP

### 5.1.1 HTTP protocol version

#### 5.1.1.1 RTC AF

Implementations of the RTC AF shall comply with clause 7.1.1 of TS 26.510 [3].

### 5.1.2 HTTP message bodies for API resources

The OpenAPI [23] specification of HTTP messages and their content bodies is contained in Annex A of TS 26.510 [3].

### 5.1.3 Usage of HTTP headers

#### 5.1.3.1 General

Standard HTTP headers shall be used in accordance with clause 5.2.2 of TS 29.500 [4] for all versions of HTTP.

#### 5.1.3.2 Media Session Handler identification

The Media Session Handler in the RTC Client shall identify itself to the RTC AF at interface RTC-5 using a User-Agent request header (see section 5.3.3 of RFC 7231 [5]) in which the first element shall be a product identified by the token RTCMediaSessionHandler and optionally suffixed with a product-version.

The Media Session Handler may additionally supply a comment element in the User-Agent request header containing a vendor-specific identification string.

#### 5.1.3.3 RTC AF identification

The RTC AF shall identify itself using a Server response header (see section 7.4.2 of RFC 7231 [5]) of the following form:

RTCAF-{FQDN}/{implementationSpecificSuffix}

where {FQDN} shall be the Fully-Qualified Domain Name of the RTC AF exposed to the requesting client, and {implementationSpecificSuffix} shall be determined by the implementation.

#### 5.1.3.4 Support for conditional HTTP GET requests

All responses from the RTC AF that carry a resource message body shall comply with clause 7.1.4.2 of TS 26.510 [3].

#### 5.1.3.5 Support for conditional HTTP POST, PUT, PATCH and DELETE requests

All API endpoints on the RTC AF that expose the HTTP POST, PUT, PATCH or DELETE methods shall comply with clause 7.1.4.3 of TS 26.510 [3].

# 6 Provisioning interface (RTC-1)

## 6.1 General

This clause defines provisioning API used by the Application Provider to provision resources for their real-time communication sessions. The Provisioning API is an extension of the Provisioning API as defined in TS 26.510 clause 8 [3].

Table 6.1-1 specifies the relevant APIs for RTC sessions in comparison with those in TS 26.510 [3]:

The relationship is categorized as follows:

**Common**:
The API is supported on RTC-1 reference point. The procedure, resource structure and data models for this API comply with the corresponding M1 API specified in 3GPP TS 26.510 [xxx] is implemented on RTC-1.

**Extended**:
The API is supported on RTC-1 reference point. The procedure, resource structure and/or data models for this API has extension to the corresponding M1 API specified in 3GPP TS 26.510 [3]. The extensions for RTC-1 API are specified in this specification.

**Not Applicable**:
The API is not supported on RTC-1 reference point.

Table 6.1‑1: List of APIs relevant to RTC-1

|  |  |  |  |
| --- | --- | --- | --- |
| API | Common | Extended | Not Applicable |
| Provisioning Sessions API |  | O |  |
| Server Certificates Provisioning API |  |  | O |
| Content Preparation Templates API |  |  | O |
| Content Protocols Discovery API |  |  | O |
| Content Hosting Provisioning API |  |  | O |
| Consumption Reporting Provisioning API | O |  |  |
| Metrics Reporting Provisioning API | O |  |  |
| Policy Templates Provisioning API |  | O |  |
| Edge Resources Provisioning API | O |  |  |
| Event Data Processing Provisioning API |  |  | O |
| Configuration Provisioning API | O |  |  |

## Editor's note: Configuration Provisioning API needs to be defined as RTC specific API in 3GPP TS 26.510.6.2 Provisioning Sessions API

The Provisioning Sessions API is used by RTC Application Provider to instantiate and manipulate Provisioning Sessions in the RTC System. The resource structure and the data model are specified in clause 8.3 of TS 26.510 [3]. When Provisioning Session API is used in RTC, the provisionedConfigurationIds object shall be present.

[Editor’s Note: The following table should be included in clause 8.3 of TS 26.510;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| provisionedConfigurationIds | Array(ResourceId) | 0..1 | C: -R: RO | A list of the provisioned configuration identifiers that are currently associated with this Provisioning Session. | rtc |

]

## 6.3 Configuration Provisioning API

The Configuration Provisioning API is used by the Application Provider to provision configuration that will be relayed to the RTC MSH for usage with RTC sessions of that Application Provider. The resource structure and the data model are specified in clause 8.xx of TS 26.510 [3].

Editor’s Note: The data model for this API, as provided in S4-231711 should be included in clause 8.xx of TS 26.510

## 6.4 Consumption Reporting Provisioning API

The Consumption Reporting Provisioning API is a RESTful API that allows the RTC Application Provider to configure the Consumption Reporting Procedure for a particular RTC Provisioning Session at interface RTC-1. The resource structure and the data model are specified in clause 8.11 of TS 26.510 [3].

## 6.5 Edge Resources Provisioning API

The Edge Resources Provisioning API is used by the RTC Application Provider to provision edge resource usage for RTC sessions associated with the parent Provisioning Session. The information serves as a template to select or instantiate the appropriate EAS instance that will serve the media session to the UE. The resource structure and the data model are specified in clause 8.6 of TS 26.510 [3].

## 6.6 Policy Templates Provisioning API

The Policy Templates Provisioning API allow a RTC Application Provider to configure a set of Policy Templates within the scope of a Provisioning Session that can subsequently be applied to RTC sessions belonging to that Application Provider using the Dynamic Policies API specified in clause 8.7 of TS 26.510 [3].

Editor’s Note: The extended features for RTC should be added in clause 8.7 of TS 26.510, including RTCQoSSpecification object proposed. Note that RTCQoSSpecification should re-named and revised for common usage.

## 6.7 Metrics Reporting Provisioning API

The Metrics Reporting Provisioning API allows a RTC Application Provider to configure the Metrics Collection and Reporting procedure for a particular RTC session at reference point RTC-1. The metric reporting scheme is signalled using in the **Scheme** element in the MetricsReportingConfiguration. The URN to be used for the **Scheme**@schemeIdUri shall be "urn:3GPP:ns:PSS:RTC:QM1".

The semantics and XML syntax of the scheme information for the RTC quality reporting scheme are specified in Table 6.7-1 and Table 6.7-2, respectively.

Editor’s Note:. The modified data model for RTC should be included in clause 8.10 of TS 26.510

Table 6.7-1: Semantics of Quality Reporting Scheme Information

|  |  |  |
| --- | --- | --- |
| Element or Attribute Name | Use | Description |
|  | @apn | O | This attribute gives the access point that should be used for sending the QoE reports. |
|  | @format | O | This field gives the requested format for the reports. Possible formats are: "uncompressed" and "gzip". |
|  | @samplepercentage | O | Percentage of the clients that should report QoE. The client uses a random number generator with the given percentage to find out if the client should report or not. |
|  | @reportingserver | M | The reporting server URL to which the reports will be sent. |
|  | @reportinginterval | O | Indicates the time(s) reports should be sent. If not present, then the client should send a report after the streaming session has ended. If present, @reportingInterval=n indicates that the client should send a report every n-th second provided that new metrics information has become available since the previous report. For each report sent, only the newly collected information since the previous report shall be reported. |
|  | @measureinterval | O | Indicates the time over which each metrics value is calculated. This field splits the session duration into a number of equally sized periods where each period is of the length specified by measureinterval field. If the "measureinterval" field is not present, the metrics resolution shall cover the period specified by the "measurerange" field. If the "measurerange" field is not present the metrics measure interval shall be for the whole session duration. |
|  | @measurerange | O | Indicates the time range in the stream for which the QoE metrics will be reported. There shall be only one range per measurement specification. If the "measurerange" field is not present, the metrics range shall be the whole session duration. |
|  | @syncthreshold | O | Indicates the maximum allowed sync loss duration between the playback time of the last played frame of the video stream and the playback time of the last played frame of the speech/audio stream. This parameter is set to control the maximum amount of allowed sync mismatch. This parameter is specified in ms. When the parameter has not been set, it defaults to 100 ms. |
|  | @jitterthreshold | O | Indicates the maximum allowed jitter duration between the actual playback time and the expected playback time. This parameter is set to control the amount of allowed jitter. This parameter is specified in ms. When the parameter has not been set, it defaults to 100 ms. |
|  | **LocationFilter** | 0..1 | When present, this element indicates the geographic area(s) or location(s) where quality metric collection is requested. When not present, quality metric collection is requested regardless of the device’s location. The **LocationFilter** element comprises one or more instances of any combination of targeted cell-IDs, polygons and circular areas. Each cell-ID entry in **LocationFilter** is announced in cellList, and each polygon and circular area entry is announced in the polygonList or and circularAreaList elements, respectively. |
|  |  Cellist | 0..N | This element specifies a list of cell identified by E-UTRAN-CGI or CGI. |
|  |  Shape |  | Geographic area comprising one or more instances of polygonList and/or circularAreaList elements. |
|  |  polygonList | 0..N | This element, when present, comprises a list of ‘Polygon’ shapes as defined by OMA MLP. |
|  |  @confLevel | O | This attribute indicates the probability in percent that the 5G-RTC client is located in the corresponding polygon area. It is defined as ‘lev\_conf’ by OMA MLP. If not present, it has default value of 60. |
|  |  circularAreaList | 0..N | This element, when present, comprises a list of ‘CircularArea’ shapes as defined by OMA MLP. |
|  |  @confLevel | O | This attribute indicates the probability in percent that the 5G-RTC client is located in the corresponding circular area. It is defined as ‘lev\_conf’ by OMA MLP. If not present, it has default value of 60. |
|  | **SliceScope** | 0..1 | When present, this element indicates a list of network slices in which the QoE collection is requested. When not present, quality metric collection is requested for all network slices. The SliceScope is a list of S-NSSAIs. |
| Legend:For attributes: M=Mandatory, O=Optional, OD=Optional with Default Value, CM=Conditionally Mandatory.For elements: <minOccurs>…<maxOccurs> (N=unbounded)Elements are bold; attributes are non-bold and preceded with an @ |

Table 6.7-2: Syntax of Quality Reporting Scheme Information

|  |
| --- |
| <?xml version="1.0"?><xs:schema targetNamespace="urn:3GPP:ns:PSS:RTC:2023:qm1"  attributeFormDefault="unqualified"  elementFormDefault="qualified"  xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="urn:3GPP:ns:PSS:RTC:2023:qm1">  <xs:annotation> <xs:appinfo>5G RTC Quality Reporting</xs:appinfo> <xs:documentation xml:lang="en"> This Schema defines the quality reporting scheme information for 5G RTC. </xs:documentation> </xs:annotation>   <xs:element name="ThreeGPQualityReporting" type="SimpleQualityReportingType"/>  <xs:complexType name="SimpleQualityReportingType"> <xs:sequence> <xs:element name="LocationFilter" type="LocationFilterType" minOccurs="0"/> <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:attribute name="apn" type="xs:string" use="optional"/> <xs:attribute name="format" type="FormatType" use="optional"/> <xs:attribute name="samplepercentage" type="xs:double" use="optional"/> <xs:attribute name="reportingserver" type="xs:anyURI" use="required"/> <xs:attribute name="reportinginterval" type="xs:unsignedInt" use="optional"/> <xs:attribute name="measureinterval" type="xs:unsignedInt" use="optional"/> <xs:attribute name="measurerange" type="xs:unsignedInt" use="optional"/> <xs:attribute name="syncthreshold" type="xs:unsignedInt" use="optional"/> <xs:attribute name="jitterthreshold" type="xs:unsignedInt" use="optional"/> <xs:attribute name="sliceScope" type="UnsignedIntVectorType" use="optional"/> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:complexType>  <xs:simpleType name="FormatType">  <xs:restriction base="xs:string"> <xs:enumeration value="uncompressed" /> <xs:enumeration value="gzip" /> </xs:restriction> </xs:simpleType> <xs:complexType name="LocationFilterType"> <xs:sequence> <xs:element name="cellID" type="xs:unsignedLong" minOccurs="0" maxOccurs="unbounded"/> <xs:element name="shape" type="ShapeType" minOccurs="0"/> <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:complexType> <xs:complexType name="ShapeType"> <xs:sequence> <xs:element name="PolygonList" type="PolygonListType" minOccurs="0"/> <xs:element name="CircularAreaList" type="CircularAreaListType" minOccurs="0"/> <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:complexType> <xs:complexType name="PolygonListType"> <xs:annotation> <xs:documentation> see [OMA MLP] </xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="Polygon" minOccurs="0" maxOccurs="unbounded"/> <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:attribute name="ConfLevel" type="xs:unsignedInt" use="optional"/> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:complexType> <xs:complexType name="CircularAreaListType"> <xs:annotation> <xs:documentation> see [OMA MLP] </xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="CircularArea" minOccurs="0" maxOccurs="unbounded"/> <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:attribute name="ConfLevel" type="xs:unsignedInt" use="optional"/> <xs:anyAttribute namespace="##other" processContents="lax"/> </xs:complexType> <xs:simpleType name="UnsignedIntVectorType"> <xs:list itemType="xs:unsignedInt"/> </xs:simpleType></xs:schema> |

# 7 Media hosting interface (RTC-2)

Interfaces of this reference point are not specified in this release.

NOTE: The usage of content hosting at reference point RTC-2 is FFS.

# 8 RTC AS to RTC AF APIs (RTC-3)

APIs for the reference point M3 in the generalized media delivery architecture, as specified TS 26.510 [3] may be used for metric reporting and consumption reporting.

# 9 Media-centric transport interface (RTC-4)

## 9.1 General

This clause deals with the interface to transport media over WebRTC session and signalling information at reference point RTC-4. TS 26.506 [2] specifies various collaboration scenario depending on the usable network entities in the trusted domain, leading to the different interactions and operations at RTC-4.

- Collaboration scenario 1: WebRTC session is completely managed over the top and no APIs at RTC-4 is specified.

- Collaboration scenario 2: ICE function is present in the trusted DN and only media transport is specified in clause 9.2 when TURN is involved for WebRTC session.

- Collaboration scenario 3 and 4: In addition to collaboration scenario 2, trusted signalling server and trusted media function is available. WebRTC framework communicates with RTC AS for both media transport and signalling exchange, as specified in clause 9.2 and 9.3 respectively.

## 9.2 Media transport (RTC-4m)

WebRTC framework in RTC endpoint may transport media data and/or other related data to RTC AS at reference point RTC-4m. For the supported media capabilities WebRTC endpoints, please refer to clause 16.

For the case of media data, RTC endpoint transmits any combination of video, audio, and speech using RTP for WebRTC (RFC 8834 [7]).

If RTC endpoint transports those media types, then it shall support the extended secure RTP profile for RTCP-based feedback (RTP/SAVPF) (RFC 5124 [13]), as extended by RFC 7007 [14]. Encoded media stream shall be encapsulated into the secure RTP packet as specified in RFC 3711 [17].

For the case of other related data such as application data or metadata, RTC endpoint shall use WebRTC Data Channel [29] and therefore support the encapsulation of SCTP over DTLS as defined in [30].

## 9.3 Signalling exchange (RTC-4s)

Signalling exchange refers to a series of interactions to exchange the configuration information between two RTC endpoints (e.g., between applications (Native WebRTC Application/Web App) via WSF) to create and manage RTCPeerConnection. It includes the available transport protocol, NAT traversal route, network addresses as well as the codecs and media types in common between two RTC endpoints or between the RTC endpoint and the trusted media function.

This signalling information is exchanged based on the full-duplex reliable WebSocket connection, as specified in clause 13.2.

NOTE: TS 26.119 [23] defines the device type and media capabilities identifiers specifically for UEs with immersive media capabilities. The use of these identifiers during the signalling exchange is FFS.

# 10 Control transport interface (RTC-5)

## 10.1 General

This clause defines Control Transport API used by the RTC Media Session Handler to access resources exposed by the RTC AF at interface RTC-5. The Control Transport API is a profile of the Network Media Session Handling API defined in TS 26.510 clause 9.

Table 10.1-1 specifies the relevant APIs for RTC sessions in comparison with those in TS 26.512 [6]:

Table 10.1‑1: List of APIs relevant to RTC-5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| API | Inherited | Extended/Modified | Not Relevant | New |
| Service Access Information API |  | O |  |  |
| Configuration Information API |  |  |  | O |
| Dynamic Policies API |  | O |  |  |
| Network Assistance API | O |  |  |  |
| Metrics Reporting API |  | O |  |  |
| Consumption Reporting API | O |  |  |  |

Editor’s Note: Service Access Information in 26.510 may be renamed, as access information has already exposed by Configuration Information

## 10.2 Service Access Information API

The Service Access Information API is used by the RTC Media Session Handler to acquire configuration information from the RTC AF that enables it to use the other Control Transport APIs in clause 103 *et seq*. The resource structure and the data model are specified in clause 9.2 of TS 26.510 [3].

When the Service Access Information API is used in RTC, streamingAccess object in ServiceAccessInformation resource shall not be present.

## 10.3 Configuration Information API

The Configuration Information API is used by the RTC Media Session Handler to acquire the configuration information such as ICE candidates from the RTC AF. It is specified to relay the identical ProvisionedConfiguration resource from the RTC AF using the Configuration procedure, if requested by the Provisioning information. The resource structure and the data model are specified in clause 9.x of TS 26.510 [3].

Editor’s Note: Context of this configuration information API should be included in clause 9.x of TS 26.510.

## 10.4 Dynamic Policies API

The Dynamic Policy API allows both the MSH and the trusted ICE or WebRTC Signalling Function AS to request a specific QoS and charging policy to be applied to the data flows of an RTC session. The resource structure and the data model are specified in clause 9.3 of TS 26.510 [3].

[When the Dynamic Policy is used for QoS Flow management, the qoSSpecification object shall be present and its type shall be set to RTCQoSSpecification, as specified in Table xx of TS 26.510.]

## 10.5 Network Assistance API

If AF-based Network Assistance is supported, then the Network Assistance API component of interface RTC-5 is first used to provision a Network Assistance Session resource. The Network Assistance Resource can then be used to obtain bit rate recommendations and to issue delivery boost requests during the ongoing RTC session.

The Network Assistance API is defined in clause 9.4 of TS 26.510 [3]. [When it is used, the qoSSpecification object shall be present and its type shall be set to RTCQoSSpecification, as specified in Table xx of TS 26.510.]

## 10.6 Metrics Reporting API

The Metric Reporting API allows the RTC Media Session Handler to report QoE metrics to the RTC AF, as configured by the SerciveAccessInformation resource in clause 10.2. For RTC, clause 15.3.1 and clause 15.3.2 specify the required MIME content type and metrics report format for the 3GPP urn:‌3GPP:‌ns:‌PSS:‌RTC:‌QM1 metrics reporting scheme.

NOTE: When the trusted WebRTC signalling function is present in RTC session, the metric reporting may be reported to the signalling function in RTC AS.

## 10.7 Consumption Reporting API

The Metric Reporting API allows the RTC Media Session Handler to report media consumption to the RTC AF, as configured by the SerciveAccessInformation resource in clause 10.2. The report procedure and report format are defined in clause 9.5 of TS 26.510 [3].

# 11 Media session handling client API (RTC-6, RTC-11)

Reference point RTC-6 is used to prepare consumption reporting parameters to be reported to RTC AF at reference point RTC-5. If consumption reporting for WebRTC session is configured, the RTC MSH shall regularly determine the consumption reporting parameters defined in clause 10.3.6 of TS 26.510 [3] shall report these values.

Reference point RTC-11 is used to collect QoE metrics in RTC MSH, which are supposed to be reported to RTC AF, if requested in the Provisioning Session. RTC endpoint supporting metric reporting shall report QoE metrics defined in clause 15.2. While metric reporting procedure is defined in clause 9.5 of TS 26.510 [3], the reporting protocol and the format are specified in clause 15.3 of this specification.

# 12 Client interface (RTC-7)

Reference point RTC-7 is used to communicate between Native WebRTC application and WebRTC framework for establishment and management of RTCPeerConnection, which is equivalent to WebRTC APIs specified by W3C [31].

# 13 Protocols of real-time media communication

## 13.1 General

The RTC endpoint supports transport protocols used in WebRTC, as specified in RFC 8834 [7], including the protocols for interaction with intermediate boxes such as firewalls, relays, and NAT boxes [8]. Figure 13.1-1 shows the protocol stack of RTC endpoint.



Figure 13.1-1: Protocol stack for a basic RTC endpoint

## 13.2 WebRTC signalling protocol

### 13.2.1 General

The Simple WebRTC Application Protocol (SWAP) supports collaboration scenario 3 described in [2].

NOTE: The signalling protocol which supports collaboration scenario 4 (and applicable to collaboration scenario 3) is specified as a different protocol in future release.

### 13.2.2 Protocol and version identification

The WebRTC signalling protocol and the version of the protocol shall be determined per WebSocket connection. The WebRTC signalling protocol and the version of the protocol shall be identified by the WebSocket URI for the HTTP upgrade request for WebSocket connection establishment (i.e., the Request-URI of the HTTP request). The WebSocket URI for the HTTP upgrade request shall be consistent with the WebSocket URI structure specified in clause 13.2.3.

The use of "Sec-WebSocket-Protocol" header field is dependent on the WebRTC signalling protocol and the version of the protocol.

### 13.2.3 WebSocket URI structure

WebSocket URI of WebSocket connection for WebRTC signalling protocol message shall be:

{protocolRoot}/<protocolName>/<protocolVersion>

"protocolRoot" shall be a concatenation of the following parts:

- scheme ("wss")

- the fixed string "://"

- authority (host and optional port) as defined in IETF RFC 3986. The host should be represented by the service provider (operator or OTT) specific FQDN (for FQDN examples see clause 28.3.2 in [9].

- an optional deployment-specific string (e.g., server prefix) that starts with a "/" character.

"protocolName" shall be protocol-specific string which indicates the name of the WebRTC signalling protocol.

"protocolVersion" shall indicate the version of the WebRTC signalling protocol. The protocol version shall be indicated as the concatenation of the letter "v" and the WebRTC signalling protocol version number. The other fields shall not be included in the URI.

For example, 'v1'.

NOTE: The "protocolVersion" will only be increased if the new protocol version contains not backward compatible changes.

A URI should not contain a trailing slash, and if it contains one, then it should be ignored/removed.

### 13.2.4 SWAP

#### 13.2.4.1 Protocol and version identification

The SWAP version shall be included in the WebSocket URI path as “/3gpp-swap/v1/".

The present version of SWAP, the Sec-WebSocket-Protocol header field with "3gpp.SWAP.v1" subprotocol identifier shall be included in the HTTP upgrade request.

#### 13.2.4.2 Transport

SWAP protocol shall operate over a full-duplex reliable WebSocket connection between the two endpoints or between an endpoint and a SWAP server. The following figures depict both scenarios.



Figure 13.2.4.2-1: Point-to-point SWAP



Figure 13.2.4.2-2: SWAP relay

In the former, one of the endpoints shall act as the WebSocket server and listen for the incoming connection request. The endpoint is not required to support more than one client connection at any point of time.

When a SWAP server is used, sufficient information shall be provided to facilitate the relaying of the messages from the server to the other endpoint.

#### 13.2.4.3 State machine

The SWAP server maintains state information about ongoing WebRTC sessions. The following state machine reflects the state tracked by the SWAP server.



Figure 13.2.4.3-1: SWAP state machine

The SWAP protocol is designed to adhere to the JSON Session Establishment Protocol (JSEP) state machine as defined in [10]. The JSEP state machine is reproduced in the following figure.



Figure 13.2.4.3-2: JSEP state machine

SWAP currently does not support preliminary answers in its version 1. Any preliminary answers that are generated by the application will not be sent by the SWAP endpoint.

SWAP version 1 does not support ICE trickling. The final list of ICE candidates is expected to be part of the initial offer message. The application shall wait for the ICE gathering phase to finish prior to sending the offer to the remote endpoint.

#### 13.2.4.4 Message syntax and semantics

##### 13.2.4.4.1 Common message fields

###### 13.2.4.4.1.1 Source (source)

Each message shall carry a unique source identifier that identifies the message source. The source identifier shall be a randomly generated string. The source identifier shall not be changed during the lifetime of a session.

A SWAP server that detects a change in the source identifier from an endpoint over the same WebSocket connection shall ignore the corresponding message. The source identifier shall at least have 10 UTF-8 characters.

###### 13.2.4.4.1.2 Message Identifier (messge\_id)

The message identifier shall be a sequence number for the message. The message identifier is scoped by the source identifier, i.e., it shall be uniquely assigned by the source of the message.

The message identifier shall be a positive monotonically increasing number.

###### 13.2.4.4.1.3 Message Type (message\_type)

The message type identifies the type of the SWAP message. The supported message types in version 1 of the specification are:

- Register

- Response

- Connect

- Accept

- Reject

- Update

- Close

- Application.

The message type shall be considered as a case-insensitive string.

##### 13.2.4.4.2 Register message

###### 13.2.4.4.2.1 Description

An endpoint registers with the SWAP server and provides the matching criteria that may be used to match this endpoint with incoming connection requests.

The register message is not required for the case of a direct connection between the two endpoints.

###### 13.2.4.4.2.2 Parameters

matching\_criteria: an object that provides the matching criteria for relaying incoming SWAP messages to their destination. The matching criteria object consists of a type and a value.

The supported types in this version of the specification are the following:

- ipv4: The IPv4 address of the target endpoint

- ipv6: The IPv6 address of the target endpoint

- fqdn: The FQDN of the target endpoint

- service: An identifier of a service or an application

- user: An identifier of the user such as a SIP address, a GPSI, or an MSISDN

- eas: An EAS identifier

- app: application-specific matching criteria that is compared using binary or string comparison

- location: one or more identifiers of a geographic location or area

- qos: a description of the QoS that is supported by the connection to the endpoint

- processing: a profile description of the processing capabilities of the endpoint.

The matching criteria may be combined together to further restrict the selection of the target endpoint. If multiple endpoints match all provided criteria, then the SWAP server shall randomly select one of the target endpoints.

An endpoint that registers without providing certain matching criteria, such as qos or processing, shall be deprioritized during the selection process, where the request contains these matching criteria.

An endpoint that supports multiple media capabilities, the processing type in matching criteria should be represented as a pair of media decoding (as input of split rendering process) and encoding (as output of split rendering process).

##### 13.2.4.4.3 Response message

###### 13.2.4.4.3.1 Description

A SWAP server shall respond to every received request with a response message. The response message shall indicate whether the message is acknowledged or erroneous.

If a message is relayed properly to an endpoint, an acknowledgement message shall be sent to the source endpoint.

If an error is detected or a target endpoint cannot be identified, the SWAP server shall respond with an error response to the source endpoint.

In addition to the common fields, the response message shall include the request message id. In case of an error response, the message shall contain a textual description of the error.

###### 13.2.4.4.3.2 Parameters

type: the type parameter may either be “ack” or “error”.

target: the identifier of the target of this message, which originated the request message corresponding to this response.

request: the message identifier of the request message that corresponds to this response.

description: in case of an error response, this field provides a description of the error message. In case of an acknowledgement, this description field is optional.

##### 13.2.4.4.4 Connect message

###### 13.2.4.4.4.1 Description

The connect message is used by the source to establish a connection with the endpoint. The request shall include the SDP offer. If connecting via a SWAP server, the request shall include the matching\_criteria parameter to identify the target endpoint.

###### 13.2.4.4.4.2 Parameters

offer: a string that includes the SDP description for the offer.

matching\_criteria: an array that contains the matching criteria for the target endpoint. Each object shall comply with the definition of a matching criteria as described in clause 13.2.4.4.2.

##### 13.2.4.4.5 Accept message

###### 13.2.4.4.5.1 Description

If the connection request is accepted by the remote endpoint, it shall reply with an accept message. The accept message shall contain the answer SDP.

###### 13.2.4.4.5.2 Parameters

target: This parameter indicates the id of the target endpoint.

answer: This parameter shall contain the answer SDP.

##### 13.2.4.4.6 Update message

###### 13.2.4.4.6.1 Description

The update message may be sent by any of the endpoints of a WebRTC session. It contains the updated SDP, which may add, update, or remove one or more local media streams. If accepted, the remote endpoint shall reply with an accept message.

###### 13.2.4.4.6.2 Parameters

target: This parameter indicates the id of the target endpoint.

sdp: The updated local SDP that is transmitted to the remote endpoint.

##### 13.2.4.4.7 Reject message

###### 13.2.4.4.7.1 Description

In case the remote endpoint does not accept the offer or update message, it shall respond with the reject message. The message shall contain a reference to the corresponding offer or update message as well as a description of the reason why the message was rejected.

###### 13.2.4.4.7.2 Parameters

target: this parameter indicates the id of the target endpoint

request: the message identifier of the request

error\_id: an identifier of the error message

description: a description of the error message.

##### 13.2.4.4.8 Close message

###### 13.2.4.4.8.1 Description

The close message may be triggered by any of the two endpoints of a WebRTC session. Upon reception, the endpoint shall respond with an accept message, after which the WebRTC session is torn down and the resources associated with the WebRTC session are released.

###### 13.2.4.4.8.2 Parameters

target: this parameter indicates the id of the target endpoint

##### 13.2.4.4.9 Application message

###### 13.2.4.4.9.1 Description

Application-specific message may be defined by the application and exchanged between the endpoints of a WebRTC session. The message shall contain a type that uniquely identifies the type of the application message. If an application message type is not supported, it shall be rejected by the remote endpoint.

###### 13.2.4.4.9.2 Parameters

target: this parameter indicates the id of the target endpoint

type: the type of the application message shall be a URN that uniquely identifies the application message type.

value: an object that contains the application message content.

#### 13.2.4.5 Integrity and security

Integrity and confidentiality protection are supported through the protection of the message information as follows:

- A key derivation mechanism is configured by the application provider to the session participants, e.g., using a shared secret algorithm

- For integrity protection, the derived key is used to provide integrity protection, e.g., using a Message Authentication Code (MAC) for message payload

- For encryption, the derived key is used to encrypt the message payload. The encrypted data may then be encoded using base64 to enable embedding it in JSON.

These mechanisms are possible to implement using the WebCrypto API, which makes them web-friendly. Consulting with SA3 on these security algorithms is recommended.

#### 13.2.4.6 JSON schema

The JSON schema of the SWAP messages is follows:

Table 13.2.4.6-1: JSON schema of SWAP message

|  |  |
| --- | --- |
|

|  |
| --- |
| {    "$schema": "http://json-schema.org/draft-07/schema",    "title": "3GPP.SWAP",    "type": "object",    "description": "The description of the SWAP messages",    "properties": {        "version": {            "description": "the version of the SWAP protocol",            "type": "integer"        },        "source\_id": {            "description": "A unique identifier of the source",             "type": "string"        },        "message\_id": {            "description": "the sequence number of the message ",            "type": "integer"        },        "message\_type": {            "description": "the type of the SWAP message",            "type": "string",            "enum": ["register", "connect", "response", "accept", "reject", "update", "close", "application"]        },        "oneOf": [            {                "type": "object",                "properties": {                    "matching\_criteria": {"type": "string", "enum": ["ipv4", "ipv6", "fqdn", "service", "user", "eas", "app", "location", "qos", "processing"]}                }            },            {                "type": "object",                "properties": {                    "type": {"type": "string", "enum": ["ack", "error"]},                    "source": {"type": "string"},                    "request": {"type": "integer"},                    "description": {"type": "string"}                }            },            {                "type": "object",                "properties": {                    "offer": {"type": "string"},                    "matching\_criteria": {"type": "string", "enum": ["ipv4", "ipv6", "fqdn", "service", "user", "eas", "app", "location", "qos", "processing"]}                }            },            {                "type": "object",                "properties": {                    "answer": {"type": "string"}                }            },            {                "type": "object",                "properties": {                    "source": {"type": "string"},                    "request": {"type": "number"},                    "error\_id": {"type": "string"},                    "description": {"type": "string"}                }            },            {                "type": "object",                "properties": {                    "type": {"type": "string"},                    "value": {"type": "object"}                }            }        ],        "extensions": {}    },    "required": ["version", "source", "message\_id"]} |

 |

#### 13.2.4.7 Protocol operation

SWAP is an acknowledged signalling protocol for WebRTC. Each message that the WebRTC signalling server receives shall be acknowledged after proper processing. This is valid for the case where one of the endpoints acts as the signalling server. The Response message may also indicate an error, in case the received message can not be processed and forwarded properly.

The error messages shall be formatted according to the Problem Details specification in RFC 7807 [11]. The following error message types are defined in this specification:

Table 13.2.4.6-1: Error message types

|  |  |
| --- | --- |
| **Error message type** | **Error message title** |
| http://forge.3gpp.org/sa4/swap/message\_unknown.html | Message type unknown |
| http://forge.3gpp.org/sa4/swap/message\_malformatted.html | Message malformatted |
| http://forge.3gpp.org/sa4/swap/target\_unknown.html | Target cannot be located |
| http://forge.3gpp.org/sa4/swap/unauthorized.html | Unauthorized |

The WebRTC Signalling Function uses the (source, target) identifier pairs of the communicating endpoints to identify the session and properly route the messages. Note that in the first connect message, the target identifier might not be known; in which case, the routing is done based on the matching criteria.

The source identifier shall be a string that uniquely identifies the source. An example of such identifier may be a randomly generated UUID.

Every message shall contain the common message fields: source, message\_id, and message\_type. The source field shall always indicate the originator of the current message. A WebRTC signalling server shall also generate and use a unique identifier.

# 14 Packet-loss handling

## 14.1 Packet-loss handling mechanisms in WebRTC endpoints

### 14.1.1 Video

#### 14.1.1.1 General

The following packet loss handling mechanisms are recommended in RFC 8834 [7] and RFC 8835 [8] for a WebRTC endpoint defined in RFC 8825 [12].

WebRTC endpoints offering video shall support extended secure RTP profile for RTCP-based feedback (RTP/SAVPF) (RFC 5124 [13]), as extended by RFC 7007 [14]. The RTP/SAVPF profile is the combination of the basic RTP/AVP profile in RFC 3551 [15], the RTP profile for RTCP-based feedback (RTP/AVPF) in RFC 4585 [16], and the secure RTP profile (RTP/SAVP) in RFC 3711 [17].

The WebRTC endpoints behaviour can be controlled by allocating enough RTCP bandwidth using "b=RR:" and "b=RS:" and setting the value of "trr-int". The attributes "b=RS:<bw>" and "b=RR:<bw>" as defined in RFC 4585 [16] may be used to assign a different bandwidth (measured in bits per second) for RTCP messages to RTP senders and receivers, respectively. The attribute "trr-int" in SDP is used to specify the minimum time interval between two Regular (full compound) RTCP packets in milliseconds for a media session.

WebRTC endpoints are recommended to use the following mechanisms to recover from packet losses:

- AVPF Generic NACK

- Picture Loss Indication (PLI) feedback message

- Slice Loss Indication (SLI) feedback message

- Full Intra Request (FIR) feedback message

- Temporal-Spatial Trade-Off Request (TSTR)

- Temporary Maximum Media Stream Bit Rate Request (TMMBR)

- RTP Retransmission

These mechanisms offer different performance trade-offs according to channel conditions such as end-to-end delay, bandwidth, rate and packet loss profile.

#### 14.1.1.2 NACK messages

AVPF NACK messages are used by WebRTC endpoints to indicate non-received RTP packets for video. WebRTC receivers may send NACKs for missing RTP packets. RTP packet stream senders are required to understand the generic NACK message defined in RFC 4585 [16], but they can choose to ignore some or all of this feedback.

#### 14.1.1.3 PLI message

The Picture Loss Indication message is used by a receiver to tell the sending encoder that it lost the decoder context and would like to have it repaired. WebRTC endpoints that are sending media shall understand and react to PLI feedback messages as a loss-tolerance mechanism. Receivers can send PLI messages.

#### 14.1.1.4 SLI message

The Slice Loss Indication message as defined in RFC 4585 [16] is used by a WebRTC receiver to tell the encoder that it has detected the loss or corruption of one or more consecutive macro blocks and would like to have these repaired somehow. It should be that receivers generate SLI feedback messages if slices are lost when using a codec that supports the concept of macro blocks. A sender that receives an SLI feedback message should attempt to repair the lost slice(s).

#### 14.1.1.5 FIR message

The Full Intra Request message defined in RFC 5104 [18] is used to make a request by a WebRTC receiver for a new Intra picture from a WebRTC sender. WebRTC endpoints that are sending media shall understand and react to FIR feedback messages they receive. Support for sending FIR messages is optional.

#### 14.1.1.6 Temporal-Spatial Trade-Off Request (TSTR)

The temporal-spatial trade-off request and notification are defined in RFC 5104 [18]. This request can be used to ask the video encoder to change the trade-off it makes between temporal and spatial resolution -- for example, to prefer high spatial image quality but low frame rate. Support for TSTR requests and notifications in WebRTC endpoints is optional.

#### 14.1.1.7 Temporary Maximum Media Stream Bit Rate Request (TMMBR)

The Temporary Maximum Media Stream Bit Rate Request (TMMBR) feedback message is defined in RFC 5104 [18]. This request and its corresponding Temporary Maximum Media Stream Bit Rate Notification (TMMBN) message defined in RFC5104 are used by a WebRTC receiver to inform the sending party that there is a current limitation on the amount of bandwidth available to this receiver. WebRTC endpoints that are sending media are required to implement support for TMMBR messages and shall follow bandwidth limitations set by a TMMBR message received for their SSRC. The sending of TMMBR messages is optional.

#### 14.1.1.8 RTP retransmission

The RTP Retransmission Payload Format RFC 4588 [19] supports retransmission of lost packets based on NACK feedback. Retransmission is useful if retransmitted packets arrive within the end-to-end delay requirements of the system. It is suitable for low RTT networks with relatively low observed packet loss.

If support for RTP retransmission payload format has been negotiated, the receivers required to support handling of RTP retransmission packets defined in RFC 4588 sent using SSRC multiplexing. Similarly, senders may use RTP retransmission packets defined in RFC 4588 for packets they retransmit using SSRC multiplexing.

The following example specifies two original, AAC and HEVC, streams on ports 49170 and 49174 and their corresponding retransmission streams on ports 49172 and 49176, respectively:

 m=audio 49170 RTP/AVPF 96

 a=rtpmap:96 MP4A-LATM/90000

 a=rtcp-fb:96 nack

 a=mid:1

 m=audio 49172 RTP/AVPF 97

 a=rtpmap:97 rtx/90000

 a=fmtp:97 apt=96;rtx-time=3000

 a=mid:2

 m=video 49174 RTP/AVPF 99

 a=rtpmap:99 H265/90000

 a=rtcp-fb:99 nack

 a=fmtp:99 profile-level-id=8;config=01010000012000884006682C209\

 0A21F

 a=mid:3

 m=video 49176 RTP/AVPF 100

 a=rtpmap:100 rtx/90000

 a=fmtp:100 apt=99;rtx-time=3000

 a=mid:4

## 14.2 Packet-loss handling mechanisms supported in RTC endpoint

### 14.2.1 General

This clause specifies some methods to handle conditions with packet losses.

The ‘a=bw-info’ attribute defined in clause 19 of TS 26.114 [20] allows for negotiating how much additional bandwidth (if any) may be used for application layer redundancy in the session. When application layer redundancy is used, the media bandwidth negotiated for the session may need to be increased, e.g., by increasing the value used for the b=AS bandwidth modifier. The b=AS bandwidth modifier is however only a single value, which also applies only to the receiving direction. When an RTC endpoint sends the SDP Offer/Answer, it is therefore not possible for the network and the other clients to know if the intention is to use the entire media bandwidth all the time (both with and without redundancy); or if the intention is to use the b=AS bandwidth only when redundancy is needed and to use a lower bandwidth when redundancy is not needed. It is also not possible to know what the RTC endpoint can do in the sending direction. The ‘a=bw-info’ attribute defined in clause 19 of TS 26.114 [20] offers an improved negotiation mechanism to better know what the RTC endpoint can do and what it intends to do.

Improved error robustness can be enabled by packet-loss handling procedures of the client or the codec in the terminal.

### 14.2.2 Video

#### 14.2.2.1 General

The RTC endpoints in terminal offering video shall support the packet-loss handling mechanisms defined in RFC 8834 [7] and RFC 8835 [8] with the below additions.

#### 14.2.2.2 NACK, PLI, SLI and FIR messages

RTC endpoints in terminal offering video should support transmission and reception of NACK RTCP messages, as an indication of non-received media packets. Note that by setting the bitmask of following lost packets (BLP) the frequency of transmitting NACK can be reduced, but the repairing action by the RTC endpoint receiving the message can be delayed correspondingly.

RTC endpoints offering video should support transmission and reception of Slice Loss Indication (SLI) RTCP messages and shall support reception of Picture Loss Indication (PLI) AVPF RTCP messages and Full Intra Request (FIR) codec control message (CCM) and react to those messages.

An RTC endpoint sending video should ignore FIR messages that arrive within Response Wait Time (RWT) duration after responding to a previous FIR message. Response Wait Time (RWT) is defined as RTP-level round-trip time, estimated by RTCP or some other means, plus twice the frame duration.

An RTC endpoint transmitting video can use NACK information, as well as the PLI, SLI and FIR messages, at its earliest opportunity to take appropriate action and recover video from errors for the RTC endpoint that sent the NACK, PLI, SLI or FIR messages. Recovery from error response is defined as sending a recovery picture that is equivalent to an Instantaneous Decoder Refresh (IDR) frame, sending Gradual Decoder Refresh (GDR), or retransmitting missing packets.

The usage of the AVPF and CCM feedback messages is negotiated by RTC endpoints using SDP offer/answer messages. Any AVPF or CCM feedback messages that have not been agreed in the SDP offer/answer negotiation should not be used in the session.

An example of how an SDP offer/answer indicates support for feedback of PLI, SLI, negative acknowledgement, and FIR is as below,

 v=0

 m=video 51372 RTP/AVPF 99

 a=rtpmap:99 H265/90000

 a=rtcp-fb:99 nack pli sli

 a=rtcp-fb:99 ccm fir

#### 14.2.2.3 TMMBR and TMMBN messages

The Temporary Maximum Media Bit-rate Request (TMMBR) and Temporary Maximum Media bit-rate Notification (TMMBN) messages of Codec-Control Messages (CCM) shall be supported by RTC endpoints in terminals supporting video. The TMMBR notification messages along with RTCP sender reports and receiver reports are used for dynamic video rate adaptation.

#### 14.2.2.4 RTP retransmission

An RTC endpoint in terminal may support RTP Retransmission as specified in clause 14.1.1.8.

An RTC endpoint in terminal supporting RTP Retransmission should offer retransmission for all media streams containing video. The binding used for retransmission stream to the payload type number is indicated by an rtpmap attribute. The MIME subtype name used in the binding is "rtx". The "apt" (associated payload type) parameter is used to map the retransmission payload type to the associated original payload type. The "rtx-time" payload-format-specific parameter indicates the maximum time a sender will keep an original RTP packet in its buffers available for retransmission.

# 15 RTC QoE metric reporting protocol

## 15.1 General

The Metrics Reporting API allows the Media Session Handler to send QoE metrics reports to the RTC AF. The metrics reporting procedure is as defined in clause 11.4.2 of TS 26.512 [6].

A RTC UE supporting Quality of Experience (QoE) shall report QoE metrics according to the QoE configuration. QoE reporting is optional, but if a RTC UE reports QoE metrics, it shall report all requested metrics.

## 15.2 Quality of Experience metrics definition

### 15.2.1 Introduction

This clause provides the general QoE metric definitions and measurement framework. A RTC UE supporting the QoE metrics feature shall support the reporting of the metrics in this clause. The metrics are valid for speech, video and text media, and are calculated for each measurement resolution interval "measureinterval". They are reported to the server according to the measurement reporting interval "reportinginterval" and after the end of the session.

The optional "measureinterval" field, if used, shall define a time over which each metrics value is calculated. The "measureinterval" field splits the session duration into a number of equally sized periods where each period is of the length specified by the "measureinterval" field. The "measureinterval" field is thus defining the time before the calculation of a QoE parameter starts over. If the "measureinterval" field is not present, the metrics resolution shall cover the period specified by the "measurerange" field. If the "measurerange" field is not present the metrics resolution shall be for the whole session duration.

The optional "measurerange" field, if used, shall define the time range in the stream for which the QoE metrics will be reported. There shall be only one range per measurement specification. The range format shall be any of the formats allowed by the media. If the "measurerange" field is not present, the metrics range shall be the whole call duration.

There are two kinds of timestamp defined i.e. *real time* (wall-clock time) and *media time*.

### 15.2.2 Corruption duration metric

Corruption duration, M, is the time period from the NPT time of the last good frame (since the NPT time for the first corrupted frame cannot always be determined) before the corruption, to the NPT time of the first subsequent good frame. A corrupted frame may either be an entirely lost frame, or a media frame that has quality degradation and the decoded frame is not the same as in error-free decoding.

A good frame is a completely received frame:

- where all parts of the image are guaranteed to contain the correct content; or

- that is a refresh frame, that is, does not reference any previously decoded frames; or

- which only references previously decoded good frames

Completely received means that all the bits are received and no bit error has occurred.

Corruption duration, M, in milliseconds can be calculated as below:

a) M can be derived by the client using the codec layer, in which case the codec layer signals the decoding of a good frame to the client. A good frame could also be derived by error tracking methods, but decoding quality evaluation methods shall not be used.

b) Alternatively, the corruption is considered as ended after N milliseconds with consecutively completely received frames, or when a refresh frame has been completely received, whichever comes first.

The optional configuration parameter N can be set to define the average characteristics of the codec. If N has not been configured it shall default to the length of one measurement interval for video media, and to one frame duration for non-video media.

The N parameter is specified in milliseconds and is used with the "CorruptionDuration" parameter. The value of N may be set by the server.

All the occurred corruption durations within each measurement period are summed and stored in the vector *TotalCorruptionDuration*. The unit of this metrics is expressed in milliseconds. Within each measurement period the number of individual corruption events are summed up and stored in the vector *NumberOfCorruptionEvents.*

The syntax for the metric "CurruptionDuration" is as defined in Table 15.2.2-1

Table 15.2.2-1: Corruption duration metric information for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| CorruptionDuration | Object |  |
|  | totalCorruptionDuration | unsignedLongVectorType | An unordered list of all occurred corrupt durations within each measurement period. |
|  | numberOfCorruptionEvents | unsignedLongVectorType | An unordered list of corruption events occurred within each measurement period. Within each measurement period the number of individual corruption events are summed up and stored.  |

### 15.2.3 Successive loss of RTP packets

The metric "SuccessiveLoss" indicates the number of RTP packets lost in succession per media channel.

All the number of successively lost RTP packets are summed up within each measurement resolution period of the stream and stored in the vector *TotalNumberofSuccessivePacketLoss*. The unit of this metric is expressed as an integer equal to or larger than 0. The number of individual successive packet loss events within each measurement resolution period are summed up and stored in the vector *NumberOfSuccessiveLossEvents.* The number of received packets are also summed up within each measurement resolution period and stored in the vector *NumberOfReceivedPackets.* These three vectors are reported by the RTC UE as part of the QoE report.

The syntax for the metric "SuccessiveLoss" is as defined in Table 15.2.3-1.

Table 15.2.3-1: Successive loss of RTP packets metric information for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| SuccessiveLoss | Object |  |
|  | TotalNumberofSuccessivePacketLoss | unsignedLongVectorType | An unordered list of all successively lost RTP packets within each measurement period. |
|  | NumberOfSuccessiveLossEvents | unsignedLongVectorType | The number of individual successive packet loss events within each measurement resolution period are summed up and stored in the vector. Provides an unordered list of successive packet loss events (occurred within each measurement period) measured during a metric reporting period.  |
|  | NumberOfReceivedPackets | unsignedLongVectorType | The number of received packets are summed up within each measurement resolution period and stored in the vector. |

### 15.2.4 Frame rate

Frame rate indicates the playback frame rate. The playback frame rate is equal to the number of frames displayed during the measurement resolution period divided by the time duration, in seconds, of the measurement resolution period.

For the Metrics-Name "framerate", the value field indicates the frame rate value. This metric is expressed in frames per second and can be a fractional value. The frame rates for each resolution period are stored in the vector *framerate* and reported by the RTC UE as part of the QoE report.

The syntax for the metric "framerate" metric is as defined in Table 15.2.4-1.

Table 15.2.4-1: Framerate metric for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| framerate | doubleVectorType | An unordered list of framerate values reported over a reporting period. The frame rates for each metric resolution period are stored in the vector. |

### 15.2.5 Jitter duration

Jitter happens when the absolute difference between the actual playback time and the expected playback time is larger than *Jitterthreshold* in milliseconds. The expected time of a frame is equal to the actual playback time of the last played frame plus the difference between the NPT time of the frame and the NPT time of the last played frame.

The optional configuration parameter *Jitterthreshold* can be set to control the amount of allowed jitter. If the parameter has not been set, it defaults to 100 ms. The *Jitterthreshold* parameter is specified in milliseconds and is used with the "JitterDuration" parameter. The value of *Jitterthreshold* may be set by the server.

All the jitter durations are summed up within each measurement resolution period and stored in the vector *TotalJitterDuration*. The unit of this metric is expressed in seconds and can be a fractional value. The number of individual events within the measurement resolution period are summed up and stored in the vector *NumberOfJitterEvents.* These two vectors are reported by the RTC UE as part of the QoE report.

The syntax for the metric "JitterDuration" is as defined in Table 15.2.5-1.

Table 15.2.5-1: Jitter duration metric information for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| JitterDuration | Object |  |
|  | TotalJitterDuration | doubleVectorType | All the jitter durations are summed up within each measurement resolution period and stored in the vector.  |
|  | NumberOfJitterEvents | unsignedLongVectorType | The number of individual events within the measurement resolution period are summed up and stored in the vector. Provides An unordered list of jitter events (occurred within each measurement period) measured during a metric reporting period.  |

### 15.2.6 Sync loss duration

Sync loss happens when the absolute difference between value A and value B is larger than *SyncThreshold* in milliseconds. Value A represents the difference between the playback time of the last played frame of the video stream and the playback time of the last played frame of the speech/audio stream. Value B represents the difference between the expected playback time of the last played frame of the video stream and the expected playback time of the last played frame of the speech/audio stream.

The optional configuration parameter s*yncthreshold* can be set to control the amount of allowed sync mismatch. If the parameter has not been set, it defaults to 100 ms. The s*yncthreshold* parameter is specified in milliseconds and is used with the "SynclossDuration" parameter. The value of *syncthreshold* may be set by the server.

All the sync loss durations are summed up within each measurement resolution period and stored in the vector *TotalSyncLossDuration*. The unit of this metric is expressed in seconds and can be a fractional value. The number of individual events within the measurement resolution period are summed up and stored in the vector *NumberOfSyncLossEvents.* These two vectors are reported by the RTC UE/endpoint as part of the QoE report.

The syntax for the metric "SynclossDuration" is as defined in Table 15.2.6-1.

Table 15.2.6-1: Syncloss duration metric information for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| SynclossDuration | Object |  |
|  | TotalSyncLossDuration | doubleVectorType | All the sync loss durations are summed up within each measurement resolution period and stored in the vector.  |
|  | NumberOfSyncLossEvents | unsignedLongVectorType | The number of individual sync loss events within the measurement resolution period are summed up and stored in the vector. Provides An unordered list of sync loss events (occurred within each measurement period) measured during a metric reporting period.  |

### 15.2.7 [Round-trip](https://www.rfc-editor.org/rfc/rfc8834#name-temporal-spatial-trade-off-) time

The round-trip time (RTT) consists of the RTP-level round-trip time, plus the additional two-way delay due to buffering and other processing in each RTC UE.

The last RTCP round-trip time value estimated during each measurement resolution period shall be stored in the vector *NetworkRTT*. The unit of this metrics is expressed in milliseconds.

The two-way additional internal client delay valid at the end of each measurement resolution period shall be stored in the vector *InternalRTT*. The unit of this metrics is expressed in milliseconds.

The two vectors are reported by the RTC UE as part of the QoE report.

The syntax for the metric "RoundtripTime" is as defined in Table 15.2.7-1.

Table 15.2.7-1: Round-trip time metric information for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| RoundtripTime | Object |  |
|  | NetworkRTT | unsignedLongVectorType | The last RTCP round-trip time value estimated during each measurement resolution period shall be stored in the vector. |
|  | InternalRTT | unsignedLongVectorType | The two-way additional internal client delay valid at the end of each measurement resolution period shall be stored in the vector. |

### 15.2.8 Average codec bitrate

The average codec bitrate is the bitrate used for coding "active" media information during the measurement resolution period.

For speech media the average codec bitrate can be calculated as the number of "active" speech bits received for "active" frames divided by the total time, in seconds, covered by these frames. The total time covered is calculated as the number of "active" frames times the length of each speech frame.

For non-speech media the average codec bitrate is the total number of RTP payload bits received, divided by the length of the measurement resolution period.

The average codec bitrate value for each measurement resolution period shall be stored in the vector *AverageCodecBitrate*. The unit of this metrics is expressed in kbit/s and can be a fractional value. The vector is reported by the RTC UE/endpoint as part of the QoE report.

The syntax for the metric "AverageBitrate " is as defined in Table 15.2.8-1.

Table 15.2.8-1: Syncloss duration metric information for Quality Reporting

|  |  |  |
| --- | --- | --- |
| Key | Type | Description |
| AverageBitrate | Object |  |
|  | AverageCodecBitrate | doubleVectorType | The average codec bitrate value for each measurement resolution period shall be stored in the vector.  |

## 15.3 Quality metrics reporting protocol

### 15.3.1 General

The quality metrics reporting protocol consists of:

- The XML-based report format defined in clause 15.3.2.

- The reporting protocol defined in clause 15.3.3.

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

### 15.3.2 Report format

The QoE report is formatted as an XML document that complies with the XML schema in Table 15.3.2-1.

Table 15.3.2-1: QoE Report XML schema

|  |
| --- |
| <?xml version="1.0"?><xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:3gpp:metadata:2023:RTC:receptionreport"xmlns:sv="urn:3gpp:metadata:2016:PSS:schemaVersion" xmlns="urn:3gpp:metadata:2023:RTC:receptionreport" elementFormDefault="qualified">  <xs:element name="ReceptionReport" type="ReceptionReportType"/> <xs:complexType name="ReceptionReportType"> <xs:choice> <xs:element name="QoeReport" type="QoeReportType" minOccurs="0" maxOccurs="unbounded"/> <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded"/> </xs:choice> <xs:attribute name="contentURI" type="xs:anyURI" use="required"/> <xs:attribute name="clientID" type="xs:string" use="optional"/> </xs:complexType> <xs:complexType name="QoeReportType"> <xs:sequence> <xs:element name="QoeMetric" type="QoeMetricType" minOccurs="1" maxOccurs="unbounded"/> <xs:any namespace="##other" processContents="skip" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:attribute name="periodID" type="xs:string" use="required"/> <xs:attribute name="reportTime" type="xs:dateTime" use="required"/> <xs:attribute name="reportPeriod" type="xs:unsignedInt" use="required"/> <xs:attribute name="mediaid" type="xs:unsignedInt" use="optional"/> <xs:attribute name="qoeReferenceId" type="xs:hexBinary" use="optional"/> <xs:attribute name="recordingSessionId" type="xs:hexBinary" use="optional"/> <xs:anyAttribute processContents="skip"/> </xs:complexType>  <xs:complexType name="QoeMetricType"> <xs:choice> <xs:element name="CorruptionDuration" type="CurruptionDurationType"/> <xs:element name="SuccessiveLoss" type="SuccessiveLossType" /> <xs:element name="framerate" type="doubleVectorType"/> <xs:element name="JitterDuration" type=" JitterDurationType"/> <xs:element name="Syncloss" type=" SynclossType"/> <xs:element name="RoundtripTime" type="RoundtripTimeType"/> <xs:element name="AverageBitrate" type="AverageBitrateType"/> </xs:choice> <xs:anyAttribute processContents="skip"/> </xs:complexType> <xs:complexType name="CurruptionDurationType"> <xs:attribute name="totalCorruptionDuration" type="unsignedLongVectorType" use="required"/> <xs:attribute name="numberOfCorruptionEvents" type="unsignedLongVectorType" use="required"/> <xs:anyAttribute processContents="skip"/> </xs:complexType> <xs:complexType name="SuccessiveLossType"> <xs:attribute name="TotalNumberofSuccessivePacketLoss" type="unsignedLongVectorType" use="required"/> <xs:attribute name="NumberOfSuccessiveLossEvents" type="unsignedLongVectorType" use="required"/> <xs:attribute name="NumberOfReceivedPackets" type="unsignedLongVectorType" use="required"/> <xs:anyAttribute processContents="skip"/> </xs:complexType>  <xs:complexType name="JitterDurationType"> <xs:attribute name="TotalJitterDuration" type="doubleVectorType" use="required"/> <xs:attribute name="NumberOfJitterEvents" type="unsignedLongVectorType" use="required"/> <xs:anyAttribute processContents="skip"/> </xs:complexType> <xs:complexType name="SynclossDurationType"> <xs:attribute name="TotalSyncLossDuration" type="doubleVectorType" use="required"/> <xs:attribute name="NumberOfSyncLossEvents" type="unsignedLongVectorType" use="required"/> <xs:anyAttribute processContents="skip"/> </xs:complexType> <xs:complexType name="RoundtripTimeType"> <xs:attribute name="NetworkRTT" type="unsignedLongVectorType" use="required"/> <xs:attribute name="InternalRTT" type="unsignedLongVectorType" use="required"/> <xs:anyAttribute processContents="skip"/> </xs:complexType> <xs:complexType name=" AverageBitrateType"> <xs:attribute name="AverageCodecBitrate" type="doubleVectorType" use="required"/> <xs:anyAttribute processContents="skip"/> </xs:complexType> <xs:simpleType name="unsignedLongVectorType"> <xs:list itemType="xs:unsignedLong"/> </xs:simpleType> <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="UnsignedIntVectorType"> <xs:list itemType="xs:unsignedInt"/> </xs:simpleType></xs:schema> |

### 15.3.3 Reporting protocol

The metrics reporting protocol is as defined in clause 11.4.3 of TS 26.512 [6]. For real-time media communication, clauses 15.3.1 and 15.3.2 specify the required MIME content type and metrics report format for the 3GPP urn:‌3GPP:‌ns:‌PSS:‌RTC:‌QM1 metrics reporting scheme.

For configuration done via the QMC functionality, the client shall also send QoE reports via the QMC functionality. For OMA-DM configuration, if a specific metrics server has been configured, the client shall send QoE reports using the HTTP (RFC 2616 [21]) POST request carrying XML formatted metadata in its body.

An example QoE reporting based on HTTP POST request signalling is as shown in Table 15.3.3-1.

Table 15.3.3-1: QoE Report HTTP POST format

|  |
| --- |
| POST http://www.exampleserver.com HTTP/1.1Host: 192.68.1.1User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0)Content-Type: text/xml; charset=utf-8Content-Length: 4408 |
| <?xml version="1.0"?><ReceptionReport contentURI="http://www.example.com/content/content.sdp" clientID="35848574673" xmlns="urn:3gpp:metadata:2023:RTC:receptionreport" xsi:schemaLocation="urn:3gpp:metadata:2023:RTC:receptionreport RTC-QoE-Report.xsd"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <QoeReport periodID="Period1" reportTime="2011-02-16T09:00:00" reportPeriod="500"> <QoeMetric> <CorruptionDuration> totalCorruptionDuration="480 0 120"  numberOfCorruptionEvents="5 0 2"</CorruptionDuration>  </QoeMetric> <QoeMetric> framerate="28.0 30.0 29.8"</framerate>  </QoeMetric> <QoeMetric> <JitterDuration> TotalJitterDuration="0 0.346 0"  NumberOfJitterEvents="0 1 0"</JitterDuration>  </QoeMetric> </QoeReport> <QoeReport periodID="Period2" reportTime="2011-02-16T09:08:20" reportPeriod="500"> <QoeMetric> <CorruptionDuration> totalCorruptionDuration="83 0 0"  numberOfCorruptionEvents="1 0 0"</CorruptionDuration> </QoeMetric> <QoeMetric> <JitterDuration> TotalJitterDuration="0 0 0"  NumberOfJitterEvents="0 0 0"</JitterDuration>  </QoeMetric> </QoeReport></ReceptionReport> |

# 16 Media capabilities

This specification primarily specifies the protocols and APIs for real-time communication. The APIs and protocols defined in this specification are not restricted to specific codecs or media capabilities. [In this specification, neither the requirements for RTC endpoints for audio codecs and processing as defined in IETF RFC 7874 [32], nor the requirements for RTC endpoints for video codecs and processing as defined in IETF RFC 7742 [33] apply. ]

However, to support minimum service interoperability, a terminal implementing the protocols and APIs defined in the present document should implement

- The UE codec requirements for speech as specified in TS 26.114 [20], if speech/audio is supported.

- The UE codec requirements for video as specified in TS 26.114 [20], if video is supported.

Transcoding free operation to UEs implementing IMS-based codecs and media capabilities as defined in TS 26.114 [20] should be supported. If supported, a terminal shall implement the UE codec and media handling requirements as specified in TS 26.114 [20].

Editor’s note: The bracketed text is not agreed and needs to be revised for language.

Annex A (informative):
RTC client in terminal

# A.1 Overview of high-level RTC data flow

The Real-Time Communication (RTC) system is designed based on the RTC architecture specified in [2] to handle an immersive media such as AR or XR. Figure 4.1-1 illustrates the high-level view of the RTC system that uses RTC AF and AS for realizing the services. RTC AF and AS provide the Control Plane (C-Plane) functionalities for setting up and controlling media and data sessions (U-Plane). The functionalities depend on supported collaboration scenarios, which are described in [2].



Figure A.1-1: High-level data flow showing two RTC endpoints in terminals.

NOTE 1: RTC AS may exist in the media/data path depending on the collaboration scenarios.

NOTE 2: RTC AF and AS are provided by MNO or 3rd party, depending on deployed collaboration scenario.

NOTE 3: Operator B is depicted for collaboration scenario 4. In other collaboration scenarios, "Operator B" is replaced with "Operator A", and the boxes representing the same functionalities are provided by an operator.

# A.2 Reference RTC endpoint model

The RTC endpoint supports a subset of WebRTC, which enables real-time communication via application programming interfaces (APIs), supporting audio, video, and generic data to be sent between peers. Functionalities of WebRTC are available as JavaScript APIs for browsers, and libraries for applications [12]. Information on use cases and requirements of WebRTC can be found in [22].

The functional components of a terminal including an RTC endpoint using 3GPP access are shown in figure A.2-1. Based on XR Baseline terminal architecture specified in TS 26.119 [23], Media Session Handler and Content delivery protocols are realized as a RTC MSH and WebRTC Framework, as specified in TS 26.506, respectively. Application may be a WebRTC application where C-plane is supported by RTC architecture or Web application (e.g., browser) where WebRTC APIs are involved for peer connection and immersive media delivery. Details of the associated APIs (RTC-6 and RTC-7) are specified in TS 26.510 [3]. The rest of functional blocks and interfaces are addressed in TS 26.119.



Figure A.2-1: Functional components of a terminal



Figure A.2-2: Functional components to handle immersive media

NOTE 1: Device information is assumed to be stored in the UE and loaded to the RTC endpoint during session setup.

NOTE 2: The RTC endpoint may exchange media and data with external devices tethered over wired/wireless links such as USB-C, 3GPP PC5 [24], or non-3GPP radio access technologies (e.g., Wi-Fi or Bluetooth).

NOTE 3: Text can be entered via user interface, typically available on display.

When a user launches a WebRTC application, a RTC MSH communicates with RTC AF to retrieve configuration information for session establishment. Note that this is exchanged via RTC-5 or alternatively, application-specific signalling function (e.g., collaboration scenario 1) as addressed in Annex A of TS 26.506 [2]. The configured information is then available to Application and Media Access Function via RTC-6 interface and the Application is ready to deliver an immersive media to the remote endpoint.

The following components are exchanged over WebRTC session.

- Video component: An RTC endpoint in terminal can be connected to one or more colour cameras, and/or to one or more depth cameras. The outputs of cameras may be pre-processed (e.g., converting data rates or representation formats) and the pre-processed media may be transmitted to the receiver of remote RTC endpoint. Then the remote client may post-process before they are input to displays (e.g., scene composition).

- Audio component: Similarly to video component, one or more microphones can be connected to an RTC endpoint. The captured audio bitstreams may perform pre-processing and/or post-processing to enhance the immersiveness (e.g., acoustically matching the perceived directions or locations of audio with those of video scenes).

- Sensor component: An RTC endpoint can utilize the information from various sensors for understanding environments, processing captured or received media, or other goals. The information may be locally utilized or transmitted with processed media.

- Signalling information: An RTC endpoint communicates to WebRTC signalling server to establish peer-to-peer connection. This signalling information is delivered through RTC-4s interface (as specified in clause 4.3.3 of TS 26.506) using WebSocket. Detailed protocol of WebRTC signalling is addressed in clause 13.2.

## A.2.1 Audio

### A.2.1.2 Microphone

An RTC endpoint in terminal can be connected to one or more microphones. The outputs of microphones are audio samples in 16-bit uniform Pulse Code Modulation (PCM) format. An RTC endpoint or audio infra may identify the direction of each microphone with a coordinate system described in figure A.2.1.2-1 and table A.2.1.2-1.



Figure A.2.1.2-1: Microphone array coordinate system

MicrophoneType, whose default value of 0 indicates an omni-directional microphone, identifies the microphones when other types are used. How to assign a value to each microphone type is left to the discretion of the implementation.

**Table A.2.1.2-1: Microphone description parameters**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Unit** | **Definition** | **Note** |
| Yaw (𝛹) | int | Direction angle | -31416 < 𝛹 ≦ 31416 |
| Pitch (𝛳) | int | Elevation angle | -31416 < 𝛳 ≦ 31416 |
| Roll (𝛷) | int | Rotation angle | -31416 < 𝛷 ≦ 31416 |
| MicrophoneType | int | A number that uniquely identifies microphone type | May be used for indicating vendor-defined microphone types |

NOTE 1: The coordinate system and two angles, yaw and pitch, are originally defined in [28] for computers.

NOTE 2: The positive X-, Y-, Z-axis shown in figure A.2.1.2-1 correspond to positive Z-, negative Y-, positive Z-axis of a coordinate system commonly used for sensors in mobile operating systems [25], [26].

### A.2.1.2 Pre/post-processor

An RTC endpoint in terminal may pre-process the outputs of microphones before they are input to audio encoders, e.g., for limiting bandwidth or converting the output into spatial audio representations. An RTC endpoint in terminal may post-process the outputs of audio decoders before they are input to speakers, e.g., for acoustically matching the perceived directions or locations of audio with those of video scenes.

### A.2.1.3 Codec

Audio codecs for the RTC endpoint in terminal are specified in [20], [23].

## A.2.2 Video

### A.2.2.1 Camera

An RTC endpoint in terminal can be connected to one or more colour cameras, and/or to one or more depth cameras. Depth cameras in this document typically consist of infrared projectors and infrared cameras that estimate the depth from measured time-of-flight or distortion of projected patterns. Resolutions and frame rates of the cameras are set to meet available bit-rate, complexity, storage, or nature of applications.

The output formats of color cameras, in the form of *Y*, *CR*, *CB* or *R*, *G*, *B* signals, are specified in [27]. The RGB signals can be input to (2D) video encoders. The output pixel of depth cameras has a value of a 16-bit unsigned number that represents the distance (in millimeters) from the reference point of a depth camera to a point in the captured scene, up to 32.7 meters. The depth signals for a rectangular area (map) can be input to a lossless or lossy encoder, or combined with RGB signals for further processing. Further each depth distance value represents a point in 3D space that was mapped onto a 2D image plane via a series of transforms illustrated below:



Figure A.2.2.1-1 Mapping of 3D points to 2D image plane

In the figure, [**R** **t**] represents the rotation and translation from a 3D world coordinate system to a 3D camera’s coordinate system, whose parameters can be supplied by UE’s motion sensors. K is the camera intrinsic matrix defined as



Table A.2.2.1-1: Parameters for camera intrinsic matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Unit** | **Definition** | **Note** |
| *fx* | float | X-axis focal length (in pixel) |  |
| *fy* | float | Y-axis focal length (in pixel) |  |
| *cx* | float | X-axis principle point (in pixel) |  |
| *cy* | float | Y-axis principle point (in pixel) |  |
| *s* | float | Skew coefficient | Zero if image axes are perpendicular |

NOTE 1: With infrared-based depth cameras, measurable distance is typically less than several meters.

NOTE 2: When the resolutions or aspect ratios of RGB and depth signals differ, the depth signals, whose resolutions are typically lower than those of RGB, can be interpolated to match the RGB signals.

### A.2.2.2 Pre/post-processor

An RTC endpoint in terminal may pre-process the outputs of cameras before they are input to video encoders, e.g., for converting the outputs into other representations, e.g., point cloud, or extracting scene information of local space. An RTC endpoint in terminal may post-process the outputs of video decoders before they are input to displays, e.g., for selecting scenes within FoV based on the extracted scene information or enhancing perceived video quality through appropriate filtering.

To support scaling and other rendering methods an RTC endpoint in terminal may identify further information on the size of captured 3D object based on the colour and depth camera properties, and data. The size of a 3D object captured with a colour camera, can be achieved with the help of the camera properties (focal length and sensor size) and the (estimated) distance to the subject. This means to estimating a physical size of an object (or user), first the image size of the object is determined in the captured image data, and secondly the relation between the image size and the physical size is determined with the help of the camera metadata (i.e., focal length) and the objects distance to the capture device (e.g., based on a depth camera or machine learning estimate). The resulting object size metadata comprises the size of the object to enable a rendering device or server to establish the “actual” size of the virtual object in the virtual environment in accordance with its physical size of the object in physical space. The size information can be signalled to a far-end RTC endpoint in terminal or conference managing server for scaling the 3D object to other objects or backgrounds. The size information may be transmitted periodically or in an on-demand fashion, depending on applications, and may also be used locally.

### A.2.2.3 Codec

Video codecs for the RTC endpoint in terminal are specified in [20], [23].

## A.2.3 Sensor

### A.2.3.1 General

An RTC endpoint in terminal can utilize the information from sensors for understanding environments, processing captured or received media, or other goals. The information can be locally utilized or transmitted with processed media, e.g., for aligning spaces in capturing and rendering process.

### A.2.3.2 Measure

For applications requiring the dimension of a captured object, e.g., for scaling or recognition, an RTC endpoint in terminal may measure its dimension, e.g., as a length, or a smallest rectangle or cuboid bounding the object. The dimension can be estimated using the relationship with physical distance and number of captured pixels.

The dimension may be represented with a length, a width, and a height, whose units are integer (in millimeters). How to overlay a rectangle or cuboid on a captured object is left to the discretion of the implementation. Depending on the required frequency of update, the dimension may be captured (and transmitted) periodically or upon request.

 Annex <X> (informative):
Change history

|  |
| --- |
| Change history |
| Date | Meeting | TDoc | CR | Rev | Cat | Subject/Comment | New version |
| 2022-05 | SA4#119 | S4-220768 |  |  |  | Skeleton for TS 26.113 | 0.1.0 |
| 2022-11 | SA4#121 | S4-221275 |  |  |  | Skeleton for TS 26.113 | 0.2.0 |
| 2023-04 | SA4#123e | S4-230590 |  |  |  | WebRTC Signalling Protocol (SWAP) | 0.5.0 |
| 2023-04 | SA4#123e | S4-230651 |  |  |  | Functional components of iRTC client in terminal | 0.5.0 |
| 2023-04 | SA4#123e | S4-230654 |  |  |  | High-level architecture, microphone description and transport protocol stack | 0.5.0 |
| 2023-05 | SA4#124 | S4-230747 |  |  |  | Editor’s update | 0.6.0 |
| 2023-05 | SA4#124 | S4-230980 |  |  |  | Updates to SWAP protocol | 0.6.0 |
| 2023-08 | SA4#125 | S4-231522 |  |  |  | Agreements in SA4#125: S4-231419, S4-231420 | 0.7.0 |
| 2023-11 | SA4#126 | S4-231748 |  |  |  | Agreements in SA4#126: S4-231848, S4-231897, S4-231898 | 0.8.0 |
| 2023-12 | SA#102 | SP-231305 |  |  |  | Version 1.0.0 created by MCC | 1.0.0 |
| 2024-01 | SA4#127 | S4-240062 |  |  |  | Editorial updates on references, clause numbers, and typosAgreement in post-126e telco: S4aR230138 | 1.0.1 |
| 2024-02 | SA4#127 | S4-240392 |  |  |  | Agreements in SA4#127:S4-240246, S4-240320, S4-240318, S4-240319, S4-240321 | 1.1.0 |
| 2024-04 | SA4#127-bis-e | S4-240789 |  |  |  | Agreements in SA4#127-bis-e:S4-240819, S4-240815, S4-240824 | 1.2.0 |