# ETSI TR 126 930 V19.0.0 (2025-10)

TECHNICAL REPORT

**LTE;**
**5G;**
**Study on the enhancement for Immersive Real-Time communication for WebRTC**
**(3GPP TR 26.930 version 19.0.0 Release 19)**

Reference
RTR/TSGS-0426930vj00

Keywords
5G,LTE

***ETSI***

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

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

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

# Intellectual Property Rights

## Essential patents

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

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

## Trademarks

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

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

# Legal Notice

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

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

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

# Modal verbs terminology

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

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

# Contents

# Foreword

This Technical Report 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.

# 1 Scope

The present document extends immersive Real-time Communication for WebRTC (iRTCW) and introduces a new concept called native WebRTC signalling.

This document includes following aspects:

1. Analysis of gaps and required enhancements of terminal device and network architectures including additional functional entities (e.g., WebRTC Signalling Server, ICE-STUN Server, IMS Interworking Gateway, NNI Gateway).
2. Impacts and possible enhancements for the WebRTC-based U-plane components in terms of adaptation, media handling, and cross-layer optimizations over 5G systems.
3. C-Plane signalling protocol details (e.g., based on JSON) for the common WebRTC-based immersive RTC session management.
4. Information elements in the C/U-Plane signal (including NNI) to enhance connectivity of media sessions with carrier assistance for WebRTC-based applications (including OTT applications).
5. Minimal functional capabilities needed to support the enhancements identified in 2, 3 and 4 (including transport, NAT-traversal, and XR conferencing), state transitions, and typical call flows.
6. Consideration of collaboration formation with other WGs in 3GPP and SDOs including IETF and W3C.
7. Enhancements for E2E QoS realizations over 5G systems for communications between MNOs and WebRTC clients operating over 5G access or non-5G access (e.g., Wi-Fi) using WebRTC-based transport. This also includes communication between WebRTC clients operating on tethering/tethered devices.
8. Security aspects and rate adaptation in tethered use cases (including coordination of Uu and non-3GPP access).

One of the main scopes of this document is to study the enhancements of RTC network for supporting collaboration scenario 4 specified in 3GPP TS 26.506 [12]. Collaboration scenario 4 enables inter-operable WebRTC services, which is extended collaboration scenario 3 with functions to support MNO to MNO inter-operability. This collaboration scenario 4 could include the roaming case, but this case is outside the scope of this document.

# 2 References

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

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".

[2] 3GPP TS 23.222: "Functional architecture and information flows to support Common API Framework for 3GPP Northbound APIs; Stage 2".

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

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

[5] 3GPP TS 23.502: "Procedures for the 5G System (5GS); Stage 2".

[5A] 3GPP TS 23.503: "Policy and charging control framework for the 5G System (5GS); Stage 2".

[6] 3GPP TS 23.548: "5G System Enhancements for Edge Computing; Stage 2".

[7] 3GPP TS 23.558: "Architecture for enabling Edge Applications".

[8] 3GPP TS 24.229: "IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3".

[9] 3GPP TS 24.371: "Web Real-Time Communications (WebRTC) access to the IP Multimedia (IM) Core Network (CN) subsystem (IMS); Stage 3; Protocol specification".

[10] 3GPP TS 26.113: "Real-Time Media Communication; Protocols and APIs".

[11] 3GPP TS 26.114: "IP Multimedia Subsystem (IMS); Multimedia Telephony; Media handling and interaction".

[12] 3GPP TS 26.506: "5G Real-time Media Communication Architecture (Stage 2)".

[13] 3GPP TS 26.510: "Media delivery; interactions and APIs for provisioning and media session handling".

[14] 3GPP TS 26.512: "5G Media Streaming (5GMS); Protocols".

[14A] 3GPP TR 26.806: "Study on Tethering AR Glasses – Architectures, QoS and Media Aspects".

[15] 3GPP TS 29.162: "Interworking between the IM CN subsystem and IP networks".

[16] 3GPP TS 29.165: "Inter-IMS Network to Network Interface (NNI)".

[17] 3GPP TS 29.501: "5G System; Principles and Guidelines for Services Definition; Stage 3".

[18] 3GPP TS 33.501: "Security architecture and procedures for 5G system".

[19] IETF RFC 768: "User Datagram Protocol".

[20] IETF RFC 791: "Internet Protocol".

[21] IETF RFC 3261: "SIP: Session Initiation Protocol".

[21A] IETF RFC 3264: "An Offer/Answer Model with the Session Description Protocol (SDP)".

[22] IETF RFC 3324: "Short Term Requirements for Network Asserted Identity".

[23] IETF RFC 3325: "Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks".

[24] IETF RFC 3489: "STUN – Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)".

[24A] IETF RFC 3629: "UTF-8, a transformation format of ISO 10646".

[24B] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax".

[24C] IETF RFC 5761: "Multiplexing RTP Data and Control Packets on a Single Port".

[24D] IETF RFC 5764: "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)".

[25] IETF RFC 5888: "The Session Description Protocol (SDP) Grouping Framework".

[26] IETF RFC 6120: "Extensible Messaging and Presence Protocol (XMPP): Core".

[27] IETF RFC 6455: "The WebSocket Protocol".

[28] IETF RFC 6598: "IANA-Reserved IPv4 Prefix for Shared Address Space".

[29] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".

[30] IETF RFC 7092: "A Taxonomy of Session Initiation Protocol (SIP) Back-to-Back User Agents".

[31] IETF RFC 7362: "Latching: Hosted NAT Traversal (HNT) for Media in Real-Time Communication".

[32] IETF RFC 7635: "Session Traversal Utilities for NAT (STUN) Extension for Third-Party Authorization".

[32A] IETF RFC 7807: "Problem Details for HTTP APIs".

[32B] IETF RFC 8035: "Session Description Protocol (SDP) Offer/Answer Clarifications for RTP/RTCP Multiplexing".

[33] IETF RFC 8200: "Internet Protocol, Version 6 (IPv6) Specification".

[34] IETF RFC 8224: "Authenticated Identity Management in the Session Initiation Protocol (SIP)".

[35] IETF RFC 8225: "PASSporT: Personal Assertion Token".

[36] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format".

[37] IETF RFC 8285: "A General Mechanism for RTP Header Extensions".

[38] IETF RFC 8441: "Bootstrapping WebSockets with HTTP/2".

[39] IETF RFC 8445: "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal".

[40] IETF RFC 8446: "The Transport Layer Security (TLS) Protocol Version 1.3".

[41] IETF RFC 8489: "Session Traversal Utilities for NAT (STUN)".

[42] IETF RFC 8656: "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)".

[43] IETF RFC 8588: "Personal Assertion Token (PaSSporT) Extension for Signature-based Handling of Asserted information using toKENs (SHAKEN)".

[44] IETF RFC 8825: "Overview: Real-Time Protocols for Browser-Based Applications".

[45] IETF RFC 8826: "Security Considerations for WebRTC".

[46] IETF RFC 8827: "WebRTC Security Architecture".

[47] IETF RFC 8829: "JavaScript Session Establishment Protocol (JSEP)"

[48] IETF RFC 8835: "Transports for WebRTC".

[49] IETF RFC 8838: "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol".

[50] IETF RFC 8839: "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)".

[50A] IETF RFC 8841: "Session Description Protocol (SDP) Offer/Answer Procedures for Stream Control Transmission Protocol (SCTP) over Datagram Transport Layer Security (DTLS) Transport".

[51] IETF RFC 8859: "A Framework for Session Description Protocol (SDP) Attributes When Multiplexing".

[52] IETF RFC 8864: "Negotiation Data Channels Using the Session Description Protocol (SDP)"

[53] IETF RFC 8866: "SDP: Session Description Protocol".

[54] IETF RFC 9110: "HTTP Semantics".

[55] IETF RFC 9111: "HTTP Caching".

[56] IETF RFC 9112: "HTTP/1.1".

[57] IETF RFC 9113: "HTTP/2".

[58] IETF RFC 9114: "HTTP/3".

[59] IETF RFC 9143: "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)".

[60] IETF RFC 9220: "Bootstrapping WebSockets with HTTP/3".

[61] IETF RFC 9293: "Transmission Control Protocol (TCP)".

[62] IETF RFC 9457: "Problem Details for HTTP APIs".

[63] AsyncAPI Initiative "AsyncAPI Specification v3.0.0" https://github.com/asyncapi/spec/releases/tag/v3.0.0

[64] OpenAPI Initiative "OpenAPI Specification v3.0.0" https://spec.openapis.org/oas/v3.0.0

[65] W3C Proposed Recommendation, "WebRTC 1.0: Real-time Communication Between Browsers", <https://www.w3.org/TR/webrtc/>.

# 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].

**Content Provider (CP)**: An RTC application provider who provide RTC services partially using operator's functionalities. CP connects to the operator network via UNI (RTC-4s/4m) as a WebRTC endpoint in order to use the operator's MF and WSF for the service specific content delivery.

**Originating RTC network**: A network which sends Media session set up request to another network (i.e., other operator network or SP network) over the NNI.

**Originating UE**: A RESPECT client on the UE which sends Media session set up request over the UNI.

**RESPECT client**: A RESPECT endpoint which acts as an WebSocket client.

**RESPECT endpoint**: A UE and RTC AS functionality supporting RESPECT protocol. RTC Aware Application on the UE or CP acting as UE, RTC AS for C-Plane signalling (i.e., WSF and IWF) that comply with RESPECT protocol are RESPECT endpoints. When distinction between UE/CP (acting as UE) and RTC AS is required, RESPECT endpoint (UE) or RESPECT endpoint (AS) is used respectively in this document.

**RESPECT server**: A RESPECT endpoint which acts as an WebSocket server.

**RTC network**: A DN dedicated to RTC. This network is compliant with the trusted DN of the RTC architecture defined 3GPP TS 26.506 [12].

**RTC resource**: A media which a media session is connected to. Media service such as conference room, media content for distribution and RESPECT endpoint are example of RTC resource.

**RTC user**: An RTC service user who connects to the RTC service by using RESPECT endpoint.

**Service provider (SP)**: An RTC application provider who provides RTC services with its own network. SP owns resources and functionalities within its network, and SP's network is connected to operator network via NNI.

**Terminating RTC network**: A network which receives Media session set up request from another network (i.e., other operator network or SP network) over the NNI.

**Terminating UE**: A RESPECT client on the UE which receives Media session set up request over the UNI.

**User Equipment (UE)**: It indicates the user equipment and servers acting as user equipment such as a content server of a content provider. User equipment includes an RESPECT endpoint.

For the purposes of the present document, the following terms and definitions given in 3GPP TS 29.165 [16] apply:

**Calling number verification using signature verification and attestation information**

For the purposes of the present document, the following terms and definitions given in IETF RFC 6455 [27] apply:

**WebSocket client**

**WebSocket server**

For the purposes of the present document, the following terms and definitions given in IETF RFC 8825 [44] apply:

**WebRTC Browser (also called a "WebRTC User Agent" or "WebRTC UA")**

**WebRTC Endpoint**

**WebRTC Non-Browser**

## 3.2 Symbols

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

| | |
|---|---|
| Rs-u | Reference Point between a WSF and a UE. |
| Rs-i | Reference Point between a WSF and another WSF in the same network (DN) or between a WSF and a WNSGF. |
| Rs-a | Reference Point between a WSF and a CSF. |
| Rs-n | Reference Point between a WNSGF and another WNSGF in an external network. |
| Rm-u | Reference Point between a WMCF and a UE. |
| Rm-i | Reference Point between a WMCF and another WMCF in the same network (DN) or between a WMCF and a WNMGF. |
| Rm-n | Reference Point between a WNMGF and another WNMGF in an external network. |
| Mc-i | Reference Point between a WSF and a WMCF. |
| Mc-r | Reference Point between a WNSGF and a WNMGF. |
| Rh-u | Reference Point between a CSF and UE. This reference point is used for providing CSF functionalities (e.g., application usage assistance such as downloading an application) to UE. |
| Rh-n | Reference Point between a CSF and Application service provider. This reference point is used for interaction between CSF and Application service provider for media session set up related interaction. |
| N5 | Reference Point between a WSF and PCF. |
| RTC-X | Reference Point between a ASWF and application service provider. |
| RTC-4 | Reference Point between an RTC network and a UE. |
| RTC-4m | Reference Point between a MF and a UE or between as ASWF and a UE. |
| RTC-4s | Reference Point between a WSF and a UE. |
| RTC-Y | Reference Point between two different operator's RTC networks or between an operator's RTC network and service provider's network. |
| RTC-Ym | Reference Point between a TGF and a TGF in another RTC network or a service provider's network. |
| RTC-Ys | Reference Point between a IWF and a IWF in another RTC network or a service provider's network. |

## 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].

| | |
|---|---|
| ASWF | Application Supporting Web Function |
| CP | Content Provider |
| CSF | Conference Supporting Function |
| IBCF | Interconnection Border Control Function |

| | |
|---|---|
| IWF | Inter-working Function |
| MDFC | Media Data Forwarding Control |
| MF | Media Function |
| NNI | Network to Network Interface |
| RESPECT | REaltime&REality media Setup Protocol, Extensible and CompacT |
| SEPP | Security Edge Protection Proxy |
| SP | Service Provider |
| TGF | Transport Gateway Function |
| UNI | User to Network Interface |
| WMCF | WebRTC Media Centre Function |
| WNMGF | WebRTC NNI Media Gateway Function |
| WNSGF | WebRTC NNI Signalling Gateway Function |
| WSF | WebRTC Signalling Function |

# 4 Motivations for native WebRTC signalling and assumptions

## 4.1 General

In 3GPP, the use of WebRTC technologies has been investigated since Release-12 (around 2014). They are a network-based architecture for WebRTC access to IMS specified in Annex U to 3GPP TS 23.228 [3] and its stage 3 has specified in 3GPP TS 24.371 [9]. They define functional entities including WIC (WebRTC IMS Client) and eP-CSCF (P-CSCF enhanced for WebRTC). The eP-CSCF is assumed to be located in the home IMS domain and communicates with other IMS entities using the existing interfaces. For the C-plane signalling between WIC and eP-CSCF, those specifications specify an option to use SIP over WebSocket, whose information model can be used for options other than SIP over WebSocket. However, a lot of real-time communication services are familiar with JSON based light weight signalling protocol which is flexible, extensible, and can be optimized for new XR conversational applications. These characteristics remind us of the original design principle of WebRTC. WebRTC, by its inherent characteristics, does not regulate C-plane signalling and allow a wide range of C-plane signalling. This document looks over this design principle again and investigates a new SIP-decoupled C-plane signalling, called native WebRTC.

Regarding the level of signalling details, 3GPP TS 24.371 [9] specifies a signalling transport mechanism using SIP over WebSocket, but it is not a mandatory mechanism for eP-SCSF. Even though there are other options such as XMPP or other application protocols over WebSocket, a RESTful based interface, etc., 3GPP TS 24.371 [9] does not specify any details of C-plane signalling using other options. Each service provider (e.g., operator) develops its own application by following the guidelines in 3GPP TS 24.371 [9]. Its subscriber downloads the application and connects to the service and other subscribers only within the same service. Detailed C-plane signalling is left open to each operator's design. In contrast, this document identifies a new C-plane signalling in detail (as an interface specification) to the extent that client implementations based on it have enough interoperability. This realizes connectivity to any operators or roaming services for new XR real-time communications. Operators can provide the interface common to them according to well-defined C-plane signalling specifications. Clients can connect to any operators via the interface (see Figure 4.1-1).

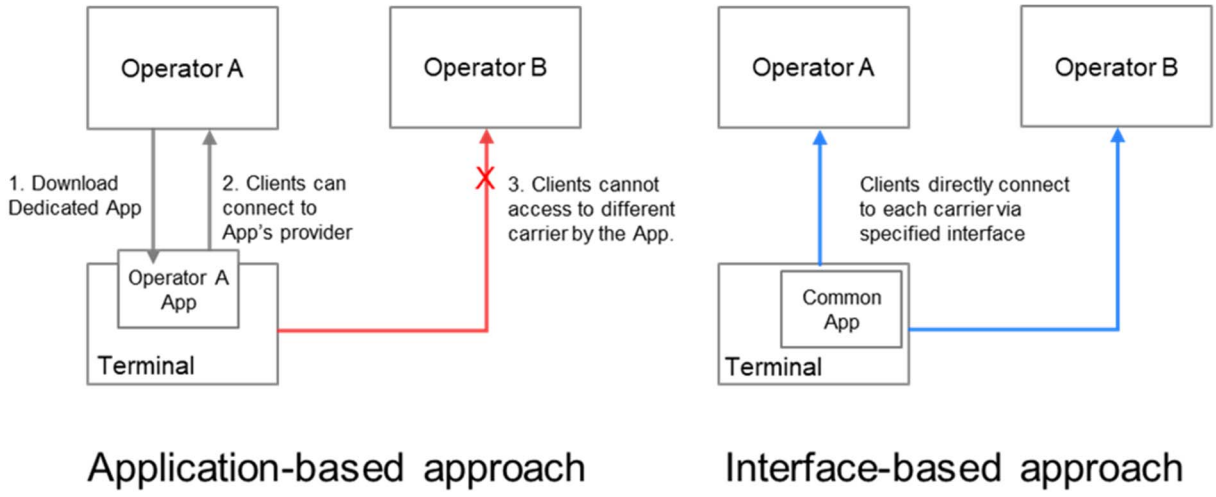


**Figure 4.1-1: Two approaches for defining specifications and their application connectivity**

## 4.2 High-level network model and target interfaces

The new C-Plane signalling protocol (i.e., RESPECT) studied in this document is intended for various media session control on the following interfaces:

- UNI: The interface between operator network and UE (e.g., smart phone, content server of the content provider).
- NNI: The interface between the two different operator networks, or that between operator network and service provider network.

A UE and a content provider can set up a media session by using RESPECT for session control on the UNI. A service operator can set up a media session by using RESPECT for session control on the NNI. Figure 4.2-1 shows the high-level network model indicating above interfaces and media sessions established via the functional entities supporting RESPECT (which described in clause 6.2) by using RESPECT.

There are following benefits to using RESPECT.

- A UE (including the equipment of content provider) which is compliant with the RESPECT can connect to any operator network which supports the RESPECT and set up a media session in the operator network, based on the same signalling requirements.
- A UE (including the equipment of content provider) which is compliant with the RESPECT can connect to services provided by other operator network or service provider network via NNI, based on the same signalling requirements.
- Content Providers can set up an operator assisted media session (e.g., media session with QoS) with UEs connected to the Operator Network via the UNI, by connecting to the operator network via the UNI.- Service providers can set up an operator assisted media session (e.g., media session with QoS) with UEs connected to the operator network via the UNI, by connecting to the operator network via the NNI.

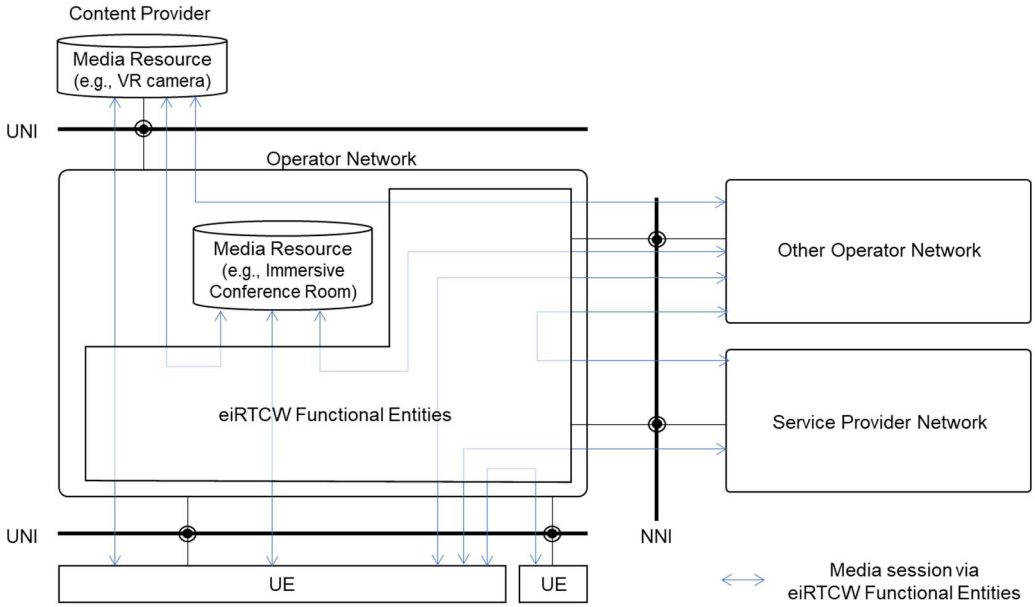


**Figure 4.2-1: High-level network model and interfaces**

**<Terminology>**

**User Equipment (UE)**: It indicates the user equipment and servers acting as user equipment such as a content server of a content provider. User equipment includes an WebRTC endpoint supporting RESPECT.

**Operator**: Mobile and Fixed network operator who provides telecommunication services.

**Service Provider (SP)**: 3rd party service provider who connects its service to operator network via NNI. OTT service is one of the typical services provided by service provider. Network Operator is excluded from the definition of this terminology in this document.

**Content Provider (CP)**: 3rd party service provider who connects its service to operator network via UNI. Network Operator is excluded from the definition of this terminology in this document.

**UNI**: User-to-Network Interface. The interface between UE and Network.

**NNI**: Network-to-Network Interface. The interface between two different Networks.

## 4.3 C-plane Signalling comparison

The C-plane signalling can be expressed as follows. Now, there are roughly four possible methods, classified in terms of their protocol stacks (see Figure 4.3-1).

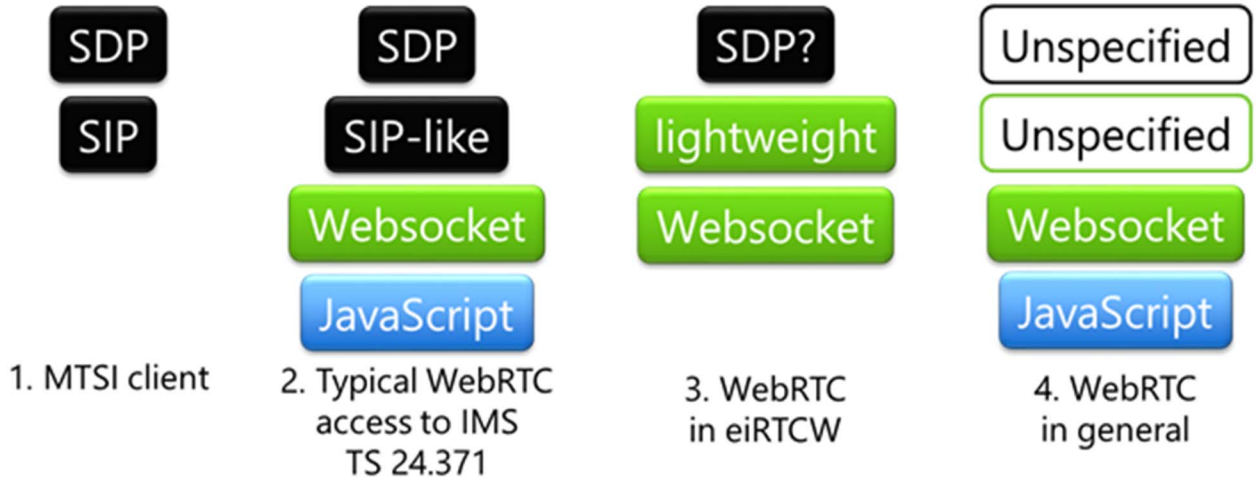


**Figure 4.3-1: Comparison of protocol stacks**

The first method is MTSI-based, using SIP and SDP. General C-plane signalling requirements for conversational services can be covered by SIP. Interoperability is fine with the existing 5G core network. It is to be treated in IMS-based AR Conversational Services (IBACS).

The second is the method specified in 3GPP TS 24.371 [9]. It enables the WebRTC clients to communicate over an IMS-based core network; only the interfaces for downloading dedicated applications and the signalling path using WebSocket are specified for C-plane signalling. Ordinary implementations adopt SIP-like protocols over WebSocket. In most cases, it is partially SIP-compliant or tightly coupled with SIP to adapt WebRTC clients in IMS domain.

The third method is an alternative to the second method that uses SIP-like protocol over WebSocket. The third method uses another signalling protocol over WebSocket, but SIP-decoupled approaches are investigated. It can be more lightweight, omitting features that is not used in XR conversational. Some constraints on SDP are necessary for interoperability. Non-browser based implementations are also in the scope. This method is the main subject of this document.

The other is a general WebRTC protocol stack that is not specified and left open to the users (i.e., service providers). C-plane may be SIP, XMPP, http, etc. A general WebRTC application uses SDP syntax compliant to RFC 8866 [53] for its internal representation, when setting the local and remote descriptions. C-plane protocol may have its own on-the-wire format for SDP, which can be constructed from SDP and be serialized out to SDP.

# 5 Key issues

## 5.1 General

This clause describes the key issues of this document.

## 5.2 Key Issue #1: Enhancements on RTC architecture

As described in 3GPP TS 26.506 [12], the detailed scenario and the RTC architecture for collaboration scenario 4 is FFS. This key issue identifies the scenarios and the possible enhancements on RTC architecture to realize collaboration scenario 4 in addition to collaboration scenario 3, based on the high-level network model described in clause 4.2.

This key issue includes:

1) Possible enhancements on functional entities and enhanced RTC architecture

   - Study possible enhancements on functional entities and RTC architecture for collaboration scenario 4 in addition to collaboration scenario 3 based on general WebRTC implementation viewpoint.

2) Interaction with 5GC

- To realize the QoS control, study the interaction between functional entities in the enhanced RTC architecture and those in 5GC.

3) Media connection model

- Study the target use cases (i.e., connection model) of user plane (U-Plane) and considerations of QoS-enabled end-to-end path.

4) IP addressing

- Study the considerations on IP addressing related issues and identify the possible additional enhancements of ICE functionality.

5) Alignment and gap analysis between the enhanced RTC architectures and the current RTC architecture.

- Study the alignment between the enhanced RTC architecture derived from 1) – 4) and the current RTC architecture defined in 3GPP TS 26.506 [12]. This also includes gap analysis of functionalities between the architectures.

6) Enhanced architecture for collaboration scenario 4

- Study the expected architecture variant for the collaboration scenario 4 and enhancements on the current RTC architecture defined in 3GPP TS 26.506 [12], based on the gap analysis studied in 5). This also includes the clarifications on the focused functions and interfaces in this document.

## 5.3 Key Issue #2: Functional requirements for C-Plane

This key issue identifies the functional requirements for C-Plane based on the architecture proposed in Solution #1 of this document.

This key issue includes:

1. Functional requirements for C-Plane; and
2. Protocol Stack for C-Plane interfaces.

## 5.4 Key Issue #3: C-Plane signalling protocol

This key issue studies the details of C-Plane signalling protocol based on the possible architecture studied in Solution #1, the functional requirements for C-Plane studied in Solution #2 and the other related requirements in the other solutions.

## 5.5 Key Issue #4: Functional requirements for U-plane

This key issue identifies the functional requirements for U-Plane on the derivative architecture. This key issue includes:

1) Functional requirements for U-Plane; and
2) Protocol stack for U-Plane interface.

## 5.6 Key Issue #5: Functional requirements for service control API

### 5.6.1 General

This key issue identifies the functional requirements for service control API that is required for the content provider, a form of RTC application provider, to provide an RTC services.

## 5.6.2 RTC Application Provider

### 5.6.2.1 General

In RTC architecture defined in 3GPP TS 26.506 [12], third-party application provider that provides WebRTC-based immersive RTC services is referred to as an RTC application provider. Such RTC application providers have several resources and functions to achieve service delivery.

NOTE 1: Depending on the service delivery form of the RTC application providers (e.g., content provider detailed in clause 5.6.2.3), several resources and functions need to be deployed in the operator network.

RTC application provider is thought to have following resources:

1) **Service specific content**: content for RTC services (e.g., back-ground graphical image/video and sound effects in VR conference services)

2) **RTC ID resource**: resources reserved in control plane and used for establishing connection to RTC exchange resources. When accessing RTC services, a UE indicates this resource as the destination during media session setup. This resource is identified by a URI called RTC resource ID. RTC ID resource is composed of the RTC resource ID and properties associated with the RTC resource ID.

3) **RTC exchange resource**: resources reserved in user plane. For example, individual RTC exchange resource is associated with a VR space or a conference room.

NOTE 2: RTC ID resources and RTC exchange resources are collectively referred to as RTC resources.

Also, RTC application provider is thought to have following functions:

1) **Service logic function**: a function to execute the logic to realize RTC services. Following functionalities are included:

   a1) **RTC ID resource manager**: a functionality responsible for controlling the creation, update, and deletion of RTC ID resources.

   a2) **RTC ID resource handling enforcer**: a functionality to perform actual creation, update, and deletion procedures on RTC ID resources as directed by the RTC ID resource manager.

   b1) **RTC exchange resource manager**: a function responsible for controlling the creation, update, and deletion of RTC exchange resources.

   b2) **RTC exchange resource handling enforcer**: a functionality to perform actual creation, update, and deletion procedures on RTC exchange resources as directed by the RTC exchange resource manager.

   c1) **Connection control manager**: a C-plane functionality responsible for determining the acceptance of UE's connection requests targeting a specific RTC ID resource.

   c2) **Connection control enforcer**: a functionality to perform connection control as directed by the connection control manager.

   d1) **Media data forwarding control manager**: a U-plane functionality responsible for configuring the rule of forwarding control for individual media or data exchanged in U- plane.

   d2) **Media data forwarding control enforcer**: a functionality to perform media data forwarding control as directed by the media data forwarding control manager.

NOTE 3: Functionalities included in service logic function can be classified into the following two types. One is "service logic manager", including a1), b1), c1) and d1). The other is "service logic enforcer", including a2), b2), c2) and d2).

2) **WebRTC endpoint function**: a function to terminate WebRTC communication. Following functionalities are included:

   a) **C-plane signalling**: a functionality to perform C-plane signalling using the RESPECT in this document.

b) **U-plane transport**: a functionality to perform U-plane media communication using WebRTC protocol stack. Service specific content is provided through this functionality.

The control achievable through these functionalities above is referred to as "service control" in this document.

An RTC application provider performs as either service provider or content provider.

### 5.6.2.2 Service provider

An RTC application provider who provides RTC services with its own network is referred to as service provider (SP) in this document. SP owns resources and functionalities described in clause 5.6.2.1 within its network, and SP's network is connected to operator network via NNI. (See Figure 5.6.2.2-1.)

When a connection is initiated from the UE toward the SP, the SP uses the ID assigned and verified by the operator in order to control the UE's connections and provides services considering status of subscription associated with that ID.

NOTE: When accessing SP's services, the UE initially attempts to connect to the operator's network using the ID assigned by the operator. After successful completion of authentication, the UE can use this ID as a network-assigned ID for establishing media sessions. The operator also treats this ID as a network-asserted ID through its verification. When a media session establishment request is made from the UE toward the SP, the SP will receive this network-asserted UE's ID in the signalling message from the operator's network.

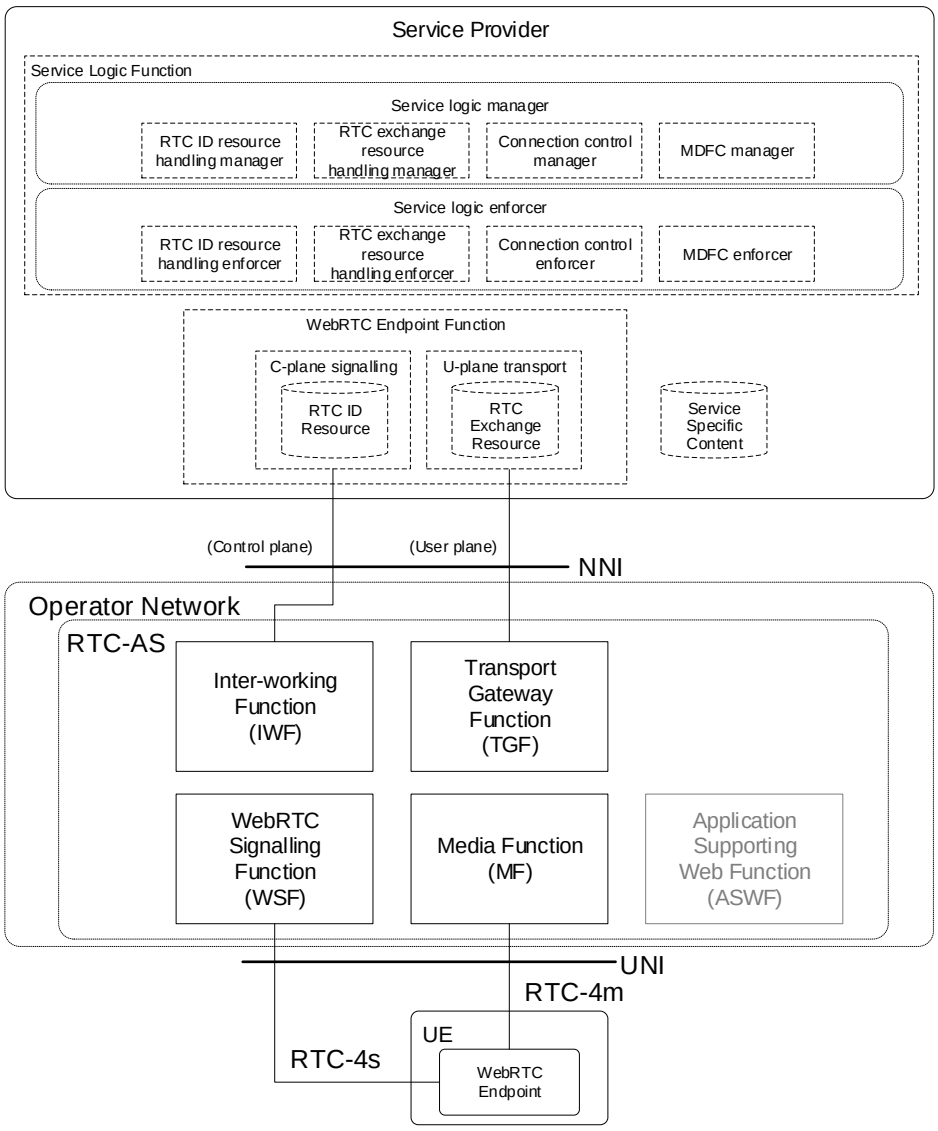


**Figure 5.6.2.2-1: Service provider-operator connection and functional deployment diagram**

#### 5.6.2.3 Content provider

Contrary to SP, an RTC application provider who provide RTC services partially using operator's functionalities is referred to as content provider (CP) in this document. CP connects to the operator network via UNI (RTC-4s/4m) as a WebRTC endpoint in order to use the operator's MF and WSF for the service specific content delivery. (See Figure 5.6.2.3-1.)

Since operator network accommodates CP via UNI, RTC ID/exchange resources are reserved within the operator's WSF and MF respectively. Therefore, in order to perform dedicated procedures on these RTC resources, RTC ID / RTC exchange resource handling enforcer functionalities needs to be deployed in the operator's WSF and MF. Also, connection control enforcer and media data forwarding control enforcer needs to be deployed in the operator network, as CP depends on C-Plane signalling and U-Plane transport functionalities of the operator network. As a result, CP's service logic managers in service logic function requires APIs for controlling related service logic enforcers deployed in the operator network. Since this functionality is not provided over RTC-4s/4m, it is considered to be provided through a different reference point than RTC-4s/4m. Required functionalities are to be discussed in clause 5.6.3.

NOTE 1: The architectural requirements and reference point name for the APIs between CP's service logic manager and operator's service logic enforcers is FFS and will be addressed in the future work.

Also, CP and SP differ in terms of UE authentication by the service logic function. In the interconnection scenario between operator and SP networks, a UE connected to an operator network always uses an ID provided by the operator. In contrast, when RTC application provider provides its RTC service as CP, there are two possible network-asserted IDs that are used for media session setup by UE:

**Operator-provided ID**: the ID allocated to UE by the operator and managed by the operator. After successful completion of authentication, UE can use this ID as the network-asserted ID when requesting media session setup. The operator network will treat this ID as network-asserted ID and CP will perform connection control and provide service considering status of subscription related to the ID.

**CP-provided ID:** the ID allocated to UE by CP and managed by CP. The operator queries the CP to verify the authenticity of ID. After successful completion of authentication by CP, UE can use this ID as the network-asserted ID when requesting media session setup. The operator network will treat this ID as network-asserted ID and CP will perform connection control and provide service considering status of subscription related to the ID.

NOTE 2: The querying the CP to verify the authenticity of CP-provided ID is performed at WSF in the authentication procedure triggered when receiving the signalling message for authentication from UE. The details are available in clause 5.4 (Key Issue #3: C-Plane signalling protocol).

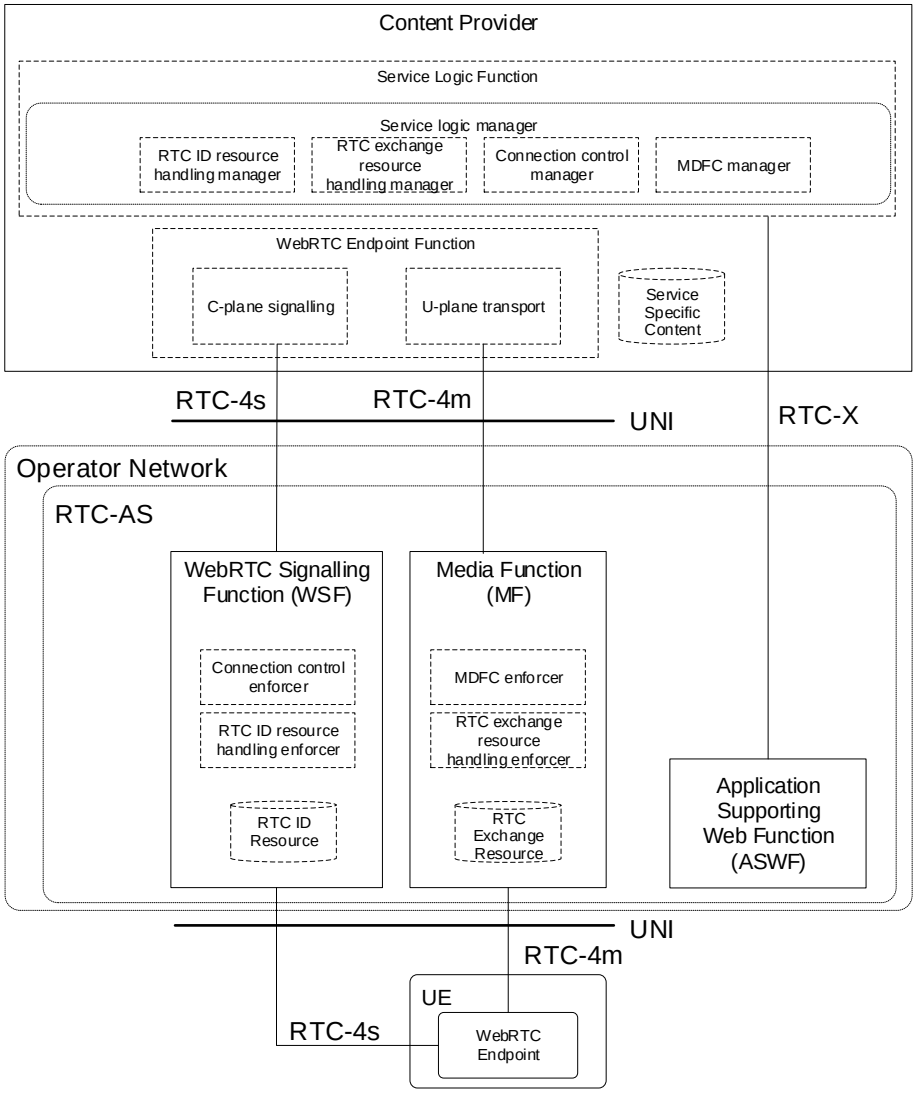


**Figure 5.6.2.3-1: Content provider-operator connection and functional deployment diagram**

## 5.6.3 Functional requirements for service control

### 5.6.3.1 General

As mentioned in clause 5.6.2.3, CP requires APIs for controlling each service logic enforcer from each service logic manager. It is assumed that the service control instructed by CP and performed by the operator network requires the capability to describe instructions related to the following three functionalities at least:

- CRUD of RTC ID resource and associated properties
- User connection control using asserted identity
- Media data forwarding control

### 5.6.3.2 CRUD of RTC ID resource and associated properties

As described in clause 5.6.2, CP provides its RTC service by registering RTC ID resources to operator's network. RTC ID resource that serves as the destination for UE and associated properties are to be registered (e.g., notification settings for events related to the RTC ID resources, expiration timing of RTC ID resource, and handling of new connections for controlling graceful shutdown).

In the deployment of CP's functions, RTC ID resource handling enforcers belong to the operator. This means that CP initiates CRUD procedure requests, but the actual handling of RTC ID resources is performed by the operator based on these requests.

As procedures on the RTC ID resources, registering RTC ID resource, update to the properties of registered RTC ID resources, deletion of RTC ID resources, and acquisition of current RTC ID resource status are essential functions instructed by CP and performed by operator network.

#### 5.6.3.3 User connection control using asserted identity

CP needs to be able to instruct connection control as part of functionalities supported in service logic manager when UE indicates an RTC ID resource as its destination and attempts to connect to the corresponding RTC exchange resource. Connection control manager in CP's service logic function can determine the acceptance of connection from its own managed user subscription information and network-asserted ID. There are two possible methods for the WSF performing the functionality of connection control enforcer to process connection control:

- CP registers specific connectable CP-provided or operator-provided IDs in the CRUD procedures described in clause 5.6.3.2.
- Operator network queries CP to determine whether to accept the connection from a UE having a CP-provided or operator-provided ID to an RTC ID resource.

#### 5.6.3.4 Media data forwarding control

In general, RTC application providers determine how individual audio/video media and non-media data from UNI are transmitted or terminated, reflecting service requirements (e.g., user experience, security., etc). Such process is defined as media data forwarding control (MDFC) in this document. Individual audio/video media refers to a single track of audio or video. Also, individual non-media data refers to the data other than audio or video that is transmitted and received over a single data channel. These are collectively referred to as RTC media/data.

MDFC deals with connections of individual RTC media/data to the endpoints of specific UE or service specific content function through MF's input and output.

For example, when it comes to an audio media in a conference, the upstream audio media from a specific UE is duplicated by the MF and sent to all other participants' UEs. On the other hand, when providing services such as audio analysis or recording, it is expected that only the audio media of UE which has consented to information collection by CP will be sent to the audio analysis or recording module. In some cases, MF can simply duplicate and transfer the video media without any processing, while in other cases, it can terminate the video media, perform video processing such as motion detection, and then send the video media as avatar animations. When using the Data Channel for text messages, in an open chat where all participants can see, the chat text is sent to all UEs. However, for the private messages, the text is only sent to specific UEs and not to others.

It is a part of MDFC functionality that optimizing the allocation of internal resource of MF depending on the patterns of RTC media/data duplication and UE connectivity. Examples of use cases with different patterns of RTC media/data duplication and UE connectivity are:

- Conference where audio and video media are connected in a full-mesh manner between participants
- Webinar where only the presenters' audio and video media are delivered to all participants
- Large-scale broadcasting where one presenter's audio and video media are delivered to much larger audience

MDFC mentioned above cannot be described by SDP. Therefore, SP implements the MDFC as an internal logic within the service logic function. In the CP's service delivery, the transfer and termination of RTC media/data are processed by the MF including MDFC enforcer, and CP's MDFC manager is responsible for creating MDFC rules and instructing MDFC enforcer. An API is required for CP to instruct the MDFC enforcer in operator.

### 5.6.4 Summary

SP performs service control using its own network functions. On the other hand, in the case of CP performing service control, at least three functions described in clause 5.6.3 should be provided through APIs to allow CP's service logic mangers to instruct and the operator's service logic enforcers to perform those functions. In this key issue, the procedures and APIs which enable CP to achieve above functionalities and related processes are to be studied.

## 5.7 Key Issue #6: WSF discovery mechanism

In collaboration scenario 3 and collaboration scenario 4 of Real-Time Communication (RTC) for WebRTC, RTC application using WebRTC connects to a WebRTC signalling function (WSF) specified in 3GPP TS 26.506 [12] in order to setup media session, where the following steps are expected to be applied at the UE.

1) Download an RTC application;

2) launch the RTC application and identify the local WSF in the operator network where the UE attached to; and

3) connect to the WSF and use the RTC services.

For step 2), the RTC application (i.e., WebRTC endpoint) is expected to connect to a local WSF in an operator's network where the UE attached to. Then the RTC application needs to identify the local WSF in the connected operator's network.

To enable WebRTC endpoint to identify the WSF without specific setting per connected operator's network, it is desirable to standardize a common WSF discovery mechanism for zero configuration. This will make benefits for both user and operator as follows:

- User perspective:
  Users do not need to change the application and/or parameters depends on the connected operator network. Then the user can use the RTC application without having to worry about the connected operator network.

- Operator perspective:
  Connection management between WebRTC endpoint and WSF becomes easier, since the RTC application behavior for WSF discovery is standardized and operators are able to control the connected WSF by modification of the operator network settings.

This key issue identifies the WSF discovery mechanism without user manual setting and applicable regardless of the connected operator network.

NOTE: Step 1) (Downloading an RTC application) is outside the scope of this key issue. Step 3) (Connecting to the WSF and using the RTC services) is studied in Key Issue #4, then step 3) is also outside the scope of this key issue.

## 5.8 Key Issue #7: Interworking with IMS network

This key issue addresses the functional requirement for interworking between RTC network and IMS network.

This key issue includes:

1) applicable interface between RTC network to IMS network,

2) supported interworking scenarios between RTC network and IMS network,

3) unctional requirements for RTC-IMS interworking and

4) RTC architecture enhancement for RTC-IMS interworking.

## 5.9 Key Issue #8: Protocol-level interworking between RTC network and IMS network

This key issue addresses the protocol-level interworking between RTC network and IMS network at the boundary of RTC network, based on the functional requirements and the enhanced RTC architecture described in Solution #4.

This key issue includes:

- C-Plane signalling interworking and

- U-Plane media related interworking.

## 5.10 Key Issue #9: Tethered cases

For a device with limited computing capability and communication capability due to the size and weight constraints such an AR glasses, it may be beneficial to tether to a nearby device that has stronger computing capability and communication capability such as a smart phone. The tethering case was studied in the SA4 SmarTAR study item and the outcome is documented in 3GPP TR 26.806 [14A].

The possible scenarios can be enumerated in several aspects:

- The type of tethered devices:
  Among the types of AR glasses studied in 3GPP TR 26.806 [14A], two types are relevant to eiRTCW: the AR glasses only as a display of the tethered device, and the AR glasses as both a display and a host carrying out XR Runtime core functions. If the AR glasses serves only as a display, it may not have an IP address, in which case the content for display can be sent over the tethering link via L2 forwarding, and it may have an IP address, which, however, is invisible to the other WebRTC Endpoint.

- How many of the end devices are tethered:
  Tethering may occur in only one of the end devices, or in both end devices. As an example, Figure 5.10-1 shows only one tethered device, which is the AR glasses.

- Tethering link is a 5G link or a non-3GPP link:
  The tethering link could be a sidelink (defined by the PC5 interface), which is a 5G link. Alternatively, the tethering link can be a non-3GPP link such as a Wi-Fi link, as shown in Figure 5.6-1.

The combination of the aspects considered above results in many scenarios, and only one of them is shown in Figure 5.10-1.

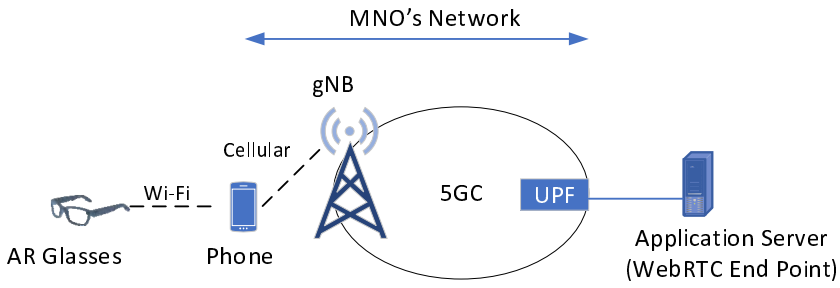


**Figure 5.10-1: A tethering case with only one MNO involved.**

The key issue may be decomposed into several sub-issues:

- Key issue #9-1: Which scenarios are in the scope of tethered cases?
- Key issue #9-2: Should the WebRTC Endpoint be on the tethered device or on the tethering device?

  To elaborate, in Figure 5.10-1, one of the WebRTC Endpoint is on the Application Server (e.g., serving as a cloud gaming server), but it is not clear where the other WebRTC Endpoint should be.

- Key issue #9-3: How to authenticate the tethered device?
- Key issue #9-4: Are there any difference between WebRTC endpoint on the tethered device and the WebRTC endpoint on a UE?
- Key issue #9-5: How to provide E2E QoS when there are non-3GPP networks also involved?

The support for multiple MNO's on the E2E path is within the scope of this key issue.

## 5.11 Key Issue #10: Security considerations

This key issue addresses the security related considerations specific to real-time media communicaiton by WebRTC-based media session setup.

In IETF RFC 8825 [44] (which gives the WebRTC overview), the following items are described as security considerations:

a) security of the components,

b) security of the communication channels, and

c) security of the partner's identities.

NOTE: IETF RFC 8826 [45] and IETF RFC 8827 [46] describes further security considerations on real-time communication on the Web.

Regarding a), RTC application is outside the scope of 3GPP TS 26.506 [12] and RTC AF/RTC AS of this specification are defined as located in trusted DN - this means the RTCAF/RTC AS are protected by adequate network domain security. Then this document assumes that the security of components in RTC network is guaranteed.

Regarding b), secure transport protocol is applied for both C-plane and U-plane of RTC network in the Release-18 stage 3 work (i.e., WI: iRTCW). This document also applies the secure transport protocol (i.e., Secure WebSocket for C-plane, SRTP and SCTP for U-plane). Then, the security of the communication channels is regarded as guranteed.

Regarding c), as an operator provided/assisted RTC service, trustable subscriber identification and verification are required to prevent unauthorized use of service and spoofing in the operator's network since a user self-claimed RTC user identity is untrusted. In addition to this, the originating RTC user identity is intended to be used not only for display at the terminating party but also for authorization at the opposite RTC network in the collaboration scenario 4, a solution to deliver the trusted user identity between two different RTC network is required.

Then, this key issue addresses the solution to deliver the trusted RTC user identity in the RTC networks considering the collaboration scenario 4 for the aspect of c).

## 5.12 Key Issue #11: Related groups considerations

This key issue studies the potential impacts on the specifications of other WGs in 3GPP and organizations including IETF and W3C.

# 6 Solutions

## 6.1 General

This clause describes the solutions for key issues in clause 5. Each solution corresponds to a key issue with the same number (e.g., Solution #1 corresponds to Key Issue #1). In subsequent clauses for each solution, solution evaluations are described per solution.

## 6.2 Solution #1: Enhancements on RTC architecture

### 6.2.1 Solution description

This solution addresses Key Issue #1.

This clause identifies the enhancements on RTC architecture considering what functionalities, functional entities and reference points are needed for WebRTC-based immersibe RTC services in collaboraion scenario 4. This includes:

1) Possible enhancements on functional entities and RTC architecture based on WebRTC view point (see clause 6.2.2);

2) Interaction between fuctional entities in the enhanced RTC architecture and 5GC (see clause 6.2.3);

3) Media connnection model (see clause 6.2.4);

4) IP addressing (see clause 6.2.5);

5) Alignment and gap analysis between the enhanced RTC architecture and the current RTC architecture (see clause 6.2.6); and

6) Enhanced RTC architecture for collaboration scenario 4.

As a conclusion of 1) to 6), the derivative RTC architecture and enhancements on 3GPP TS 26.506 [12] are proposed as a solution for Key Issue #1 in clause 6.2.8.

## 6.2.2 Possible enhancements on functional entities and RTC architecture based on WebRTC viewpoint

### 6.2.2.1 Overview

Figure 6.2.2.1-1 depicts a possible enhanced RTC architecture based on the WebRTC viewpoint. It contains the functional entities described in clause 6.2.2.2 and reference points described in clause 6.2.2.3. The names of functional entities and reference points described here are only for discussion of this solution and the alignment of these names with 3GPP TS 26.506 [12] are considered in clause 6.2.7.

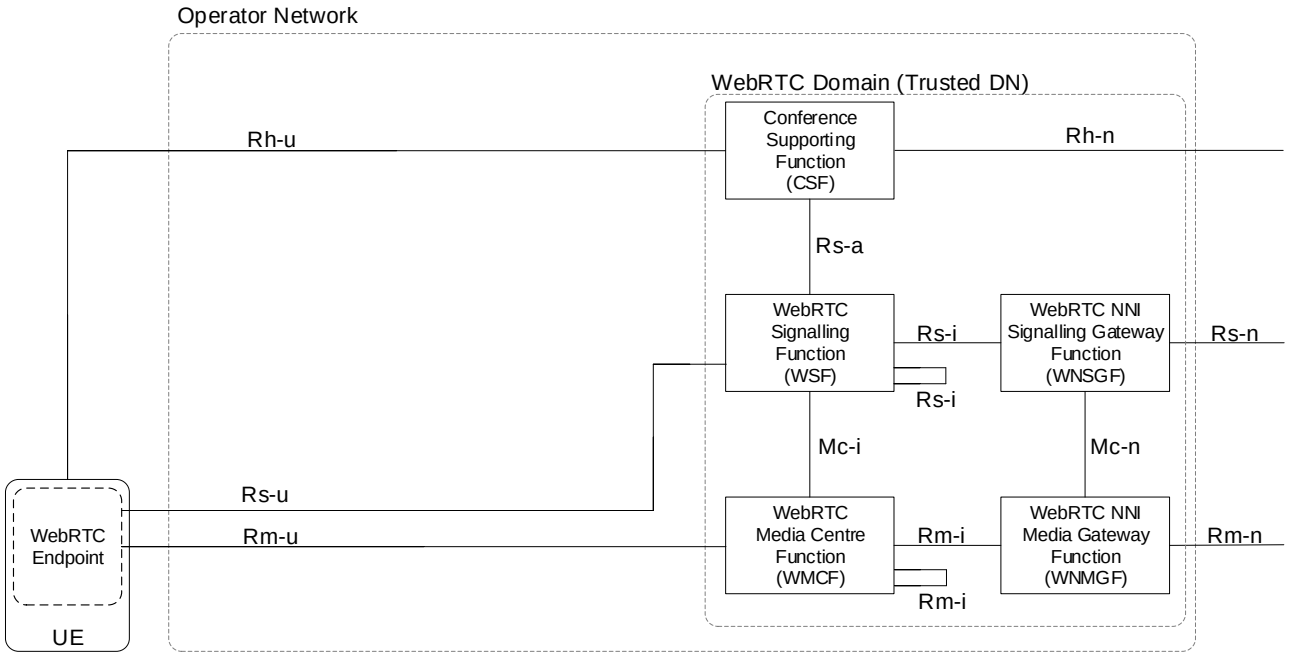


**Figure 6.2.2.1-1: Possible RTC architecture from WebRTC viewpoint**

WebRTC Signalling Function (WSF) and Conference Supporting Function (CSF) could co-locate in a physical node. WebRTC NNI Signalling Gateway Function (WNSGF) and WebRTC NNI Media Gateway Function (WNMGF) are optional when gateway functions are not needed at the network boundary.

### 6.2.2.2 Functional entities for WebRTC

#### 6.2.2.2.1 General

This clause enumerates functional entities in terms of 1) WebRTC-related standardized specifications, 2) WebRTC implementations, and 3) providing inter-operator services.

1) Functional entities that are essential for this document and those already defined in IETF RFCs or 3GPP specifications concerning WebRTC (see clause 6.2.2.2.2).

2) Functional entities that are not directly defined in WebRTC-related specifications of IETF RFCs or 3GPP specifications but considered to be widely implemented for realizing WebRTC services; they are essential for this document (see clause 6.2.2.2.3).

3) Functional entities that may be specifically required for inter-operator or third-party collaboration services if modification of signalling and termination of media at network boundaries are needed (see clause 6.2.2.2.4).

#### 6.2.2.2.2 Functional entities defined in WebRTC-related standardized specifications

##### 6.2.2.2.2.1 UE (User Equipment)

###### 6.2.2.2.2.1.1 General

User Equipment (UE) contains a user agent function which is equivalent to "WebRTC Endpoint" as described below. For the purposes of the present document, the following terms and definitions given in IETF RFC 8825 [44] are applied:

**WebRTC Endpoint**

- Either a WebRTC browser or a WebRTC non-browser. It conforms to the protocol specification.

**WebRTC Browser (also called a "WebRTC User Agent" or "WebRTC UA")**

- Something that conforms to both the protocol specification and the JavaScript API specification (W3C WebRTC 1.0 [65]).

NOTE: WebRTC browser is also called a "web app" in this document.

**WebRTC Non-Browser**

- Something that conforms to the protocol specification but does not claim to implement the JavaScript API. This can also be called a "WebRTC device" or "WebRTC native application".

Both "WebRTC Browser" type endpoint and "WebRTC Non-Browser" type endpoint are supported on the enhanced RTC architecture proposed in this document, as same as the current RTC architecture defined in 3GPP TS 26.506 [12]).

###### 6.2.2.2.2.1.2 Considerations specific to WebRTC endpoint types

There are two types of WebRTC Endpoint; one is "WebRTC Browser" type, and the other is "WebRTC Non-Browser" type. This clause shows possible functional model for each type of endpoints on enhanced RTC architecture for identifying the specific issues related to the WebRTC endpoint types. If the RTC application provider connects its server (e.g., media server, content server) to a WSF in an operator network without providing WSF functionality (i.e., connect to the operator's WebRTC DN via UNI not NNI), the server is treated as UE (WebRTC endpoint) for the operator's network.

Regarding the "WebRTC Browser" type WebRTC endpoint, a JavaScript application runs on a web browser that has capabilities of JavaScript APIs including WebRTC APIs defined by W3C (see Figure 6.2.2.2.2.1.2-1). According to the concept of WebRTC described in IETF RFC 8829 [47], the procedures and protocols stated in this document are expected to be fully writable only with JavaScript.

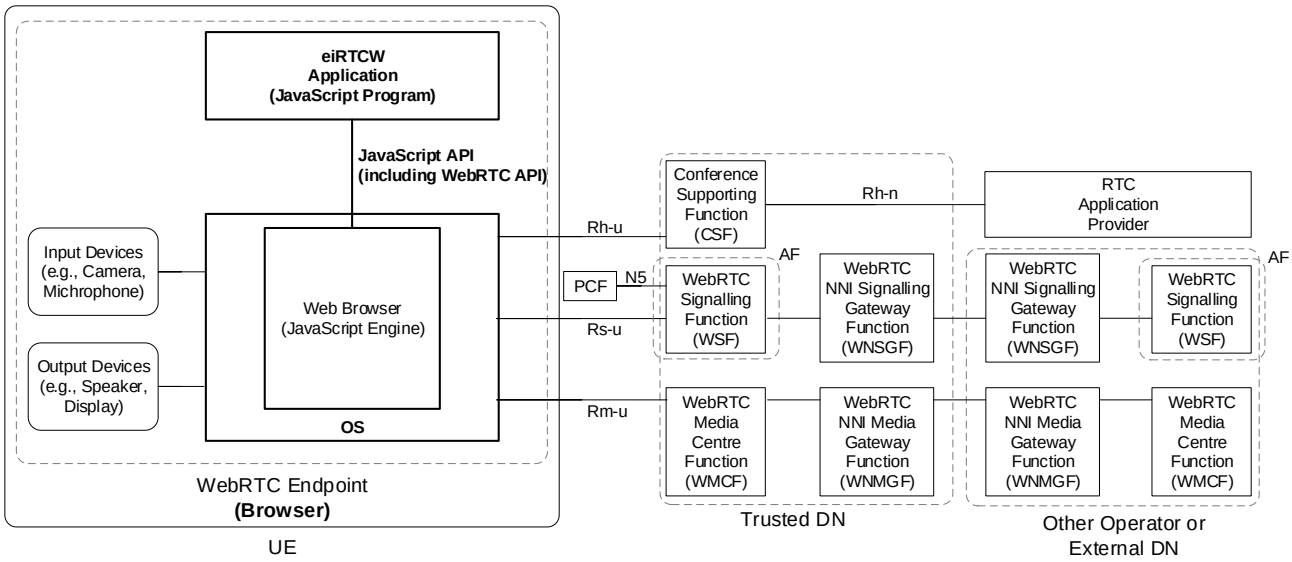


**Figure 6.2.2.2.2.1.2-1: "WebRTC Browser" type endpoint**

However, in the current situation, most of the OSS (e.g., android, iOS) and the web browsers (e.g., chrome, firefox) do not support/provide the enablers (provided by RTC MSH) for immersive RTC as JavaScript API. Therefore, to provide functionalities for realizing immersive RTC on "WebRTC Browser" type WebRTC endpoint, the mechanisms other than RTC MSH need to be supported. In order to support "WebRTC Browser" type endpoint, the protocols and procedures in this document can be implemented without RTC MSH.

Regarding the "WebRTC Non-Browser" type WebRTC endpoint, an application written in a programming language specific to the UE platform runs on UE using libraries and/or system call handlers. (See Figure 6.2.2.2.2.1.2-2)

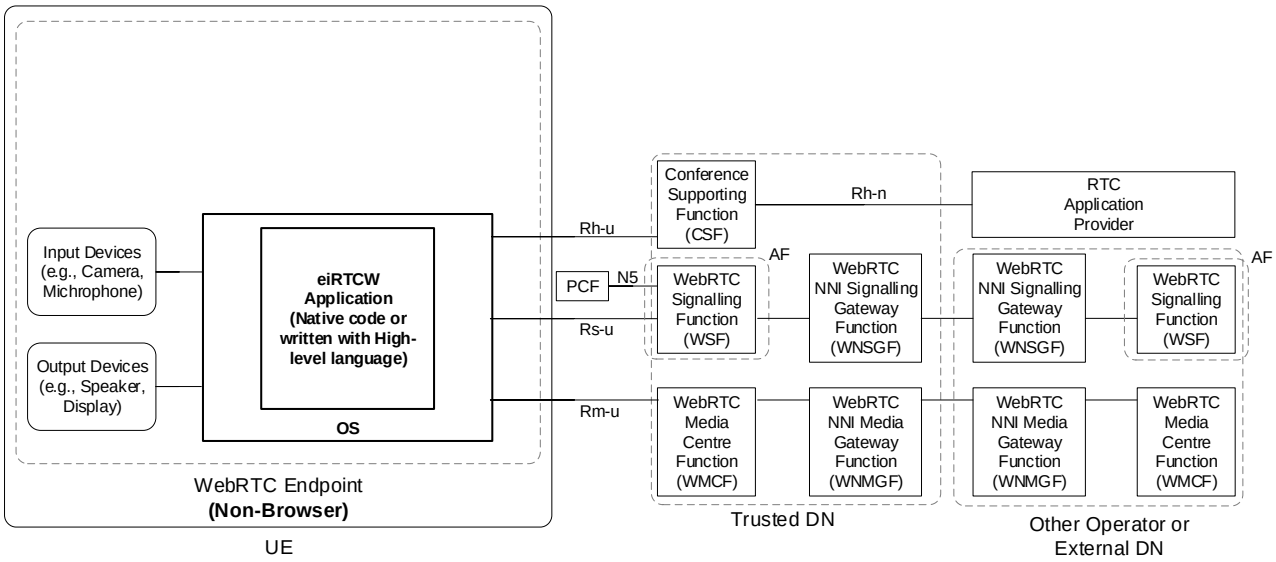


**Figure 6.2.2.2.2.1.2-2: "WebRTC Non-Browser" type endpoint**

NOTE: The programming language and programming APIs used to write applications depend on the UE platform. For example, Java and Android API (SDK) will be selected for Android platform UEs, Swift and its libraries will be selected for iOS platform UEs, and C++ and Win64 API will be selected for Windows platform UEs.

The application can be realized in a way other than JavaScript running on a web browser. The application can support the functions provided by RTC MSH since the application can be developed proprietary.

In this document, the solution which realizes the immersive RTC services without using RTC MSH is addressed to support both "WebRTC Browser" type endpoint and "WebRTC Non-Browser" type endpoint.

This document does not state details of the application's implementation; but the network interfaces, which is applicable for both "Browser" and "Non-Browser" type UEs are mainly described.

#### 6.2.2.2.2.2 WSF (WebRTC Signalling Function)

The WebRTC Signalling Function (WSF) is a functional entity that is responsible for WebRTC signalling mechanism including capability exchange and management of media sessions between UEs and the network. This functional entity is described as "Servers" or "Web Server" in clause 3 of IETF RFC 8825 [44]. Each operator or third-party in this document is assumed to have their own WSF(s) in their network.

WSF also provides the following functionalities:

- Interaction with WMCF for media session (real-time streaming and data channel) control.
- Interaction with CSF for collaboration with web applications/services.
- Interaction with 5GC, using Network Support function AF's (NS-AF) functionality.

### 6.2.2.2.3 Functional entities widely implemented for WebRTC

#### 6.2.2.2.3.1 WMCF (WebRTC Media Centre Function)

The WebRTC Media Centre Function (WMCF) is a functional entity that performs media processing. WMCF terminates media path (including audio/video stream and data channel) and performs media processing (e.g., mixing, selective forwarding, transcoding) which are required for immersive RTC applications. It may also perform decryption and encryption of media packets if DTLS, SRTP, or TLS is used for a transport layer. It also has the function of storing contents (including text or other static material as well as audio and video) and providing them to the UE. For media transport control, the WMCF interacts with WSF.

In the case that the WMCF acts as a simple media relay function, the WMCF simply relays media data packets and supports IP packet connectivity. When UE behaves as ICE Agents defined in IETF RFC 8445 [39] or IETF RFC 8838 [49], WMCF may be either STUN servers defined in IETF RFC 8489 [41] for connectivity check or TURN servers defined in IETF RFC 8656 [42] for relaying media data packets. This functional entity facilitates NAT traversal of UE and the connectivity between UE and other network functions.

This functional entity is generally implemented in WebRTC Multipoint Control Unit (MCU) or Selective Forwarding Unit (SFU).

#### 6.2.2.2.3.2 CSF (Conference Supporting Function)

The Conference Supporting Function (CSF) is a functional entity that provides various functionality to realize WebRTC based RTC services with operator assistance. The CSF is expected to provide the following functionalities:

- Conference session management, i.e., "CRUD" operation – create, read, update, delete of conference instances.

NOTE 1: CSF needs to support service control APIs. The details of these APIs are addressed in Key Issue #5 and Solution #5.

- Providing supplementary files (e.g., icon images of participants, and shared documents) via best-effort transport different from the channels for real-time media.
- Storage of user subscription data specific to MNO's WebRTC services.

NOTE 2: In this document, it is assumed that a single user (i.e., identity) and its subscription data (associated with the identity) are assigned, owned, and managed by both operator and RTC application provider independently. The two identities have a link with each other via some technique. User subscription data specific to RTC application provider's services are stored in their networks.

- Authorization endpoint and token endpoint of OAuth 2.0 described in IETF RFC 6749 [29] for establishing authentication linkage between MNO's ID and RTC application provider's ID.

NOTE 3: OAuth token will be used to C-Plane authentication at WSF and RTC application providers. STUN/TURN authentication with OAuth token is defined in IETF RFC 7635 [32]. Portal http(s) servers of WebRTC services provide this function in general implementations.

#### 6.2.2.2.4 Functional entities needed for inter-operator services

##### 6.2.2.2.4.1 WNSGF (WebRTC NNI Signalling Gateway Function)

The WebRTC NNI Signalling Gateway Function (WNSGF) is located at the boundary of the RTC networks where different operator's or third-party RTC network connects.

Each operator or third-party has its own WebRTC Signalling Functions (WSF) so that WSFs are connected to each other with border control functions such as security, policy management, charging, etc. WNSGF is inserted into "Signalling Path" in Figure 2 of IETF RFC 8825 [44] and responsible for border control functions and supports session establishment between disparate address realm's networks.

Also, WNSGF is able to support the functionality for interworking between WebRTC based signalling message and SIP message of IMS as a border control function.

NOTE: The details of interworking with IMS network are addressed in Key Issue #7 and Solution #7.

##### 6.2.2.2.4.2 WNMGF (WebRTC NNI Media Gateway Function)

The WebRTC NNI Media Gateway Function (WNMGF) is a media relay located at the boundary of the RTC networks where different operator's or third-party RTC network connects. WNMGF is responsible for the border control and transport of media data packets between different RTC networks. WNMGF may also transcode media data packets.

Also, WNMGF is able to support the functionality for interworking between WebRTC media and IMS media (e.g., transcoding of codec) as a border control function.

NOTE: The details of interworking with IMS network are addressed in Key Issue #7 and Solution #7.

### 6.2.2.3 Reference Points

The reference points shown in Figure 6.2.2.1-1 are enumerated as follows.

Reference points for signalling are called as "control plane" or "C-Plane" in this document. Reference points for media are similarly called as "user plane" or "U-Plane" in this document.

Reference Points for C-Plane:

**Rs-u:** Reference Point between a WSF and a UE.

**Rs-i:** Reference Point between a WSF and another WSF in the same network (DN) or between a WSF and a WNSGF.

**Rs-a:** Reference Point between a WSF and a CSF.

**Rs-n:** Reference Point between a WNSGF and another WNSGF in an external network.

Reference Points for U-Plane:

**Rm-u:** Reference Point between a WMCF and a UE.

**Rm-i:** Reference Point between a WMCF and another WMCF in the same network (DN) or between a WMCF and a WNMGF.

**Rm-n:** Reference Point between a WNMGF and another WNMGF in an external network.

Reference Points for signalling nodes to control media nodes:

**Mc-i:** Reference Point between a WSF and a WMCF.

**Mc-r:** Reference Point between a WNSGF and a WNMGF.

Other Reference Points:

**Rh-u:** Reference Point between a CSF and UE. This reference point is used for providing CSF functionalities (e.g., application usage assistance such as downloading an application) to UE.

**Rh-n:** Reference Point between a CSF and RTC application provider. This reference point is used for interaction between CSF and RTC application provider for media session set up related interaction.

Detailed protocol for each reference point will be discussed in the dedicated key issue and solution.

## 6.2.3 Interaction between functional entities in the enhanced RTC architecture and 5GC

### 6.2.3.1 Overview

This clause shows a solution for integrating the enhanced RTC architecture based on pure WebRTC with 5GC. In other words, this clause identifies the possible interaction between the functional entities of the enhanced RTC architecture and the functional entities of 5GC.

NOTE: "pure WebRTC" means the original WebRTC described in IETF work, which basically does not consider domain specific functions or features (e.g., mobile networks).

### 6.2.3.2 Mapping of functional entities for interaction with 5GC

#### 6.2.3.2.1 General

This clause identifies the mapping of functional entities shown in Figure 6.2.2.1-1 into 5GC functional entities defined in 3GPP TS 23.501 [4].

In this document, the mapping of WSF and AF, and the mapping of WNSGF and 5GC functional entities are considered. Other functional entities (i.e., CSF, WMCF, WNMGF) are not considered since these functional entities are not expected to interact with 5GC.

#### 6.2.3.2.2 WSF and AF

WSF is connected from UE and is expected to process the following:

1) authenticate a UE.

2) setup a WebRTC media session required by a UE, which may be in another network.

3) manage QoS for the media path of a WebRTC session.

Then it is expected that the WSF interacts with functional entities of 5GC and UE to perform 1) and 3) as the following:

1) WSF can retrieve the identity of a UE from 5GC, then authenticates and authorizes the UE.

3) WSF can request PCF to enable QoS control for the media path through e.g., N5, N32 (specified in 3GPP TS 23.501 [4]) or CAPIF reference points (specified in 3GPP TS 23.222 [2]).

NOTE: These processes are close to the processes of IMS functional entities such as P-CSCF and S-CSCF defined in 3GPP TS 23.228 [3]. The process of 1) is similarly performed by S-CSCF and UDM, and 3) is similarly performed by P-CSCF and PCF.

WSF can be mapped into "AF (Application Function)" of 5GC according to the definition of AF in 3GPP TS 23.501 [4] clause 5.2.10 due to the following reasons:

- WSF interacts with the 3GPP core network to provide services.

- The interaction between 5GC (e.g., PCF/UDM) and WSF is close to the interaction between 5GC and IMS entities (e.g., P-CSCF) that are AFs.

#### 6.2.3.2.3 WNSGF

##### 6.2.3.2.3.1 Overview

This clause identifies the mapping of WNSGF to a 5GC functional entity. There are a couple of possibilities currently identified. The following two 5GC functional entities can be mapped from WNSGF:

- NEF (see clause 6.2.3.2.3.2)
- SEPP (see clause 6.2.3.2.3.3)

As another possibility, it may be appropriate that WNSGF is mapped to a new functional entity (like Interconnection Border Control Function (IBCF) in IMS). The exact mapping of WNSGF is clarified in the alignment with 3GPP TS 26.506 [12] (clause 6.2.6 of this document).

##### 6.2.3.2.3.2 WNSGF and NEF

When WSF is mapped into an AF and if WNSGF is deployed as 5GC functional entity, WNSGF can be mapped into an NEF due to the following reasons:

- When WSF processes 2) of clause 6.2.3.2.2 and the media session relates to other operator's network, WSF (mapped to an AF) of operator-A is requested to interact with WNSGF on the boundary of operator-B to communicate with WSF (mapped into an AF) in operator-B due to operator-B's policy. In this model, the relationship between WSF (in operator-A) and WNSGF (in operator-B) is close to the relationship between AF and NEF described in clause 6.2.10 of 3GPP TS 23.501 [4].
- The major function of WNSGF is close to the former three functionalities described in 3GPP TS 23.501 [4] clause 6.2.5.0; WNSGF exposes WSF's WebRTC signalling capability and events. WNSGF interworks with WebRTC C-Plane signalling between Rs-i and Rs-n reference points in terms of security and translation of internal-external information.

When WNSGF is mapped into an NEF, the definition of the NEF may need to be modified as follows:

- Descriptions for the exposure of WSF's WebRTC signalling capability and the events by WNSGF are added in 3GPP TS 23.501 [4] clause 7.2.8.
- Descriptions for the event exposure details are added in 3GPP TS 23.502 [5] clause 4.15.3.
- Descriptions for the capability exposure details are added in 3GPP TS 23.502 [5] clause 5.2.6.

##### 6.2.3.2.3.3 WNSGF and SEPP

Security Edge Protection Proxy (SEPP) is defined in 3GPP TS 33.501 [18] and 3GPP TS 23.501 [4]. The SEPP is an entity sitting at the perimeter of the PLMN for protecting control plane messages, hiding network topology. The SEPP enforces inter-PLMN security on the N32 interface that is a reference point between a SEPP in one PLMN and a SEPP in another PLMN.

If WNSGF is deployed as 5GC functional entity, WNSGF is also located at the perimeter of the PLMN and its function is protecting control plane messages and hiding network topology. The function of WNSGF is close to that of SEPP.

The difference between WNSGF and SEPP is the type of located PLMN. WNSGF is located at the edge of inter-HPLMN. On the other hand, SEPP is expected to be used for N32 that lies between HPLMN and VPLMN.

##### 6.2.3.2.3.4 New functional entity

WNSGF is a border control function over C-Plane signalling path and located at the boundary of the networks where different operators or third-party network connects, as described in clause 6.2.2.2.4.1. Then, WNSGF is not expected to interact with 5GC functional entities and act as the gateway function for SBI.

In this document, the C-Plane signalling messages are expected to be exchanged via a DN over N6 interfaces and WNSGF is located at the DN. Therefore, WNSGF needs to be treated as a new border control function for C-Plane signalling path in WebRTC domain.

### 6.2.3.3 Possible architecture integrated with 5GC

The functional entities shown in Figure 6.2.2.1-1 can be connected to 5GC as described in Figure 6.2.3.3-1.

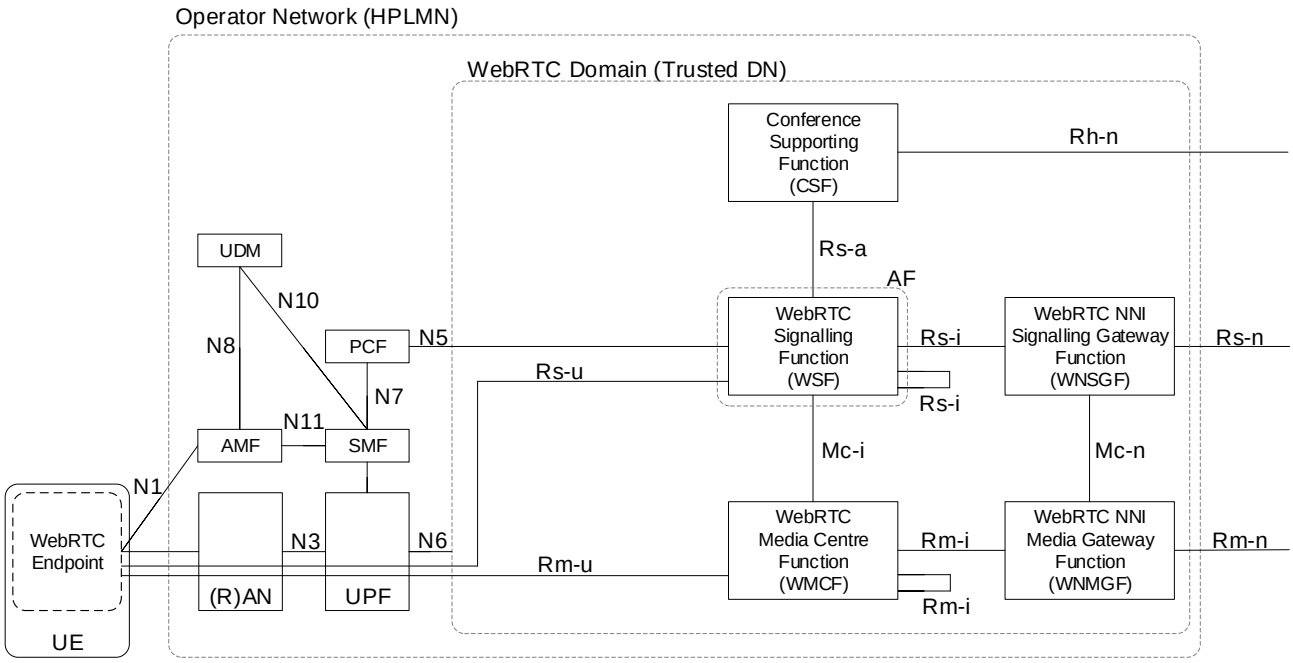


**Figure 6.2.3.3-1: Possible architecture integrated with 5GC**

WSF (with NS-AF functionality of RTC architecture) is mapped into an AF as the 5GC viewpoint.

WSF (with NS-AF functionality of RTC architecture) is interconnected with PCF via N5 interface. WSF manages QoS of real-time media packets and C-Plane signalling packets via N5 interface. WSF may interact with UDM to authenticate and to authorize the UE.

Both signalling packets and media packets between UE and the network are transmitted via N6 interface. Signalling packets (C-Plane packets) from UE are transmitted to WSF, and real-time media packets (U-Plane packets) from UE are transmitted to WMCF. C-Plane signals may travel to WNSGF via Rs-i, and may travel further to other operator's WNSGF via Rs-n. U-Plane signals may travel to WNMGF via Rm-i, and may travel further to other operator's WNMGF via Rm-n. (see Figure 6.2.3.3-2)

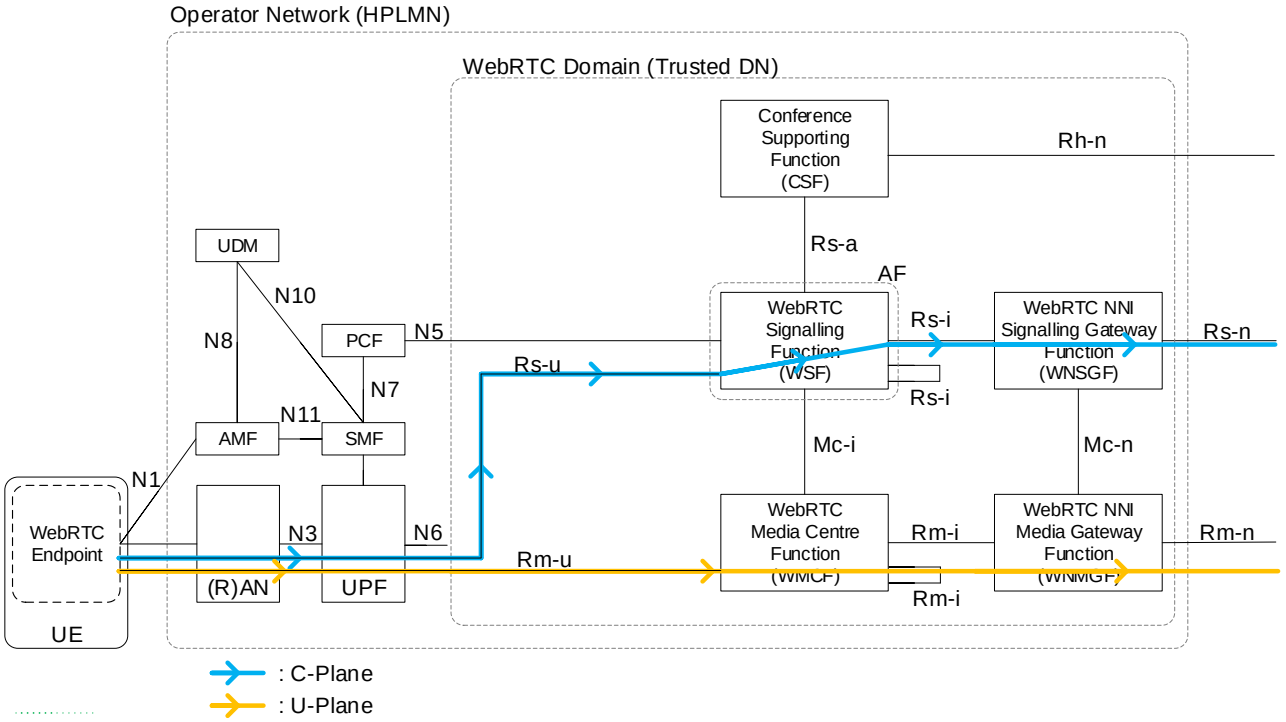


**Figure 6.2.3.3-2: Possible architecture from 5GC viewpoint with data flows of C/U-Planes**

### 6.2.3.4 Mapping to RTC collaboration scenarios

The following table shows the mapping of functional entities in this document into the collaboration scenarios described in 3GPP TS 26.506 [12]. Each box shows the condition (required or not) for MNO. The targets of this document are collaboration scenarios 3 and 4.

**Table 6.2.3.4-1: Functional entities required for each collaboration scenarios**

| Functional Entity | Collaboration Scenario 3 | | Collaboration Scenario 4 |
|---|---|---|---|
| | 3A / Service Provider provides WebRTC services and MNO assists the services | 3B / MNO provides WebRTC services only in the MNO's network | MNO's WebRTC service interconnects with other MNO's or Service Provider's service |
| WSF | Required | Required | Required |
| WMCF | Required | Required | Required |
| CSF | Required | Required | Required |
| WNSGF | N/A (NOTE) | N/A | Required |
| WNMGF | N/A (NOTE) | N/A | Required |
| NOTE: Scenario 3A in this table assumes service provider's WebRTC functions communicate with WSF and WMCF via UNI-like interface, i.e., WSF and WMCF work as a gateway by themselves. Further Operator-Assistance models may be introduced. | | | |

## 6.2.4 Media connection model

### 6.2.4.1 General

In the original WebRTC design, the communication between UEs is thought to be peer-to-peer (P2P). In most of the existing WebRTC implementations, however, the media connection is not P2P. An intermediate server (or servers) between UEs is used. In the multi-party call, the intermediate server which performs media processing is helpful for a UE because, for a UE, decoding all media from other UEs is a heavy load. Direct full-mesh connections among multiple UEs consumes a lot of network resources. Additionally, such an intermediate server is useful even for a one-to-one communication for offloading immersive media processing which needs more computation power than conventional media. This leads to the discussion about split rendering.

This document mainly focuses on the media connection model with intermediate servers.

P2P connection has some benefit for one-to-one communication (i.e., no need for an intermediate server and less server-relayed delay). For that reason, P2P connection is also considered for some special cases.

### 6.2.4.2 Target use cases from network view

Based on the high-level network model and target interfaces described in clause 4.2 and the enhanced RTC architecture described in clause 6.2.2, a signalling protocol which is called as "RESPECT" in this document supports the following use cases of media session set up from network view.

<Media session set up with RTC resource served in the operator network via UNI>

a. UE → RTC Resource (served by the same operator) ← UE

b. UE → RTC Resource (served by the same operator) ← UE (CP)

<Media session set up with RTC resource via NNI>

c. UE → RTC Resource (served by the other operator)

d UE → RTC Resource (served by an SP)

e. UE (served by the other operator) → RTC Resource ← UE (CP)

f. UE → Transit network (served by the other operator) → RTC Resource (served by an SP)

<Media session set up between UEs>

g. UE → UE (served by the same operator) without media gateway

h. UE → UE (served by the other operator) without media gateway

i. UE → UE (CP) without media gateway

j. UE (served by the other operator) → UE (CP) without media gateway

The overviews of these use cases are described below.

NOTE 1: Content provider connected via UNI is treated as UE.

NOTE 2: CSF is not shown in the figures for simplicity.

a. **UE → RTC Resource (served by operator) ← UE**

- UE establishes a media session destined for an RTC ID resource (e.g., immersive conference room) served by the same operator. Figure 6.2.4.2-1 shows an example that UE_A and UE_B establish media sessions destined for the same RTC ID resource (e.g., immersive conference room) to communicate with each other.

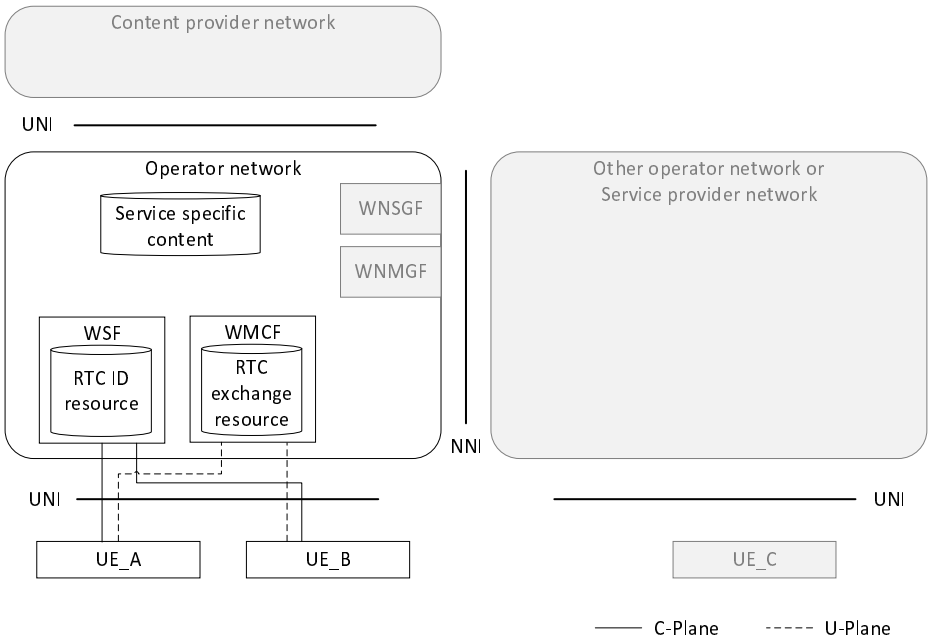


**Figure 6.2.4.2-1: UE → RTC Resource (served by the same operator) ← UE**

b. **UE → RTC Resource (served by operator) ← UE (CP)**

- UE_A and UE (CP) establish a media session destined for a RTC ID resource served by an operator network which UE_A and UE (CP) connected to.

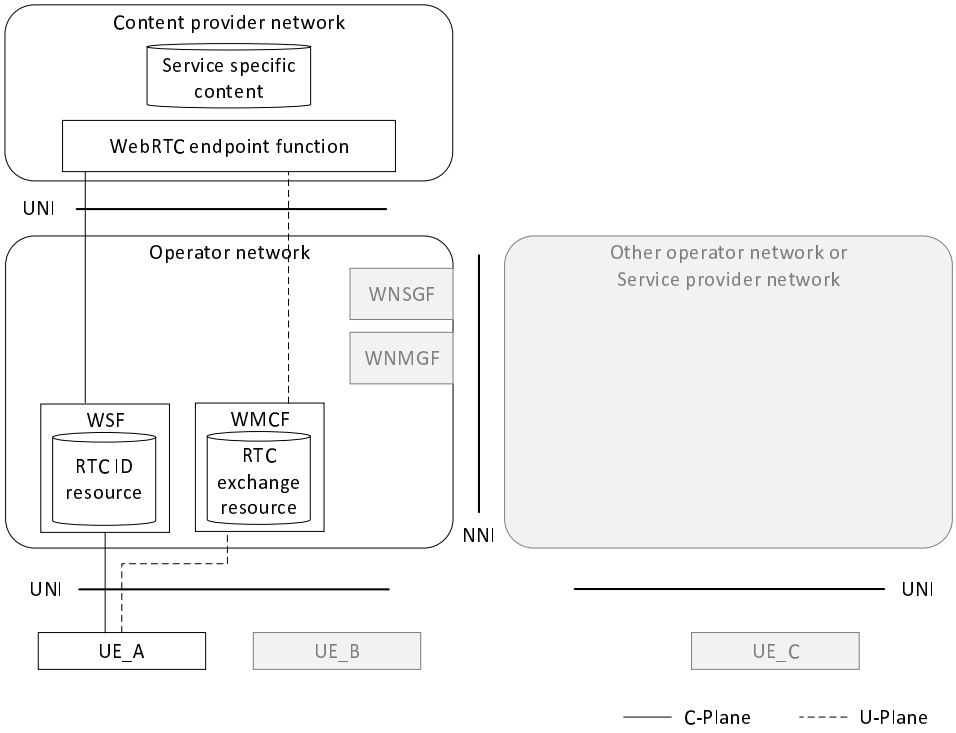


**Figure 6.2.4.2-2: UE → RTC Resource (served by the same operator) ← UE (CP)**

c. **UE → RTC Resource (served by the other operator)**

- UE_A establishes a media session destined for a RTC ID resource (e.g., Immersive conference room) served by the operator that different from the network which the UE_A is connected to. In this scenario, the C-Plane signalling messages and media/data are sent over the NNI. Other UEs can connect to the RTC ID resource as same as use case a.

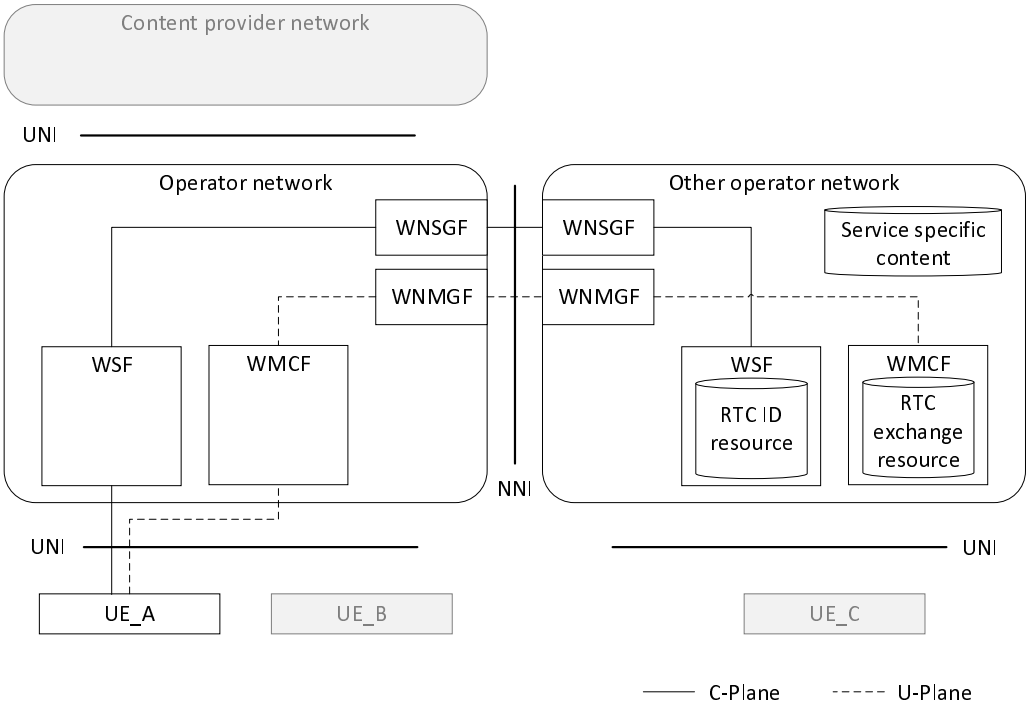


**Figure 6.2.4.2-3: UE → RTC Resource (served by the other operator)**

d. **UE → RTC Resource (served by an SP)**

- UE_A establishes a media session destined for a RTC ID resource (e.g., Immersive conference room) served by an SP. In this scenario, the C-Plane signalling messages and media/data are sent over the NNI.

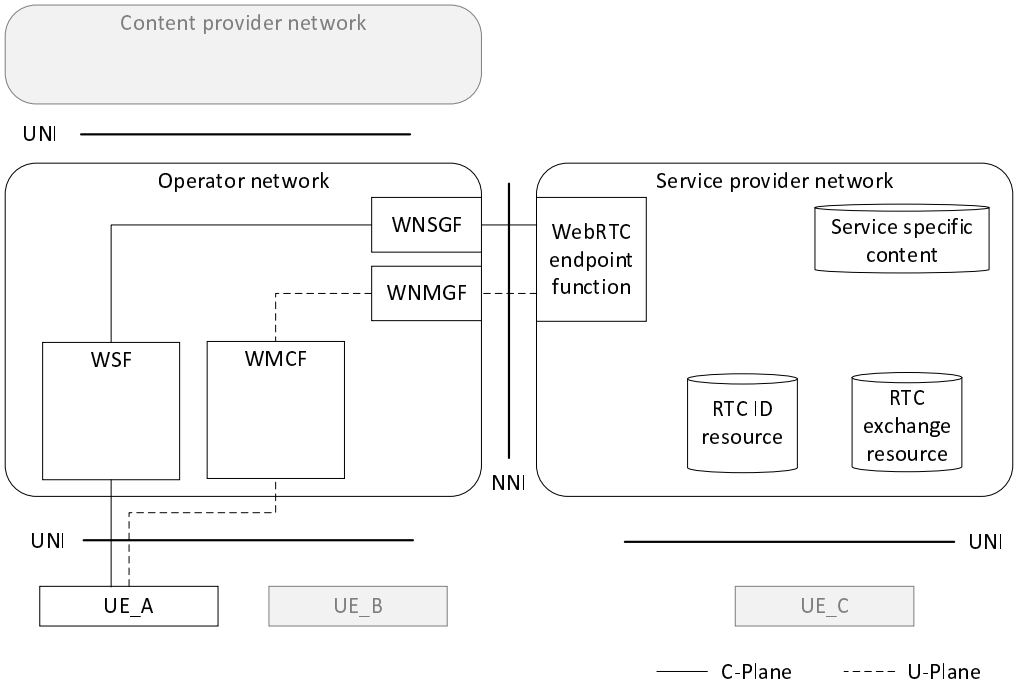


**Figure 6.2.4.2-4: UE → RTC Resource (served by an SP)**

e. **UE (served by other operator) → RTC Resource ← UE (CP)**

- UE_C in the other operator network than an operator network serving a RTC ID resource (e.g., Immersive conference room) and UE (CP) in the operator network serving the RTC ID resource establish a media session destined for the RTC ID resource. In this scenario, the C-Plane signalling messages and media/media from UE are sent over the NNI.

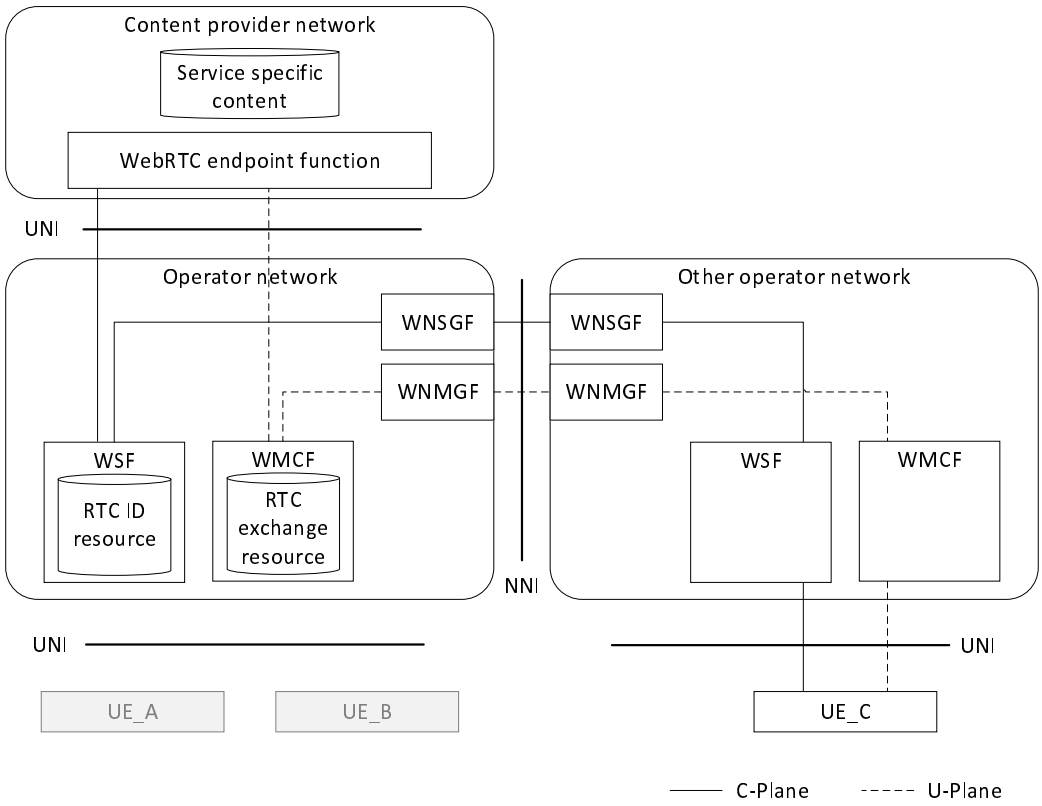


**Figure 6.2.4.2-5: UE (served by the other operator) → RTC Resource ← UE (CP)**

f. **UE → Transit NW (other operator) → RTC Resource (served by an SP)**

- UE_A establishes a media session with a RTC ID resource (e.g., Immersive conference room) served by an SP via transit network served by the another operator. In this scenario, the C-Plane signalling messages and media/data are sent over the two different NNIs.

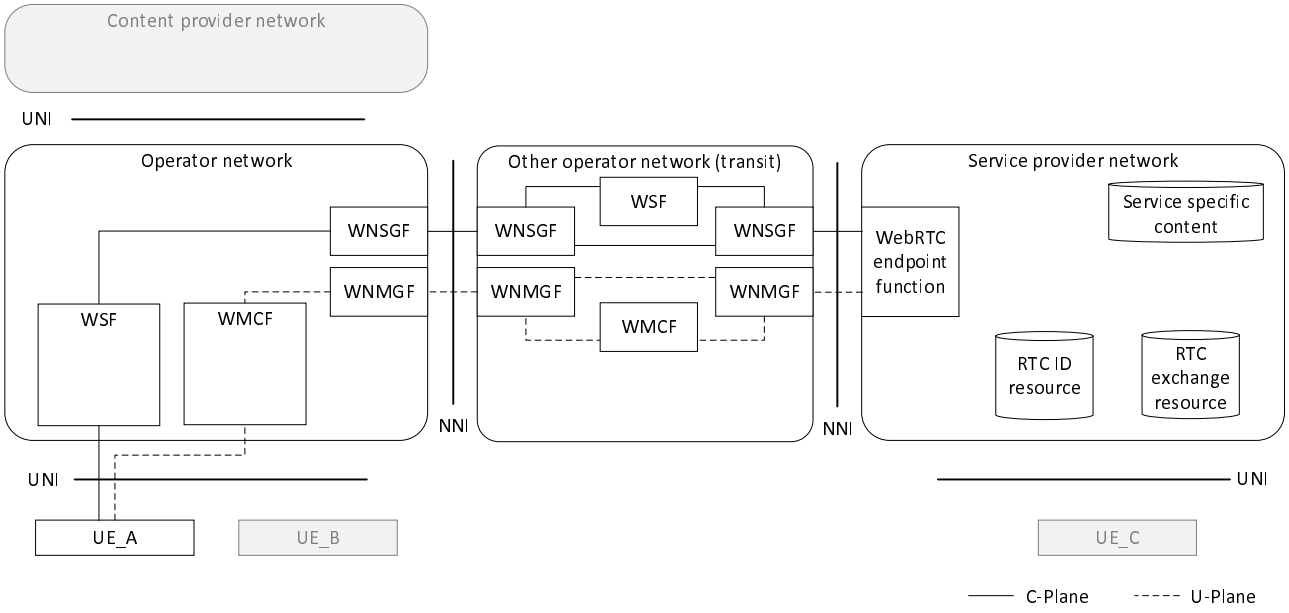


**Figure 6.2.4.2-6: UE → Transit network (served by the other operator) → RTC Resource (served by an SP)**

g. **UE - UE (served by the same operator) without WMCF**

- UE_A establishes a media session (e.g., voice chat) with UE_B served by the same operator, without using WMCF.

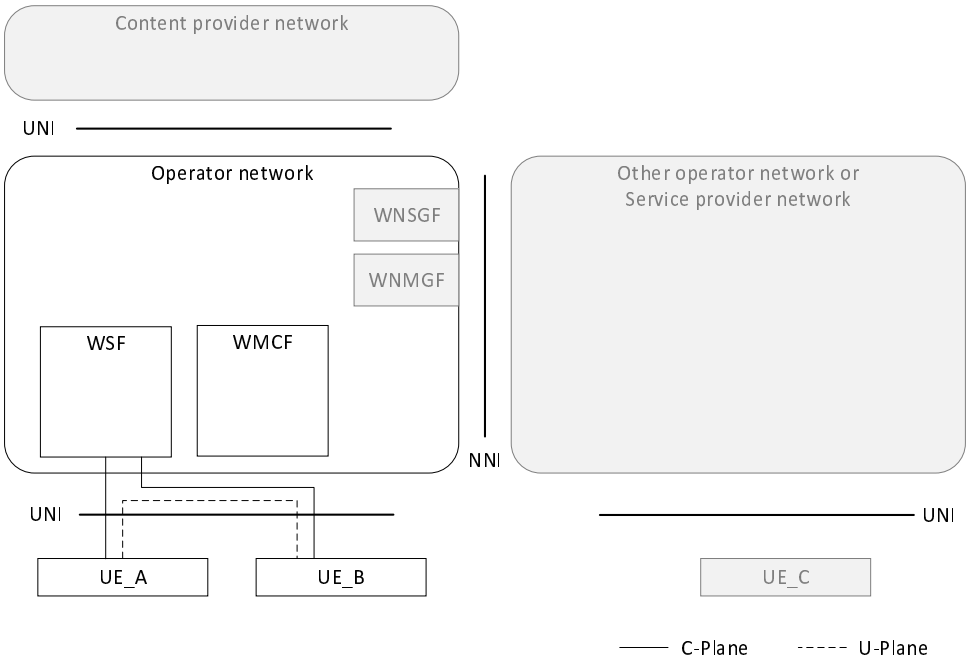


**Figure 6.2.4.2-7: UE - UE (served by the same operator) without media gateway**

h. **UE → UE (served by the other operator) without WMCF**

- UE_A establishes a media session (e.g., voice chat) with UE_C served by the different operator, without using WMCF. In this scenario, the C-Plane signalling messages and media/data are sent over the NNI.

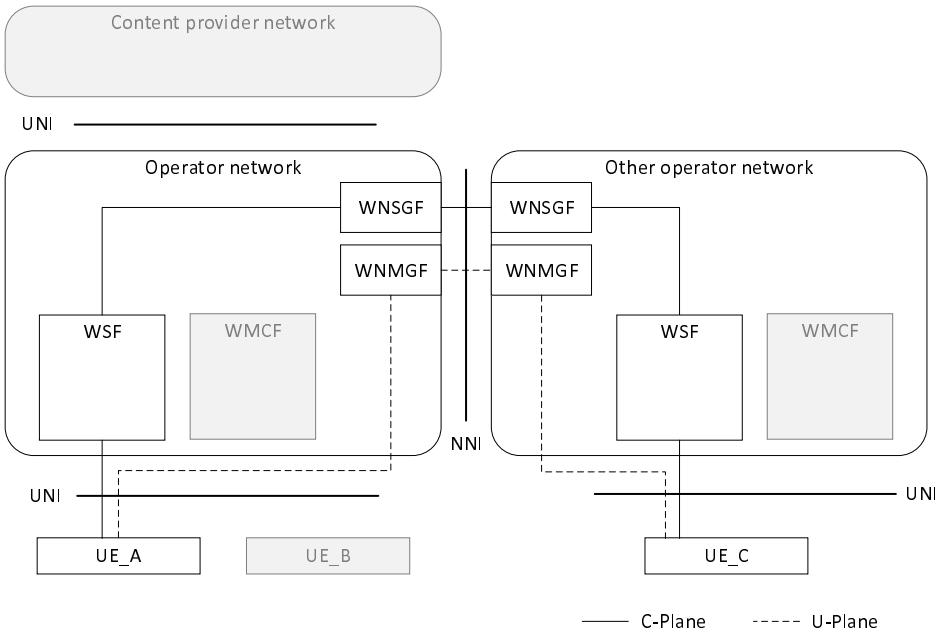


**Figure 6.2.4.2-8: UE → UE (served by the other operator) without media gateway**

i. **UE → UE (CP) without WMCF**

- UE_A establishes a media session with a UE (CP) which is connected to the same operator, without using WMCF.

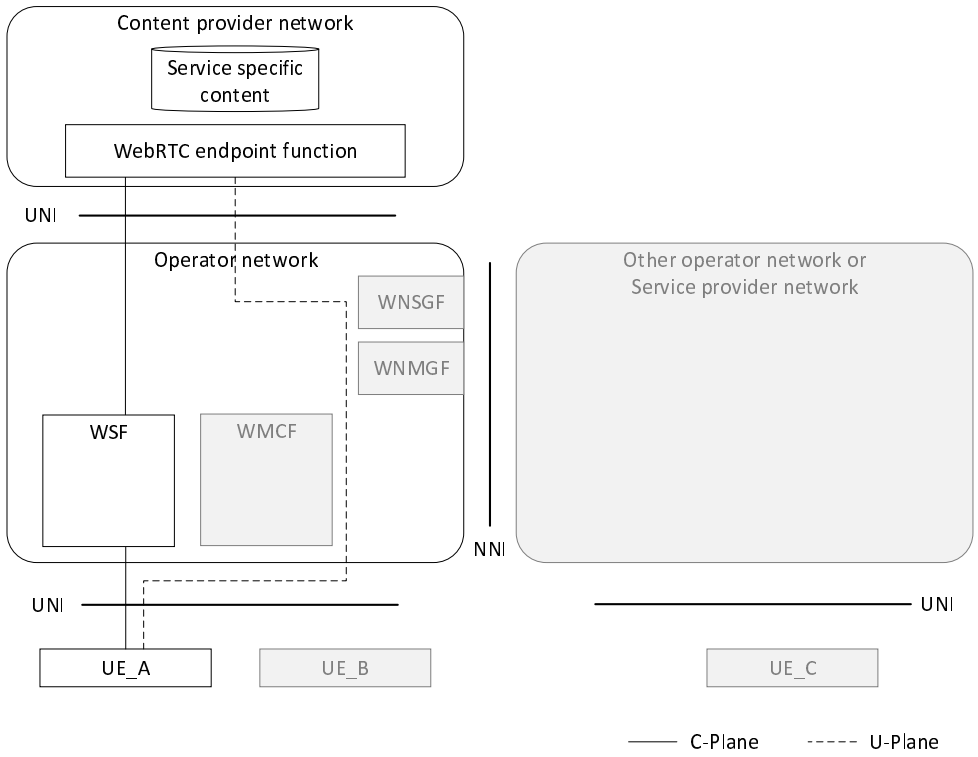


**Figure 6.2.4.2-9: UE → UE (CP) without media gateway**

j. **UE (connected to the other operator) → UE (CP) without WMCF**

- UE_C establishes a media session with a UE (CP) which is connected to the different operator, without using WMCF. In this scenario, the C-Plane signalling messages and media/data are sent over the NNI.

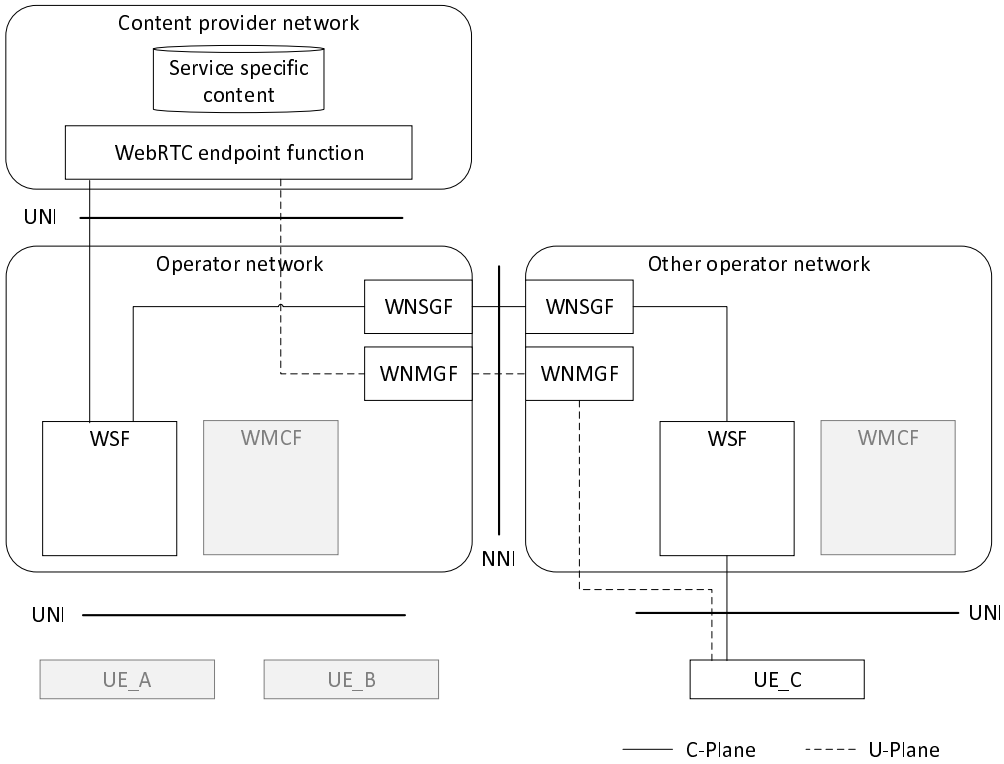


**Figure 6.2.4.2-10: UE (connected to the other operator) → UE (CP) without media gateway**

### 6.2.4.3 QoS-enabled end-to-end path

In the collaboration scenario where the WebRTC functions in an operator's network assists an external provider (i.e., service provider, content provider, or another operator), setting up a QoS-enabled media path across different networks needs to be considered.

The media path from a UE to the external provider is roughly divided into the following four sections:

Section 1) Between a UE and the UPF (operator's CN section)

Section 2) Between the UPF and the operator's network edge (operator's DN section)

Section 3) Between the operator's network edge and the external provider network edge

Section 4) A network in the external provider

Section 4) is a matter of an external provider and out of scope of this document.

Regarding Section 1), this section includes the operator's core network. In this section, QoS is controlled by the PCF.

Regarding Section 2), operator's DN may have sufficient bandwidth or the other QoS mechanism may be adopted.

Regarding Section 3), this section's QoS control needs a bandwidth guaranteed path (i.e., a dedicated line). On the enhanced RTC architecture described in clause 6.2.2, when the media path is connected to a RTC resource in other operator's network or service provider's network, the media packets to be prioritized are transmitted to WNMGF placed in the operator's network and the WNMGF relays the media to the main media server in the other operator's network or service provider's network via guaranteed path as shown in Figure 6.2.4.3-1 (red-line). If the media path is connected to a WebRTC endpoint function in a content provider's network via WSF and WMCF (which work as a gateway) in the operator's network, this section is treated as UNI, as shown in Figure 6.2.4.3-1 (blue-line).

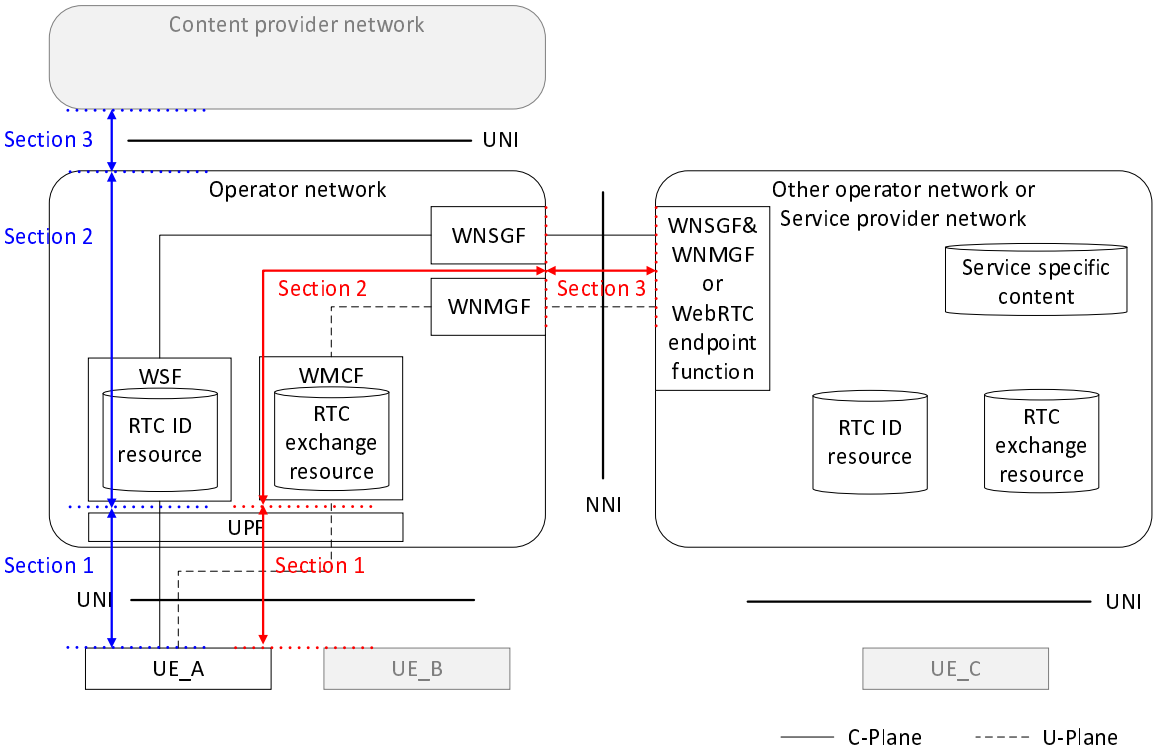


**Figure 6.2.4.3-1: Sections of end-to-end media path**

## 6.2.5 IP Addressing

### 6.2.5.1 Overview

IP addressing for UE has some options: assigning IPv4 address only, IPv6 address only, or both.

In the operator deployment, the number of available IPv4 addresses would be insufficient for its subscribers. Generally, operators use IPv4 private address (and ISP shared address defined in IETF RFC 6598 [28]) with network address translation (NAT).

In clause 6.2.5, appropriate IP addressing is identified, discussing NAT-traversal in the WebRTC user plane and network verified ID retrieval.

### 6.2.5.2 NAT

#### 6.2.5.2.1 Overview

NAT, including port translation as NAPT (Network Address and Port Translation), is a method of mapping an IP address space into another, which is mainly used to translate a private IP address into a global IP address, and vice versa, for communicating with external networks.

Generally, UE can be assigned with an IP address through a PDU session in operator networks. When an IPv4 address is allocated, as mentioned in clause 6.2.5.1, a private IP address or an ISP shared address is used. On the contrary, when an IPv6 address is allocated, a global unicast address is assigned.

NAT is essential for carrier-grade network deployment. Subscribers can be much more than usually available IPv4 global address space, and they are treated by using IPv4 private address and NAT. The same private address can be reused in each different domain behind NAT. Although NAT deployments have a wide variety, NAT is generally installed in a DN (data network) and often put in the middle between the UPF and other functional entities (see Figure 6.2.5.2-1).

On the other hand, IPv6 global unicast addresses basically do not require NAT, except for special security reasons or some transition method between IPv6 and IPv4 domains.

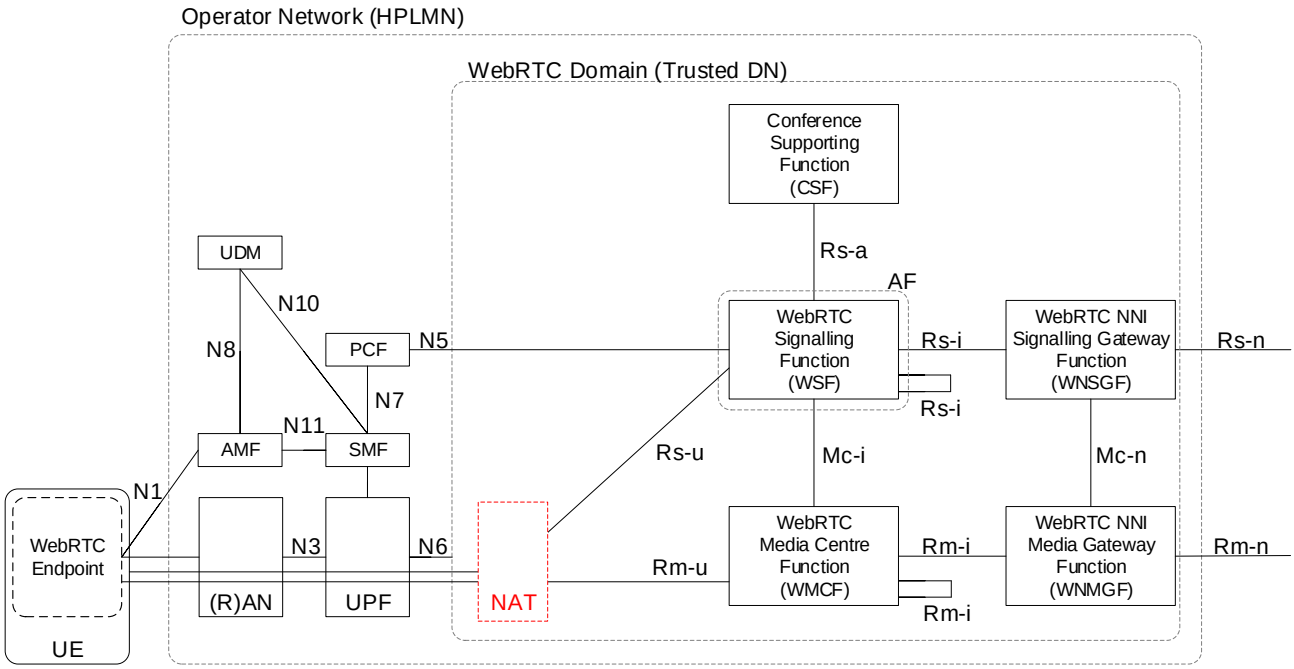


**Figure 6.2.5.2-1: Possible NAT location**

#### 6.2.5.2.2 NAT Variation

NAT is classified into some types by its address translation and packet filtering behavior.

The first version of STUN in IETF RFC 3489 [24] defines three types:

- Full Cone NAT,
- Restricted NAT (Restricted Cone NAT or Restricted Port Cone NAT), and

- Symmetric NAT.

Full cone NAT does not limit access to an internal UE from external network entities, which have not communicated with the internal UE. Any external entities can re-use the external IP address and port number mapped to a specific internal UE and can access to it (Figure 6.2.5.2.2-1). Full cone NAT is less restrictive than other NATs. Restricted NAT only permits external entities to access the internal UE if the NAT have received any packets from the internal UE directed to the external UE (Figure 6.2.5.2.2-2). Symmetric NAT uses a different pair of an external IP address and port, which are specific to each external entity and only the external entity can access to the internal UE through the IP address and port pair.

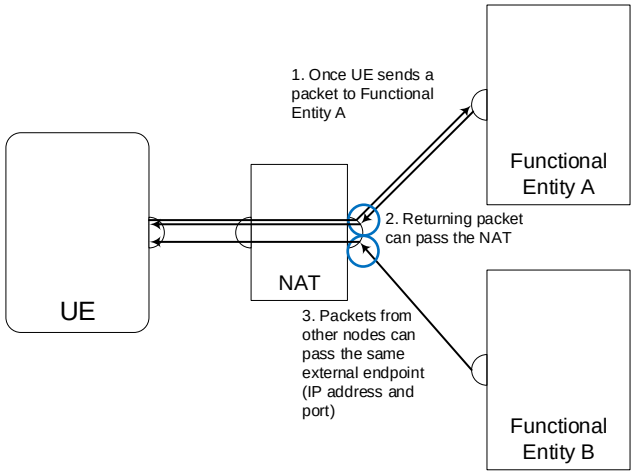


**Figure 6.2.5.2.2-1: Full Cone NAT behaviour**

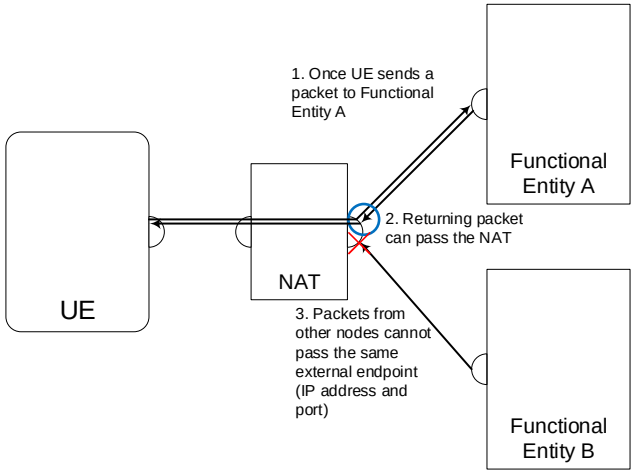


**Figure 6.2.5.2.2-2: Restricted or Symmetric NAT behaviour**

### 6.2.5.2.3 Existing NAT-traversal

#### 6.2.5.2.3.1 General

An effective NAT-traversal method is different depending on the NAT type described in clause 6.2.5.2.2.

In the original WebRTC design, STUN and TURN are listed, included as ICE, for major NAT-traversal methods. In addition, Hosted NAT Traversal (HNT, described in IETF RFC 7362 [31]) and its similar mechanism are frequently used in real implementations for conversational applications.

#### 6.2.5.2.3.2 STUN

STUN is the method for UE behind the NAT to discover its external IP address observed by external networks. This method supports P2P communications and only works for full-cone NAT.

This document excludes STUN because the main communication model is not P2P but with intermediate servers (as described in clause 6.2.5), and general NATs deployed in operator networks are not limited to full-cone type.

#### 6.2.5.2.3.3 TURN

TURN is the method for UE behind the NAT to communicate with external nodes via an intermediate server. TURN is a protocol for the session management and requires an intermediate server.

Generally, this method is regarded as the last resort for NAT-traversal for UDP-based conversational services. This method does not require the alignment with other control plane signalling, but is equipped as its own user plane connection management mechanism. This method needs additional message exchanges and has a protocol overhead.

The TURN server has its authentication mechanism for UEs and can be used for the purpose of traffic steering for an inter-operator communication scenario detailed in clause 4.2.4.3.

#### 6.2.5.2.3.4 HNT

HNT (Hosted NAT Traversal) is the mechanism that a session border controller (SBC) placed at the edge of networks intermediates the communication between UEs behind NAT.

The problem tackled by HNT is that a UE behind a NAT tries to set up a session with its private address and port number for media, which have no clue to the SBC for the real media which comes later.

Regarding the control plane signalling, the signalling part of the SBC modifies media-related information represented by the private IP address and port number set in the SDP offered by an originating node into a global IP address and a new port number. This modification enables a terminating node to target the accessible IP address and port pair provided by the SBC. In the signalling return path, the SBC also modifies the terminating node's IP address and port number set in the SDP answered by the terminating node into new ones, and forwards it to the originating node. This is to solicit the originating node to send media to the SBC. Once the SBC receives the first media packet from the originating node targeting at the solicitation, the SBC recognizes the real NAT-ed IP address and port pair of the originating node. The SBC captures that information and uses it for relaying packets from the terminating node to the originating node. This is called "latching".

This method is embedded in the control plane signalling and does not require extra message exchange. For that reason, it has no additional protocol overhead. It is a better feature than TURN in the same condition requiring an intermediate server.

Since this document focuses on the connection model with an intermediate server, the NAT issues can be argued differently. Let's assume that all communication services are provided by the intermediate server as a conference. UEs can just join the open channel provided by the server and receives media from the server. UEs can also send their media to the intermediate server and the server mixes the media and distributes to other UEs. In this model, the first join packet from a UE to the NAT and the NAT to the server creates an address mapping at the NAT. The server simply sends packets to the source address of the join packet from the UE.

This mechanism does not need the dedicated protocol and there is no additional protocol overhead for NAT-traversal by sending media to the specific IP address and port pair exposed by WMCF. That points are analogous to HNT (Figure 6.2.5.2.2.4-1).

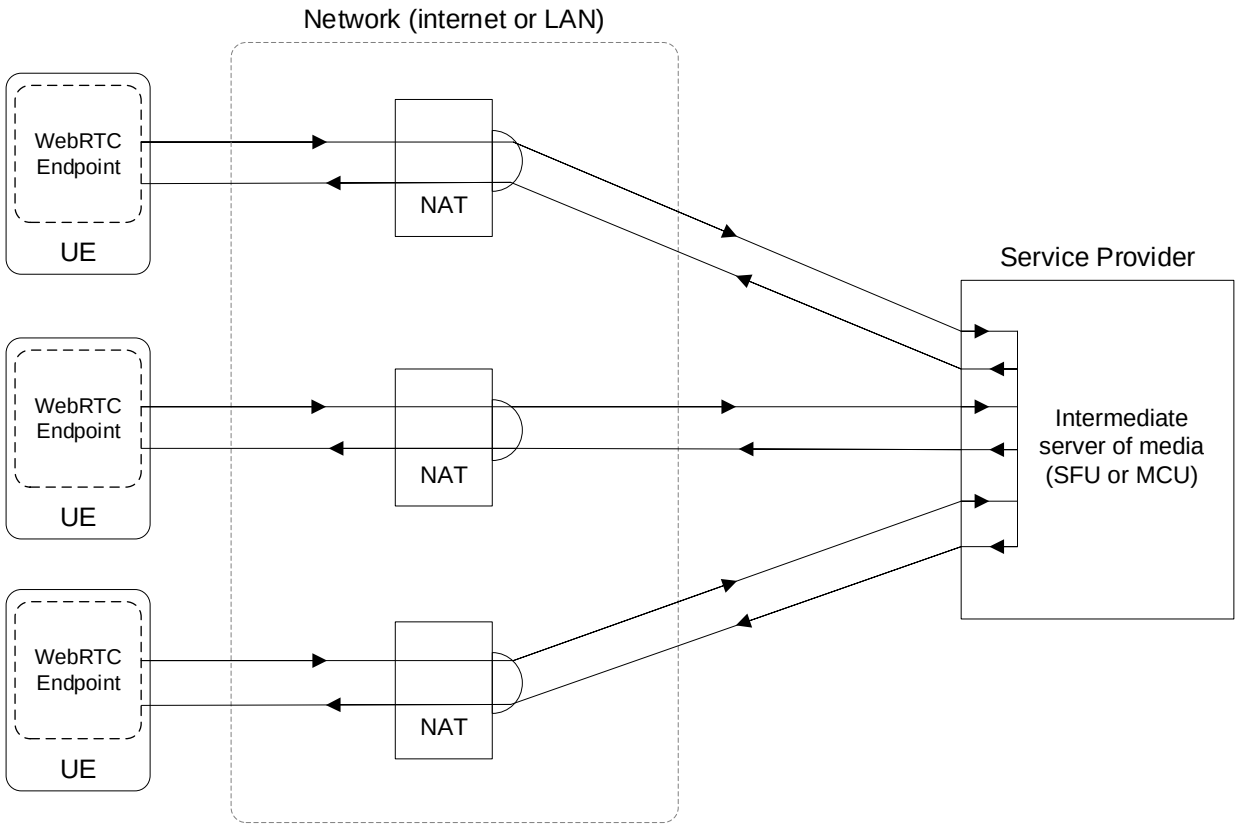


**Figure 6.2.5.2.2.4-1: HNT like NAT-traversal**

### 6.2.5.2.4 Conclusion of NAT handling

NAT-traversal problems have been discussed and several solutions have been proposed as described above. However, if equipment for NAT-traversal is not required, certainly less server resources would be needed.

In short, it is preferable that only IPv6 global unicast address be assigned to UE and no dedicated NAT-traversal equipment be used. Intermediate servers are used mainly for media processing and for the media relay when there is no direct IP reachability (e.g., across inter-operator connection).

## 6.2.5.3 IP Address and Trustable Subscriber Identifier

The operator uses subscription identifiers (e.g., GPSI (Generic Public Subscription Identifier) in 5GC) for managing its customer's service subscription and charging. In WebRTC support, the operator needs to check customer's service requests by checking against operator's subscriber database organized with the subscription identifier. An OTT-specific ID and password may be insufficient even in the collaboration scenario with external service providers because they cannot be securely linked with subscriber information in the viewpoint of the operator. The issue is how the MNO deduces (or retrieves) the trustable subscriber identifier from customer's requests, which are carried by IP packets.

Trustable subscriber identifiers in the MNO network are required for certain validity check, since a UE's self-claimed GPSI and source IP address are untrusted.

The EDGEAPP architecture specifies the method how the EAS function block retrieves the GPSI from terminal's source IP address. The AF regarded as an EAS can retrieve the GPSI bound to the UE by Eees_UEIdentifier API in EDGEAPP. This mechanism and its flow contain authentications conducted at the related network functions (i.e., EES and NEF), which enable the EAS to acquire the valid GPSI in the operator network as a trustable subscriber identifier.

Validity of the terminal's source IP address needs consideration. UE's self-claimed IP address, especially presented in an application level, is not trustable. The source IP address presented in an IP header can be relatively trustable when the IP packet is transmitted through a connection with some handshake procedures.

The IP address linkage with a subscriber identifier also has an issue when NAT is deployed. In release 18, the method with which the AF can identify the trustable subscriber identifier (e.g., GPSI) to invoke the 3GPP network service API

for the UE (Application client) remains to be investigated in eEDGEAPP. In VoLTE, this linkage with NAT can be achieved with the help of additional operator-specific information (e.g., PDN session related value). In the AF for WebRTC, it depends on which additional information element can be acquired by the AF. There is no clear answer for the ID linkage between the NAT-ed IP address and the subscriber identifier.

Contrarily, the UE IP address without translated by NAT can be linked with GPSI by Eees_UEIdentifier API (though detailed specification is needed).

In terms of ID linkage, using IPv6 global unicast address for UE is reasonable.

Using IPv4 private address will be studied further when NAT-ed ID linkage issue is solved.

#### 6.2.5.4 Conclusion of IP Addressing

In terms of the required server resources for NAT-traversal and unclear retrieval of the trustable subscriber identifier, using IPv6 global unicast address for UE is reasonable. NAT deployments have a wide variety of behaviors and cannot be treated straightforward (refer to clause 6.2.5.2). Using media relay servers that act as either TURN or HNT covers most cases with NAT-traversal. However, there are still issues using IPv4 private address with NAT, such as ID linkage (refer to clause 6.2.5.3). For the sake of simplicity and to concentrate on identifying signalling requirements, this document considers IPv6-only use.

Then the use of ICE Function and the enhancements of ICE function are excluded from the scope of this document.

NOTE 1: As specified in 3GPP TS 26.506 [12], the use of ICE Function is optional and is not restricted.

NOTE 2: IPv6-only use is acceptable for future services because IPv6 address allocation to UEs is now widely available among operators. Also, IPv6-only deployment leads to efficient system development and equipment utilization.

### 6.2.6 Alignment and gap analysis between the enhanced RTC architecture and the current RTC architecture

#### 6.2.6.1 General

This clause identifies the architectural and functional mapping between enhanced RTC architecture described in clause 6.2.2 of this document and the current RTC architecture defined in 3GPP TS 26.506 [12]. Figure 6.2.6.1-1 shows the RTC general architecture specified in 3GPP TS 26.506 [12].

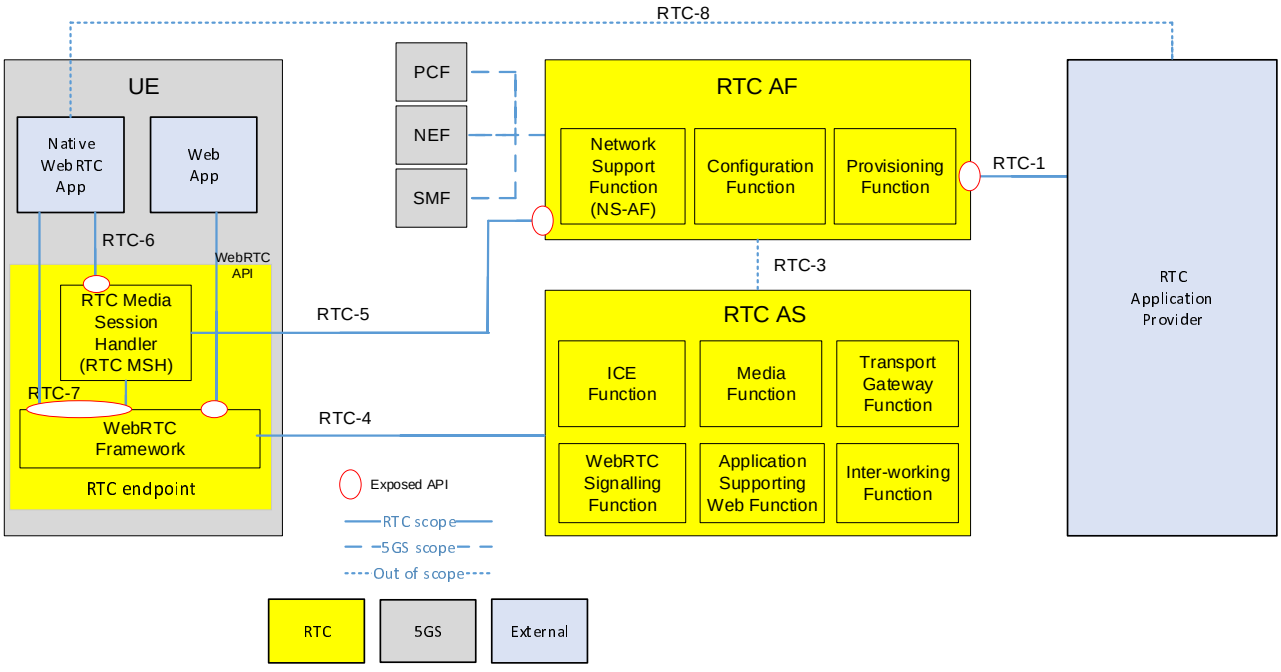


**Figure 6.2.6.1-1: RTC General Architecture**

#### 6.2.6.2 WebRTC endpoint and RTC endpoint on UE

WebRTC endpoint on the UE is expected to be mapped to RTC endpoint on the UE on the RTC architecture with the following consideration.

- An WebRTC endpoint includes signalling related aspects of applications on the UE, however, an RTC endpoint does not include applications on the UE. To support the signalling protocol for media session setup, the signalling related functionality of application needs to be included in the scope of the RTC endpoint. Note that application itself is not included in this scope.

#### 6.2.6.3 WSF and (RTC) WSF

WSF is expected to be mapped to WSF (integrated with NS-AF) on RTC architecture with the following considerations.

- WSF needs to support the functionality for interaction with Application Supporting Web Function (ASWF) for collaboration with web applications/services.
- WSF needs to support the functionality for interaction with 5GC, using network Support function (NS-AF) functionality.
- WSF needs to support the functionality for retrieval of the identity of a UE from 5GC, and authentication of the UE.

Regarding the retrieval of the identity of a UE from 5GC, as described in clause 6.2.5.4, the WSF is not able to retrieve the identity from 5GC in the case that the UE is assigned an IPv4 private address behind NAT in the current release. In that case, the authentication mechanism in commercial use such as SMS OTP (One Time Password) is possibly applicable for enhancement of authentication of the UE.

#### 6.2.6.4 WNSGF and Inter-working Function

Inter-working Function (IWF) is specified in 3GPP TS 26.506 [12] as an inter-working functionality to enable MNO-facilitated WebRTC sessions that involve endpoints across different MNOs (e.g., providing cross-network signalling functionality). This is the expected functionality for WNSGF, since WNSGF is a gateway function for signalling messages between MNOs. Then, WNSGF is expected to be mapped to IWF on RTC architecture.

No gap is found between WNSGF and IWF.

#### 6.2.6.5 CSF and Application Supporting Web Function

CSF is expected to be mapped to ASWF on RTC architecture. Also, the ASWF is expected to support the additional functionalities described in clause 6.2.2.2.3.2 in addition to the current functionality defined in 3GPP TS 26.506 [12].

#### 6.2.6.6 WMCF and Media Function

WMCF is expected to be mapped to Media Function (MF) on RTC architecture. Also, the MF is expected to support the following functionalities.

- Performing decryption and encryption of media packets if DTLS, SRTP, or TLS is used for a transport layer.
- Storing contents (including text or other static material as well as audio and video) and providing them to the UE.

#### 6.2.6.7 WNMGF and Transport Gateway Function

WNMGF is expected to be mapped to Transport Gateway Function (TGF) on RTC architecture.

No gap is found between WNMGF and TGF.

### 6.2.7 Enhanced RTC Architecture for collaboration scenario 4

This clause identifies the enhanced architecture for collaboration scenario 4 specified in 3GPP TS 26.506 [12] based on the consideration in above clauses. Figure 6.2.7-1 shows the derivative RTC architecture for collaboration scenario 4.

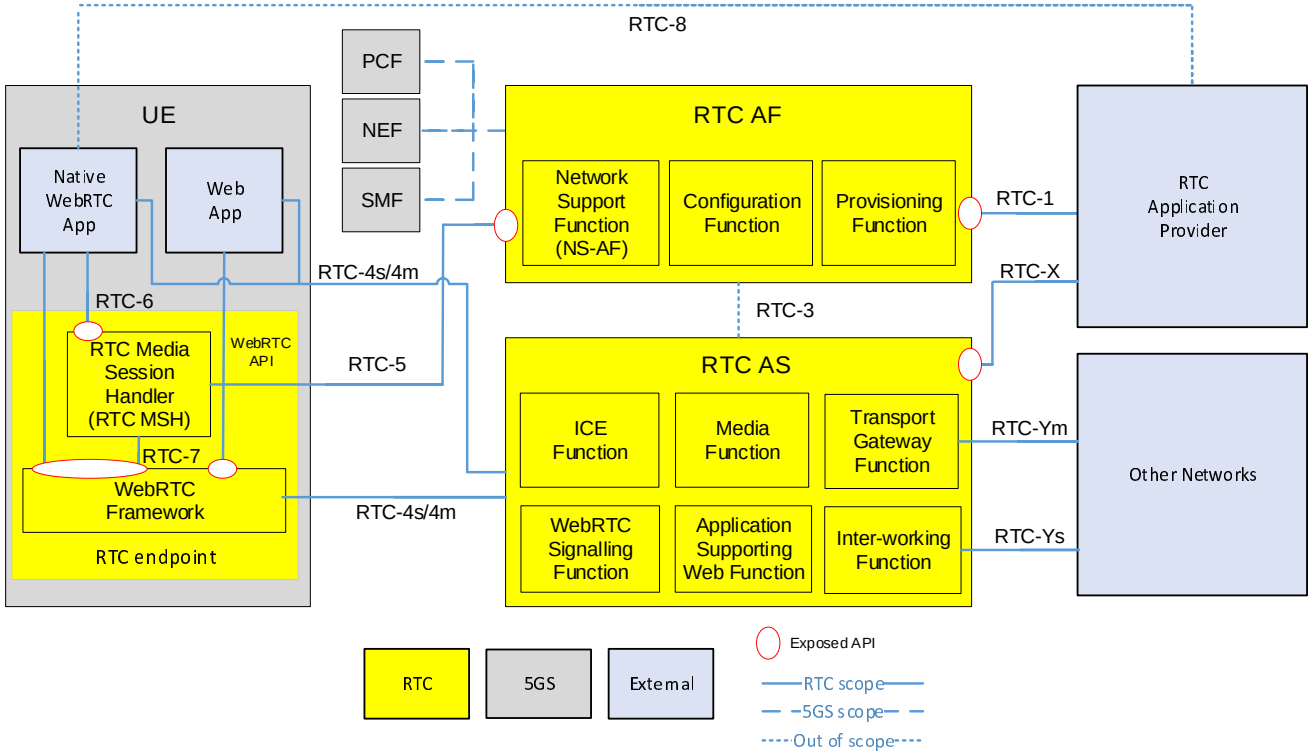


**Figure 6.2.7-1: Possible derivative RTC architecture for collaboration scenario 4**

NOTE 1: Other network includes RTC ASs in different MNO and service provider.

NOTE 2: If RTC AF and RTC AS are controlled by a single operator and located in the same operator network, these functions are trusted. Inter-working Function and Transport Gateway Function act as a border controller function at the boundary of the network.

The following interfaces are expected to be introduced for collaboration scenario 4.

- **RTC-Y**: This reference point is for C-Plane signalling and U-Plane media transport between RTC AS (Inter-working Function) and other RTC network or service provider network. This interface is necessary for inter-connect RTC-AS with other RTC network or service provider network to realize collaboration scenario 4. RTC-Y may further be grouped into two sub-interfaces as follows.

  i) **RTC-Ys**: This interface is for C-Plane signalling between Inter-working Function and other RTC network or service provider network.

  ii) **RTC-Ym**: This interface is for U-Plane media transport between Transport Gateway Function and other RTC network or service provider network.

The following interfaces are expected to be introduced/extended for collaboration scenario 3 and collaboration scenario 4. These interfaces are to enable operator assistance for RTC application providers and UEs, then these interfaces are used not only for inter-MNO scenario (Collaboration scenario 4) but also single MNO assistance scenario (Collaboration scenario 3).

- **RTC-X**: This interface is application interface between RTC AS and content provider, a form of RTC application provider. The interface is used for providing RTC AS functionalities via ASWF. (e.g., subscription of RTC resource in RTC-AS.). This interface is necessary for real-time interaction between RTC-AS and content provider for service control.

- **RTC-4m**: This interface needs to be extended for providing ASWF functionalities (e.g., application usage assistance such as downloading an application) to UE. This extension is necessary for providing RTC AS functionalities to UE as operator assistance.

The functional entities in Figure 6.2.2.1-1 correspond to the functions defined in 3GPP TS 26.506 [12] as follows:

- **WSF (WebRTC Signalling Function)**: WebRTC Signalling Function

- **WMCF (WebRTC Media Centre Function)**: Media Function
- **CSF (Conference Supporting Function)**: Application Supporting Web Function
- **WNSGF (WebRTC NNI Signalling Gateway Function)**: Inter-working Function
- **WNMGF (WebRTC NNI Media Gateway Function)**: Transport Gateway Function

NOTE 3: As described in 3GPP TS 26.506 [12], the integration/collocation of RTC AF and WebRTC signalling server is possible. Co-located WebRTC signalling server is able to act as a RTC AF which is accessible to 5GC, and replace some of this RTC AF's interfaces and APIs with WebRTC signalling. For example, interfaces and APIs between this RTC AF and UE will be replaced to avoid concurrent/redundant requests from UE.

The reference points in Figure 6.2.2.1-1 correspond to those defined in TS 26.506 [12] as follows:

- **Rs-u**: RTC-4s
- **Rs-n**: RTC-Ys
- **Rm-u**: RTC-4m
- **Rm-n**: RTC-Ym
- **Rh-u**: RTC-4m
- **Rh-n**: RTC-X

In the C-Plane signalling aspects, this document focuses on RTC-4 based solutions as shown in Figure 6.2.7-2 to support the collaboration scenario 4 and the case for the application which is not able to use MSH (e.g., Web App).

- RTC AF functionalities are integrated in WebRTC signalling function, since MSH is not used. Then, MSH related interfaces are omitted in Figure 6.2.7-2.
- Functions of RTC AF are integrated within WebRTC Signalling Function, then RTC-3 is out of the scope.
- The use and usage of ICE Function is optional functionality and is not used for non- NAT case. Then the extension of ICE functionality and its usage are outside the scope of this document, since no further extension is not identified in this document.
- The representation of RTC-4s and RTC-4m are simplified. Web App and Native WebRTC App are expected to use these interfaces as follows:
  * Web App utilizes the web browser's JS API (including WebRTC API) to send/receive signalling message on RTC-4s and media/data on RTC-4m.
  * Native WebRTC App utilizes the SDK provided by the OS of the UE to send/receive signalling message on RTC-4s and media/data on RTC-4m.

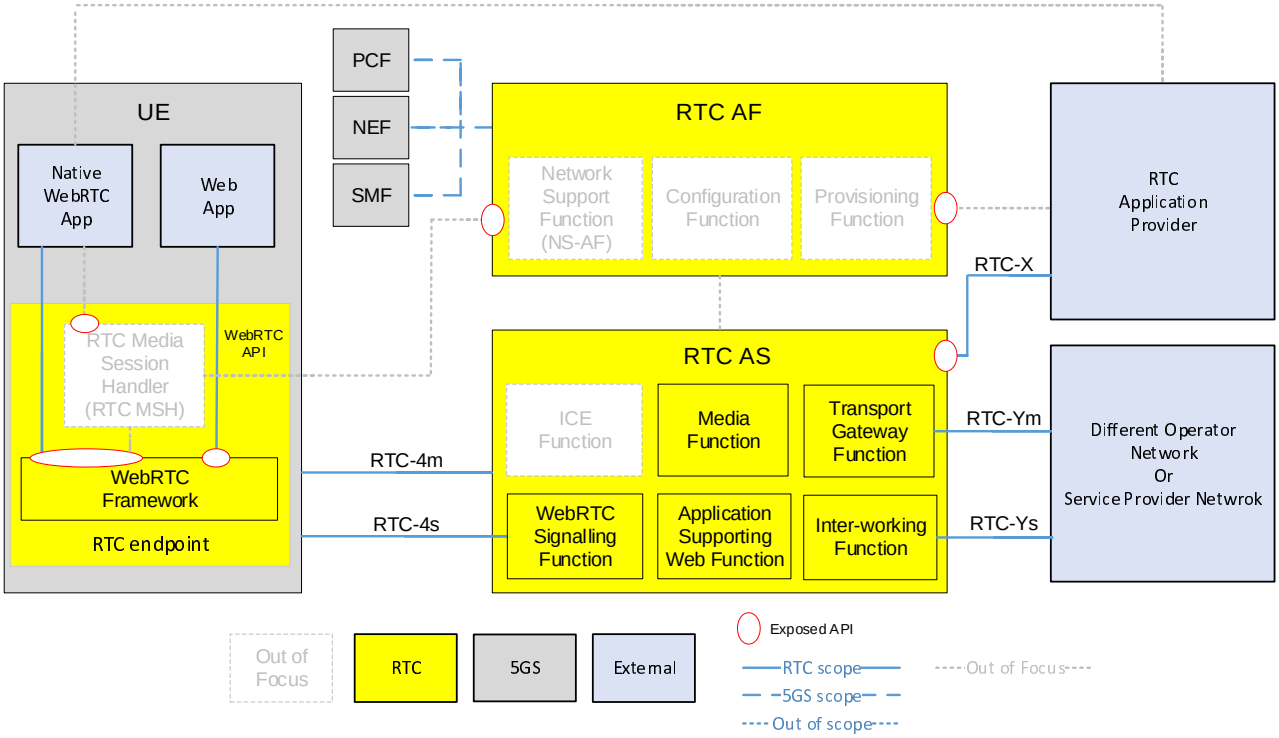


**Figure 6.2.7-2: The focused interface of C-Plane signalling protocol**

NOTE 4: RTC-4m is connected to ICE function when TURN server needs to be used. Otherwise, RTC-4m is connected to Media Function (MF) or Application Supporting Web Function (ASWF).

NOTE 5: The interfaces and the functionalities related to MSH, NS-AF, configuration function and provisioning function are not in the focus.

## 6.2.8 Proposed enhancements on RTC architecture

### 6.2.8.1 General

In this clause, the followings are described as proposed enhancements on RTC architecture.

1) The derivative RTC architecture supporting collaboration scenario 3 and 4 (see clause 6.2.8.2)
2) Enhancements on functionality in RTC AS functional entities (see clause 6.2.8.3)
3) Enhancements on reference points (see clause 6.2.8.4)
4) Enhancements on architecture diagrams in 3GPP TS 26.506 (see clause 6.2.8.5)

### 6.2.8.2 Derivative RTC architecture supporting collaboration scenario 3 and 4

This clause describes the derivative RTC architecture for collaboration scenario 3 and 4 according to the previous considerations. Figure 6.2.8.2-1 shows the derivative RTC architecture and reference points between RTC AS functions and other entities.

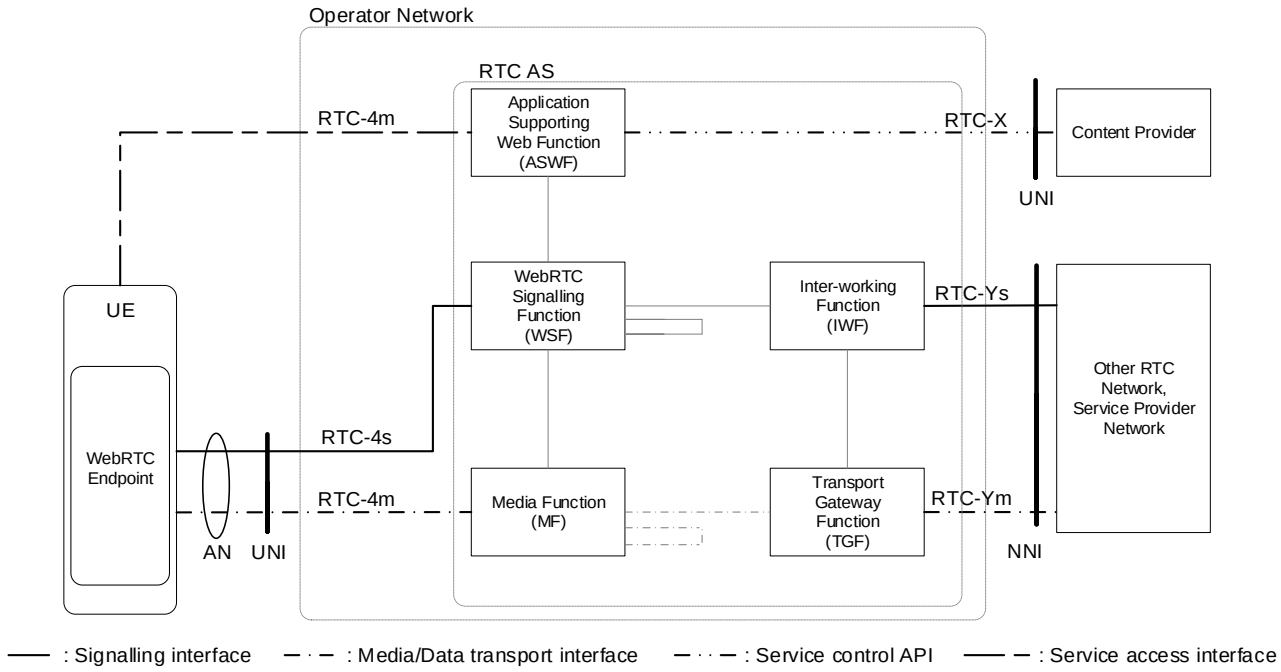


**Figure 6.2.8.2-1: Derivative RTC architecture diagram**

NOTE 1: WebRTC endpoint function of content provider connects to RTC AS via RTC-4s/RTC-4m (UNI). For simplicity, this line is snipped in this figure.

NOTE 2: NAT functionality and ICE functionality can be applied. However, these are snipped in this figure.

The derivative RTC architecture for collaboration scenario 3 and 4 with 5GC interaction viewpoint is shown in Figure 6.2.8.2-2. Network Support function (NS-AF) defined in 3GPP TS 26.506 [12] is integrated in the WSF to interact with 5GC via N5 interface.

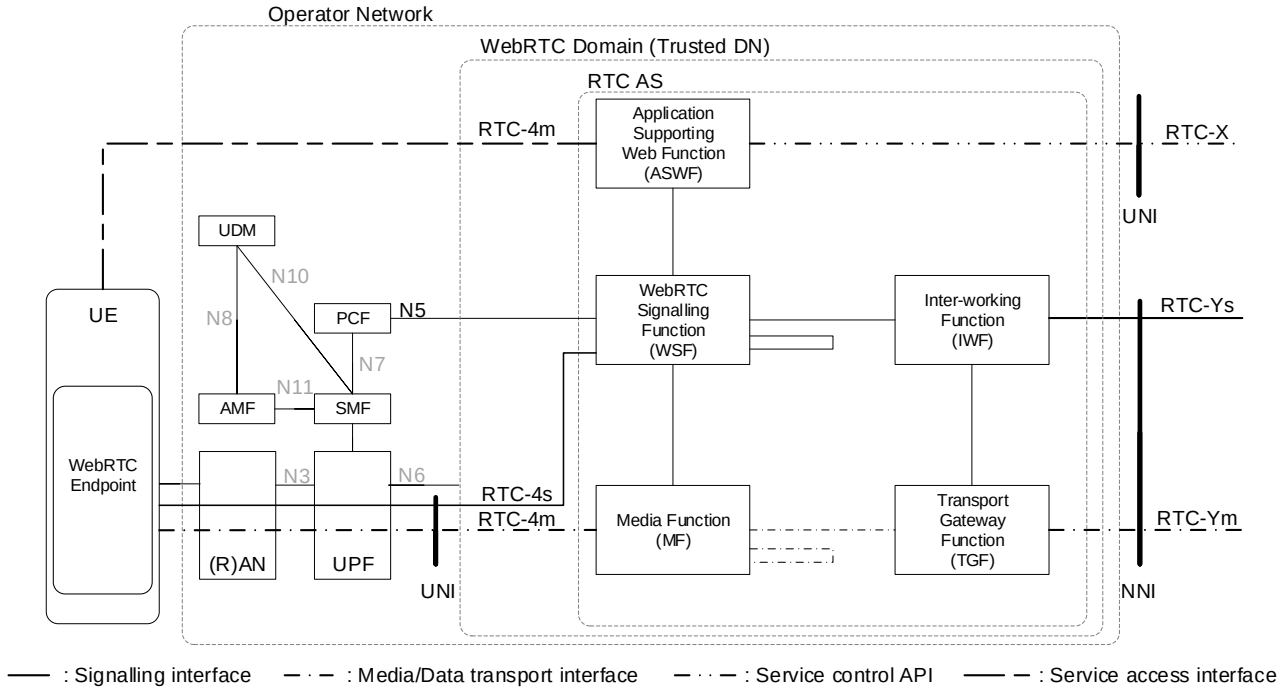


**Figure 6.2.8.2-2: Derivative RTC architecture diagram with 5GC interaction viewpoint**

### 6.2.8.3 Enhancements on functionality in RTC AS functional entities

#### 6.2.8.3.1 General

This clause describes the functionalities needed for the functional entities in the derivative RTC architecture, which are identified in this document.

#### 6.2.8.3.2 User Equipment

The User Equipment (UE) contains a user agent function for WebRTC. The user agent function is equivalent to "WebRTC Endpoint", which is either a WebRTC browser or a WebRTC non-browser as defined in IETF RFC 8825 [44]. The definitions of Web Browser and WebRTC Non-Browser in IETF RFC 8825 [44] are given below. WebRTC endpoint is the RTC endpoint supporting signalling related functionality of the application. Application itself is not scope of this document.

**WebRTC Browser (also called a "WebRTC User Agent" or "WebRTC UA")**: Something that conforms to both the protocol specification and the JavaScript API specification (W3C WebRTC 1.0 [65]).

NOTE 1: WebRTC browser is also called "web app" in this document.

**WebRTC Non-Browser**: Something that conforms to the protocol specification but does not claim to implement the JavaScript API. This can also be called a "WebRTC device" or "WebRTC native application".

When a content provider provides a service via UNI, the content provider acts as UE (i.e., WebRTC endpoint). Since this is not considered in the current versions of 3GPP TS 26.506 [12], a certain clarification on "content provider" in 3GPP TS 26.506 [12] is expected.

This document identifies the following functionality needed for the UE. Since these functionalities are not defined in the current versions of 3GPP TS 26.506 [12], the enhancement on the functional definition in 3GPP TS 26.506 [12] is expected.

- Support of WSF discovery mechanism (NOTE 2)

NOTE 2: This solution does not address the details of WSF discovery mechanism since this is addressed in Key Issue #6 and Solution #6.

#### 6.2.8.3.3 WebRTC Signalling Function

The WebRTC Signalling Function (WSF) is one of the RTC AS functional entities defined in 3GPP TS 26.506 [12]. The WSF is responsible for WebRTC signalling including capability exchange and management of media sessions between UEs and the RTC network. This functional entity is described as "Servers" or "Web Server" in clause 3 of IETF RFC 8825 [44]. Each operator or third-party in this document is assumed to have their own WSF in their RTC network.

This document identifies the following functionalities needed for the WSF. Since these functionalities are not defined in the current versions of 3GPP TS 26.506 [12], the enhancements on the functional definition in 3GPP TS 26.506 [12] are expected.

- Interaction with MF for media session control.
- Interaction with ASWF for collaboration with web applications/services.
- Interaction with 5GC, using network Support function (NS-AF).
- Authentication/authorization of the UE.
- Functionalities derived from service control API (i.e., connection control enforcer and RTC ID resource handling enforcer). (NOTE 1)
- Signing and verification of network-asserted UE's ID. (NOTE 2)

NOTE 1: This solution does not address the details of service control API since this is addressed in Key Issue #5 and Solution #5.

NOTE 2: This solution does not address the details of signing and verification of network-asserted UE's ID since this is addressed in Key Issue #10 and Solution #10.

NOTE 3: Regarding the retrieval of the identity of a UE from 5GC for authentication of UE, the WSF is not able to retrieve the identity from 5GC in the case that the UE is assigned an IPv4 private address behind NAT in the current release. In that case, the authentication mechanism in commercial use such as SMS OTP (One Time Password) is possibly applicable for enhancement of authentication of the UE.

#### 6.2.8.3.4 Media Function

The Media Function (MF) is one of the RTC-AS functional entity defined in 3GPP TS 26.506 [12]. The MF performs media processing. MF terminates media path (including data channel path) and performs media processing (e.g., mixing, selective forwarding, transcoding) which are required for immersive RTC applications. The MF is able to perform decryption and encryption of media packets if DTLS, SRTP, or TLS is used for a transport layer. The MF has the function of storing contents (including text or other static material as well as audio and video) and providing them to the UE. For Media transport control, the MF is able to interact with WSF.

In cases where an MF performs as a simple media relay function, the MF simply relays media data packets and supports IP packet connectivity. When a UE behave as ICE agents defined in IETF RFC 8445 [39] or IETF RFC 8838 [49], the MF may be either STUN servers defined in IETF RFC 8489 [41] for connectivity check or TURN servers defined in IETF RFC 8656 [42] for relaying media data packets. This functional entity facilitates NAT traversal of UE and the connectivity between UE and other network functions.

This functional entity is generally implemented in WebRTC Multipoint Control Unit (MCU) or Selective Forwarding Unit (SFU).

This document identifies the following functionality needed for the MF. Since the functionality is not defined in the current versions of 3GPP TS 26.506 [12], the enhancement on the functional definition in 3GPP TS 26.506 [12] is expected.

- Functionalities derived from service control API (i.e., media data forwarding control enforcer and RTC exchange resource handling enforcer for service control).

NOTE: This solution does not address the details of service control API since this is addressed in Key Issue #5 and Solution #5.

#### 6.2.8.3.5 Application Supporting Web Function

The Application Supporting Web Function (ASWF) is one of the RTC AS functional entities defined in 3GPP TS 26.506 [12]. This document identifies the following functionalities needed for the ASWF. Since these functionalities are not clearly defined in the current versions of 3GPP TS 26.506 [12], the enhancements on the functional definition in 3GPP TS 26.506 [12] are expected.

- Exposing the service control APIs. (NOTE 1)
- Storage of user subscription data specific to MNO's WebRTC services. (NOTE 2)
- Authorization endpoint and token endpoint of OAuth 2.0 described in IETF RFC 6749 [29] for establishing authentication linkage between MNO's ID and RTC application provider's ID. (NOTE 3)
- Providing supplementary files (e.g., icon images of participants, and shared documents) via best-effort transport different from the channels for real-time media.
- Providing WSF discovery functionality (NOTE 4).

NOTE 1: This solution does not address the details of service control APIs since this is addressed in Key Issue #5 and Solution #5.

NOTE 2: In this document, it is assumed that a single user (i.e., identity) and its subscription data (associated with the identity) are assigned, owned, and managed by both MNO and application provider independently. The two identities have a link with each other via some technique. User subscription data specific to application provider's services are stored in their networks.

NOTE 3: OAuth token will be used to C-Plane authentication at WSF and RTC application providers. STUN/TURN authentication with OAuth token is defined in IETF RFC 7635 [32]. Portal http(s) servers of WebRTC services provide this function in general implementations.

NOTE 4: This solution does not address the details of WSF discovery functionality since this is addressed in Key Issue #6 and Solution #6.

#### 6.2.8.3.6 Inter-working Function

The Inter-working Function (IWF) is one of RTC AS functional entity defined in 3GPP TS 26.506 [12]. The IWF is located at the boundary of the RTC network where different operator or third-party network inter-connects.

The IWF is inserted into "Signalling Path" in Figure 2 of IETF RFC 8825 [44] and responsible for border control functions and supports session establishment between disparate address realm's networks. By inserting the IWF into "Signalling Path", each operator or 3rd-party network can securely inter-connect with the other network.

This document identifies the following functionalities needed for the IWF. Since these functionalities are not defined in the current versions of 3GPP TS 26.506 [12], the enhancements on the functional definition in 3GPP TS 26.506 [12] are expected.

- C-plane signalling protocol interworking between RTC network and IMS network. (NOTE 1)
- Signing and verification of network-asserted UE's ID. (NOTE 2)

NOTE 1: This solution does not address the details of interworking with IMS network since this is addressed in Key Issue #7 and Solution #7.

NOTE 2: This solution does not address the details of signing and verification of network-asserted UE’s ID since this is addressed in Key Issue #10 and Solution #10.

#### 6.2.8.3.7 Transport Gateway Function

The Transport Gateway Function (TGF) is one of RTC AS function entity defined in 3GPP TS 26.506 [12]. The TGF is a media relay located at the boundary of the RTC network where different operator or third-party inter-network connects. The TGF is the function responsible for the border control and transport of media data packets between different networks. The TGF is responsible for the border control and transport of media data packets between different networks.

This document identifies the following functionality needed for the TGF. Since the functionality is not defined in the current versions of 3GPP TS 26.506 [12], the enhancement on the functional definition in 3GPP TS 26.506 [12] is expected.

- U-Plane protocol interworking between RTC network and IMS network.

NOTE: This solution does not address the details of interworking with IMS network since this is addressed in Key Issue #7 and Solution #7.

### 6.2.8.4 Enhancements on reference points

The reference points shown in Figure 6.2.8.2-1 (Derivative RTC architecture diagram) are listed in Table 6.2.8.4-1.

The reference points marked as "No" in the 3rd column of Table 6.2.8.4-1 are expected to be introduced in 3GPP TS 26.506 [12].

**Table 6.2.8.4-1: Reference points used for derivative RTC architecture**

| Reference point (NOTE) | Descriptions | 3GPP TS 26.506 [12] already define? |
|---|---|---|
| RTC-4 | Reference Point between an RTC network and a UE for C/U-Plane. | Yes |
| RTC-4s | Reference Point between a WSF and a UE for C-Plane signalling. | Yes |
| RTC-4m | Reference Point between a MF and a UE or between an ASWF and a UE for U-Plane. | Yes |
| RTC-X | Reference Point between a ASWF and a content provider network for service control. | No |
| RTC-Y | Reference Point between an RTC network and another network (i.e., other operator network or service provider network) for C/U-Plane. | No |
| RTC-Ys | Reference Point between a IWF and another network (i.e., other operator network or service provider network) for C-Plane signalling. | No |
| RTC-Ym | Reference Point between a TGF and another network (i.e., other operator network or service provider network) for U-Plane. | No |
| N5 | Reference Point between a WSF and PCF in 5GC. | Yes |
| NOTE: RTC-X/Y reference points need to be assigned/defined considering the common architecture for 5GMS and RTC. | | |

## 6.2.8.5 Enhancements on architecture diagrams in 3GPP TS 26.506

This clause describes the expected enhancements on architecture diagrams in 3GPP TS 26.506 [12].

The expected enhancements on RTC general architecture are shown in Figure 6.2.8.5-1. RTC-4 reference point is connected to UE rather than WebRTC Framework since the interface is used for C-Plane signalling between application in the UE and RTC AS in addition to U-Plane media (audio/video stream) and data between RTC endpoint and RTC AS. RTC-X reference point and RTC-Y reference point are newly introduced.

NOTE 1: RTC-X is applicable between the RTC AS (ASWF) and the content provider in Figure 6.2.8.5-1.

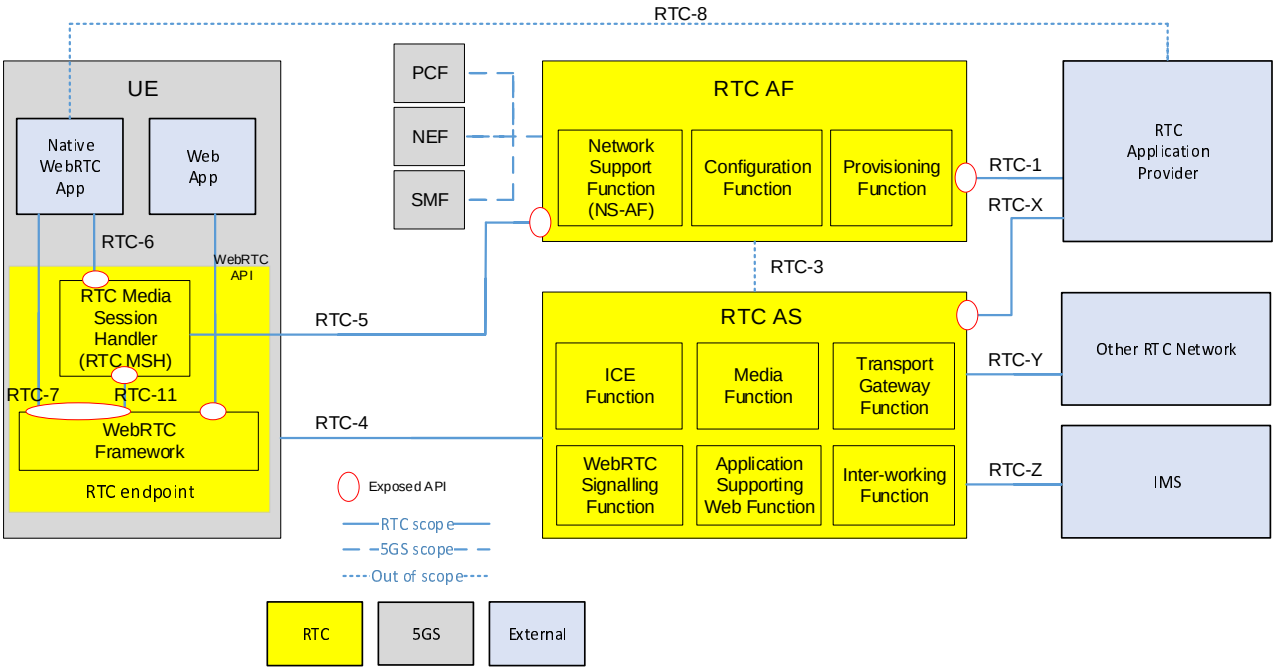


**Figure 6.2.8.5-1: Expected enhancements on RTC General Architecture**

Figure 6.2.8.5-2 shows the expected enhancements on derivative RTC architecture for collaboration scenario 3 defined in 3GPP TS 26.506 [12]. RTC-4m reference point is clarified that this interface is used for providing ASWF functionality to UE, and RTC-X reference point is newly introduced to provide the service control API for content provider, a form of RTC application provider, from ASWF.

NOTE 2: RTC-X is applicable between the RTC AS (ASWF) and the content provider in Figure 6.2.8.5-2.

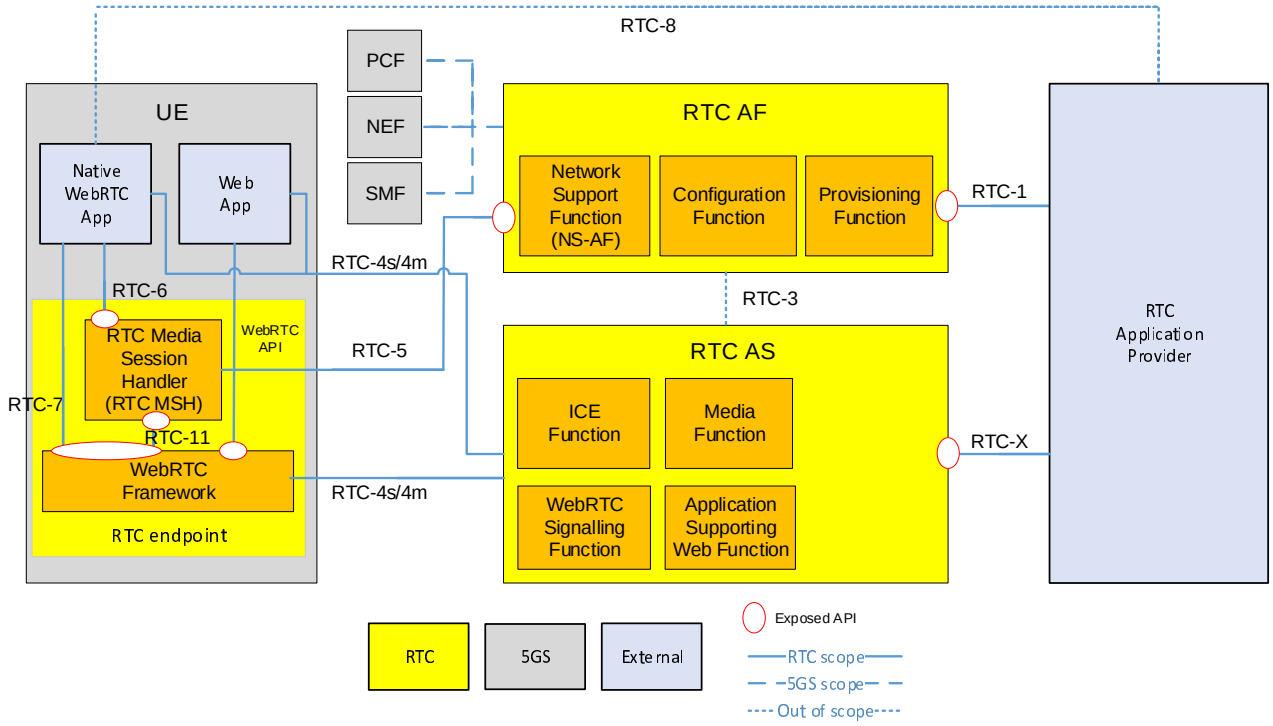


**Figure 6.2.8.5-2: Expected enhancements on derivative architecture for collaboration scenario 3**

Figure 6.2.8.5-3 shows the expected derivative RTC architecture for collaboration scenario 4. Collaboration scenario 4 supports inter-operable WebRTC services. Then collaboration scenario 3 is extended with functions and interfaces to support MNO to MNO inter-operability. RTC-Y (RTC-Ys and RTC-Ym) reference point is introduced to support the inter-connection between MNO's RTC ASs.

NOTE 3: RTC-X is applicable between the RTC AS (ASWF) and the content provider in Figure 6.2.8.5-3.

NOTE 4: Other RTC network in Figure 6.2.8.5-3 includes other operator's network and service provider's network.

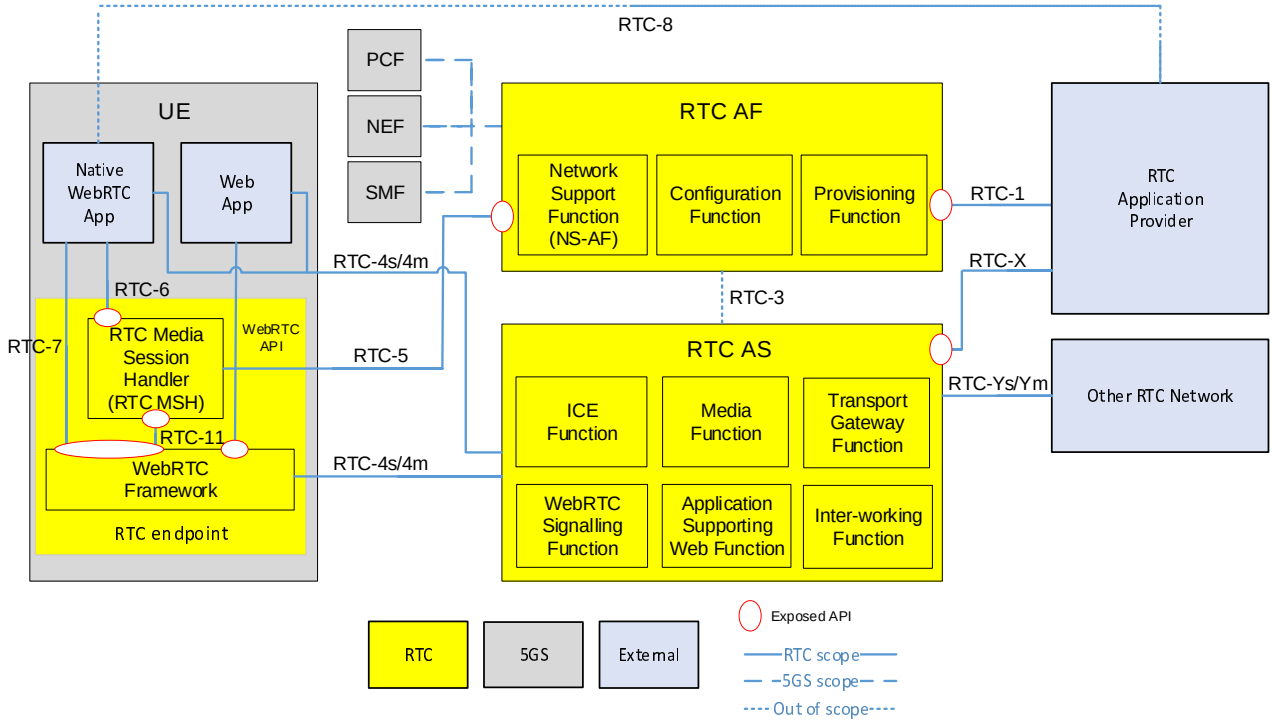


**Figure 6.2.8.5-3: Expected derivative architecture for collaboration scenario 4**

### 6.2.9 Solution evaluation

The proposed architecture in clause 6.2.8 supports the functionalities and capabilities to support immersive RTC services for collaboration scenario 4 (also applicable for collaboration scenario 3) and the architecture is consistent with RTC architecture in 3GPP TS 26.506 [12]. Then it is proposed to;

- reflect the architectural enhancements on functional entities, reference point described in clause 6.2.8 into the stage 2 specification of RTC (i.e., 3GPP TS 26.506 [12]) and
- based on the above enhancements, update the architecture diagrams (RTC general architecture, derivative architectures for collaboration scenario 3 and 4.

## 6.3 Solution #2: Functional requirements for C-Plane

### 6.3.1 Solution description

This solution addresses Key Issue #2.

This solution identifies the functional requirements for control plane (C-Plane) signalling for WebRTC-based RTC session supporting inter-operator connection (i.e., collaboration scenario 4 in 3GPP TS 26.506 [12]) in addition to collaboration scenario 3 in 3GPP TS 26.506 [12].

Figure 6.3.1-1 shows the C-Plane reference points on the derivative RTC architecture. RTC-4s an RTC-Ys are focussed reference points as described in Solution #1.

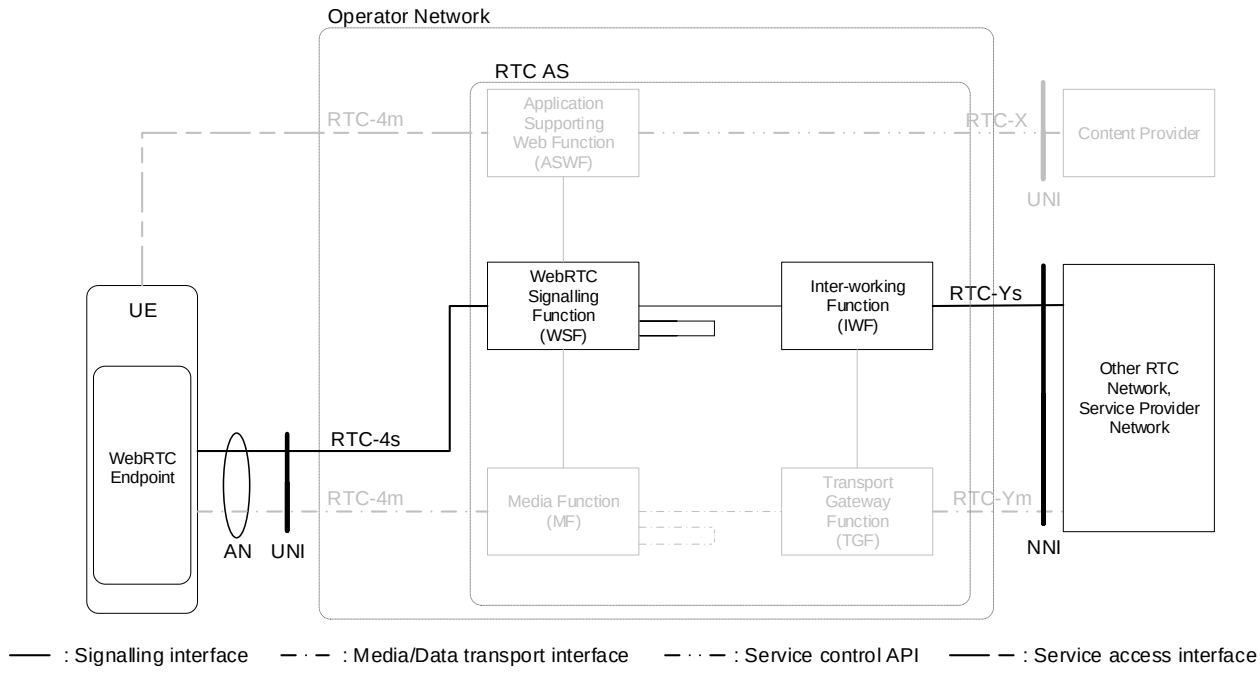


**Figure 6.3.1-1: Reference points for C-Plane**

To support collaboration scenario 3, RTC-4s needs to be supported. According to 3GPP TS 26.506 [12] clause 4.3.3, RTC-4s supports the exchange of signalling information related to the WebRTC session between two or more WebRTC endpoints using trusted application servers.

To support collaboration scenario 4, RTC-Ys is introduced in the enhanced RTC architecture as a new C-Plane interface for signalling information exchange between two different operator's networks or between an operator network and a service provider network.

## 6.3.2 Functional requirements for C-Plane interface

### 6.3.2.1 General

This clause identifies the functional requirements for C-Plane interface to enable WebRTC-based RTC media session supporting collaboration scenario 3 and 4. The requirements are considered based on following aspects:

1. Support of WebRTC based RTC services (general aspect)
2. Transport of signalling message
3. Media session control and management

### 6.3.2.2 Support of WebRTC based RTC services

This clause identifies the functional requirements on C-Plane interface to support WebRTC based RTC services.

1. It is required to support any WebRTC application (i.e., it should not be overfit for a specific use case.).
    a. It is required to support any kind of WebRTC endpoints (e.g., web browser).
    b. It is required to allow application specific methods and information elements.
2. It is required to be web-friendly to support rapid and easy deployment in web environments:
    a. by using web technologies such as HTTP, JSON, etc…
    b. complying with WebRTC standards (e.g., SDP for session description and supporting the exchange of ICE candidates, etc…) defined in IETF and W3C, with an exception for codecs.
3. It is required to be able to be simple to implement and deploy (e.g., simpler in complexity compared to SIP).
4. It is required to be able to authenticate and authorise the UE using RTC services.
5. It is required to protect user privacy and mitigate the linkability and tracking attack caused by unnecessary user information disclosure.

### 6.3.2.3 Transport of signalling message

This clause identifies the functional requirements on transport of signalling message.

1. It is required to be web-friendly to support rapid and easy deployment in web environments by using web technologies such as WebSocket, etc…
2. It is required to support the secure exchange of messages supporting integrity-protection and/or encryption.
3. It is required to be support connection management mechanisms (e.g., keep alive) for reliable exchange of signalling messages.

### 6.3.2.4 Media session control and management

This clause identifies the functional requirements on media session control and management.

1. It is required to support following methods for media session control.
    a. media session set up
    b. media session update
    c. media session disconnection
2. It is required to support a method for querying information from a connected network. The information includes the service configuration information such as server address.

3. It is required to be able to set up a media session with any kind of RTC resources (e.g., WebRTC endpoint on the UE, conference, metaverse).
4. It is required that an WebRTC endpoint is able to set up multiple media sessions simultaneously.
5. It is required to support incoming call set up (i.e., be able to receive a media session set up request).
6. It is required to be able to set up a media session with RTC resources in different operator network or RTC application provider network. This requirement is applied for both of the following cases.
   a. The connected network support RTC AS functionalities. (i.e., connected via NNI)
   b. The connected network does not support RTC AS functionalities. (i.e., connected via UNI)
7. It is required to be familiar with existing web-services to exchange media capabilities. It is also required that WebRTC endpoints can notify own media capabilities to a network, and network can handle the notified media capability appropriately.
8. It is required to support a mechanism to exchange meta data associated with media session.
9. It is required to support QoS control of a media session based on the information contained in the signalling message related to the media session.
10. It is required to be able to negotiate the use of optional features.
11. It is required to support the mechanisms for reliable media session control. (e.g., error handling).
12. It is required to be able to identify the RTC service user uniquely. The identity of the user is able to be associated with multiple devices (WebRTC endpoints) belongs to the user.
13. It is required to be able to enable communicating parties to verify each other's identity, if required by application.

## 6.3.3 Protocol stack for C-Plane interface

### 6.3.3.1 General

This clause identifies the appropriate protocol stack for C-plane interfaces, considering the requirements in clause 6.3.2. Especially, the following requirements are considered:

- It is required to support the secure exchange of messages supporting integrity-protection and/or encryption.
- It is required to protect user privacy and mitigate the linkability and tracking attack caused by unnecessary user information disclosure.
- It is required to be web-friendly to support rapid and easy deployment in web environments:
  - by using web technologies such as JSON, WebSockets, etc…
  - complying with WebRTC standards (e.g., SDP for session description and supporting the exchange of ICE candidates) defined in IETF and W3C, with an exception for codecs.
- It is required to be simple to implement and deploy (e.g., simpler in complexity compared to SIP).

### 6.3.3.2 Base protocol

HTTP (IETF RFC 9110 [54], IETF RFC 9111 [55], IETF RFC 9112 [56], and IETF RFC 9113 [57]) / HTTPS and WebSocket (IETF RFC 6455 [27]) are available options to transport signalling message between UE and WSF so that connection setup procedure could be invoked by JavaScript API as described in clause 3 of IETF RFC 8825 [44]. Nevertheless, HTTP / HTTPS is less appropriate for two reasons described in clause 1.1 of IETF RFC 6455 [27]:

- Server load caused by http transactions (based on request-response)
- A connection has two sessions each for sending and receiving signalling packets

In addition, when a notification from the network to the UE is required, for such as an incoming call, an HTTP(S) connection is originated from the network side, but this case has some problem. Generally, NAT box is placed between UE and network entities, therefore NAT-traversal problem should be resolved. Besides, in terms of security configuration, UEs often deny incoming TCP (IETF RFC 9293 [61]) connections.

WebSocket fulfils the requirement for secure transport of signalling messages since WebRTC supports the secure transport over TLS.

For those reasons mentioned above, only WebSocket over TLS is utilized as the base protocol for transport of signalling messages. WebSocket can solve the three problems, server load, number of sessions and the NAT-traversal.

#### 6.3.3.3 Upper layer protocol over WebSocket

In IETF RFC 8825 [44], upper layer protocols over WebSocket are not specified and are thought to be application specific. In the IETF RFC 8825 [44], SIP (IETF RFC 3261 [21]) and XMPP (IETF RFC 6120 [26]) are listed as candidate upper layer protocols for C-Plane signalling.

##### 6.3.3.3.1 SIP

Utilizing SIP for C-Plane signalling for WebRTC is already described in clause 5 of 3GPP TS 24.371 [9]. One of the main advantages of using SIP is the ease of interworking between WebRTC-aware network and IMS network. On the other hand, disadvantages of using SIP are as follows:

- UE and network must be able to understand both WebRTC and SIP. SIP is not widely used outside of telephony. If SIP must be used in conjunction with WebRTC, the advantage of WebRTC, friendliness to web-based development environments and developers, is to be spoiled.
- SIP has a strictly managed communication model as SIP dialog. In principle, the originated signalling is transparently relayed through the network and the terminals manage the dialog with each other. These characteristics are not compatible with the UE-network relation model, which is the scope of this document.
- SIP specifies methods divided by each signalling characteristic (i.e., INVITE, ACK, BYE, CANCEL, OPTIONS, PRACK, UPDATE, SUBSCRIBE, NOTIFY, REFER, PUBLISH, INFO). Adding control for a new characteristic may need to start from the method definition.
- Less affinity with cloud environment where HTTP is mainly used. For example, raw values of the IP addresses related to the SIP dialog (consisting of a communication path of SIP trapezoid) are in the protocol header or message body, therefore changing communication elements is difficult once the call session is established.

For the reasons above, more optimal protocol for the upper layer of C-Plane signalling is expected to be identified.

##### 6.3.3.3.2 XMPP

There is no specification using XMPP for the upper layer protocol of the C-Plane signalling in 3GPP and no major commercial implementations of WebRTC either. The reason seems that XMPP can be used on its own and does not need to be combined with other protocols. WebSocket encapsulation of XMPP has little benefit except the case that an application using XMPP is implemented using JavaScript.

For the reasons above, more optimal protocol for the upper layer of C-Plane signalling is expected to be identified.

##### 6.3.3.3.3 Other existing implementations

Among the existing implementations of WebRTC communication services, JSON (IETF RFC 8259[36]) format is mainly used as a format for the upper layer of C-Plane signalling. This is because JSON format is easy to handle in JavaScript. Taking this advantage, this document investigates more optimal protocol using JSON format for the upper layer of C-Plane signalling protocol.

In 3GPP specifications, RESTful APIs (such as service-based interface and Northbound APIs) are often defined using OpenAPI 3 (OpenAPI [64]) and the message-body of the APIs are based on JSON. However, OpenAPI is mainly suitable for RESTful APIs and not suitable for message-driven APIs such as C-Plane signalling over WebSocket. There is another possible API specification for JSON based API. AsyncAPI [63] (managed by Linux Foundation) is a message/event-driven architecture concept and familiar with message-driven API. For this reason, AsyncAPI [63] is used for identifying API schemas in this document.

#### 6.3.3.4 Proposed Protocol Stack

The protocol stack for C-Plane interface is shown in Figure 6.3.3.4-1. As described above, JSON based protocol over WebSocket over TLS is an expected solution for C-plane signalling protocol.

WebSocket can be deployed over several versions of HTTP.

- WebSocket with HTTP/1.1 is specified in IETF RFC 6455 [27] and used in this document. HTTP/1.1 is not, however, shown in the protocol stack because HTTP/1.1 does not remain after upgrading into WebSocket.
- WebSocket with HTTP/2 is specified in IETF RFC 8441 [38] and used in this document. HTTP/2 is shown in the protocol stack because HTTP/2 framing remains after a stream in HTTP/2 connection is upgraded into WebSocket.
- WebSocket with HTTP/3 (IETF RFC 9114 [58]) is specified in IETF RFC 9220 [60] but not used in the current version of this document. The transport protocol used over HTTP/3 needs to be selected in alignment with IETF/W3C discussions.

The sub layers of each protocol are according to the existing specifications.

- TLS under HTTP/1.1 and HTTP/2 is specified in IETF RFC 8446 [40].
- TCP under TLS is specified in IETF RFC 9293 [61].
- IPv4 and IPv6 under TCP are specified in IETF RFC 791 [20] (IPv4) and IETF RFC 8200 [33] (IPv6).

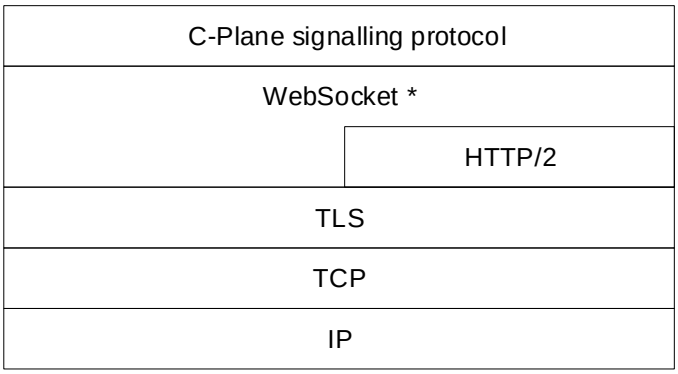


**Figure 6.3.3.4-1: Protocol Stack for C-Plane interface**

### 6.3.4 Solution evaluation

There is no misalignment between the functional requirements proposed in clause 6.3.2 and 6.3.3, and those specified in 3GPP SA4 RTC specifications (i.e., 3GPP TS 26.506 [12] and 3GPP TS 26.113 [10]), therefore it is proposed to develop the C-Plane signalling protocol based on the proposed functional requirements and protocol stack.

## 6.4 Solution #3: C-Plane signalling protocol

### 6.4.1 Solution description

This solution addresses Key Issue #3.

This clause describes a possible control plane signalling protocol for WebRTC-based immersive RTC session management supporting the inter-operator connection (i.e., collaboration scenario 4 in 3GPP TS 26.506 [12]) based on the enhanced architecture described in clause 6.2 (Solution #2) and functional requirements for C-Plane in clause 6.3 (Solution #3). This control plane protocol is also applicable to collaboration scenario 3 since collaboration scenario 4 is an extension of collaboration scenario 3.

The C-Plane signalling protocol specification in this document is named as RESPECT (REaltime&REality media Setup Protocol, Extensible and CompacT).

## 6.4.2 Overview

### 6.4.2.1 General

The RESPECT is a signalling protocol intended for various media session as described in the clause 4 (motivation).

This clause describes the architectural model considered in the design of signalling protocol.

### 6.4.2.2 Basic connection model

#### 6.4.2.2.1 General

This clause describes basic connection model considered in the design of RESPECT protocol.

#### 6.4.2.2.2 Trapezoid model

The RESPECT protocol is designed under the trapezoid model as shown in Figure 6.4.2.2.2-1.

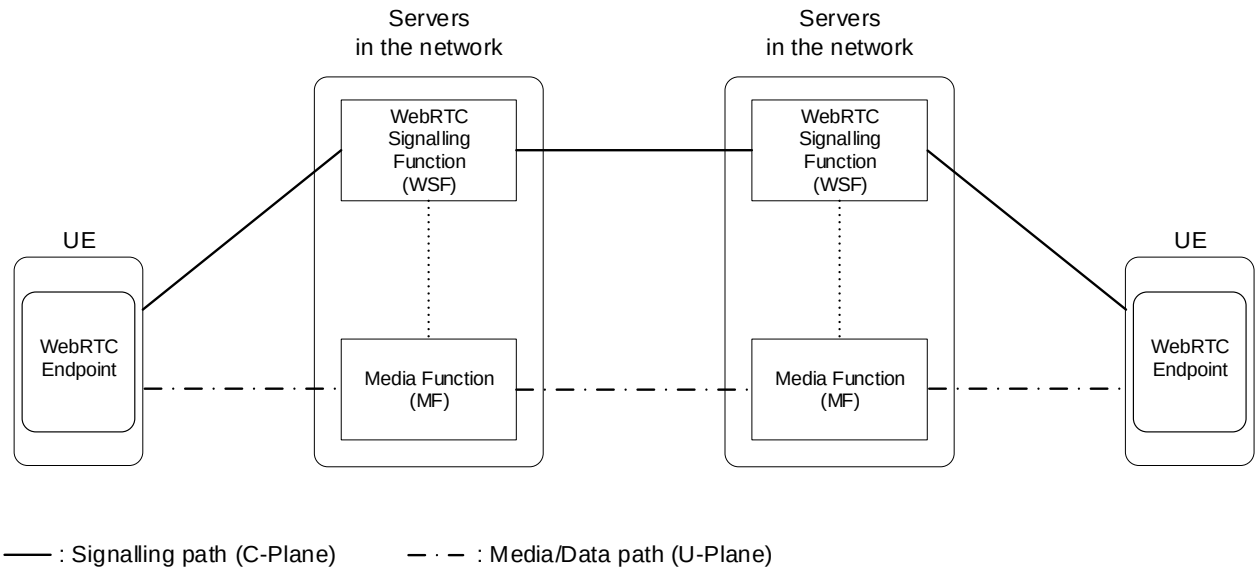


**Figure 6.4.2.2.2-1: Trapezoid model in the protocol design**

WebRTC Signalling Function (WSF) and Media Function (MF) in the network are responsible for providing reliable high-quality RTC services.

The WSF for C-Plane terminates all signalling messages from the WebRTC endpoints and the other WSFs. This behaviour is equivalent to the behaviours of a back-to-back user agent (B2BUA) in SIP as defined in IETF RFC 7092 [30]. By terminating all signalling messages, the WSF fully manages the media session and provides QoS interacting with the 5GC and the MF.

The MF for U-Plane involves all media/data paths of WebRTC endpoints unless a direct media session is established between the WebRTC endpoints, or the other function (e.g., Transport Gateway Function (TGF)) provides U-Plane functionalities instead of the MF. The MF can control and monitor all media/data sessions.

#### 6.4.2.2.3 Client-Server model

The RESPECT protocol is also designed under the client-server model between two entities. Figure 6.4.2.2.3-1 shows an example model between a WebRTC endpoint as a RESPECT client and a WSF as a RESPECT server.

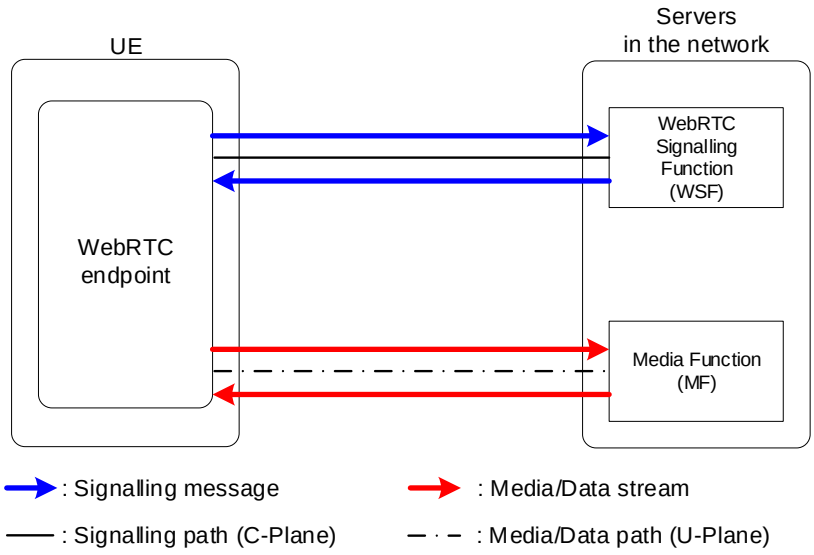


**Figure 6.4.2.2.3-1: Client – Server model in the protocol design**

The WebRTC endpoint on the UE is aware of a single entity (i.e., a WSF in the network). The WSF takes care of everything behind the WSF toward the destination RESPECT endpoint in case where three or more RESPECT endpoint(s) are involved in a media session like trapezoid model.

The benefits brought by the model is simplification of the protocol between two entities (e.g., client and server) rather than among three and more (e.g., two endpoints and servers). For example;

- All request/response messages are defined between two RESPECT endpoints (e.g., WebRTC endpoints on the UE and WSF in the network). The WebRTC endpoint on the UE does not need to care the transactions behind the connected WSF.
- Signalling message routing is performed by the network without involvement of UE. Therefore, only thing to do at the UE for routing of signalling messages is to specify the destination identifier like a URI.
- The endpoint/application specific characteristics (e.g., capabilities and services) can be converted/terminated by the server in the network.

### 6.4.2.3 Target use case

See clause 6.2.4.2 (Target use cases from network view) and annex A (use cases) as examples of media connection to be achieved by the media session control of the RESPECT protocol.

### 6.4.2.4 Target architecture and reference points

See clause 6.2.8 (Proposed enhancements on RTC architecture). The RESPECT protocol is intended to be applied on C-Plane signalling interfaces (i.e., RTC-4s and RTC-Ys) on the enhanced RTC architecture.

### 6.4.2.5 Protocol stack

See clause 6.3.3.4 (Proposed Protocol Stack) for the protocol stack of C-Plane interface.

See clause 6.5.3 (Protocol Stack) for the protocol stack of U-Plane interface.

## 6.4.3 High-level features

### 6.4.3.1 General

This clause describes the high-level features of C-Plane signalling required to be considered for realizing RTC services.

#### 6.4.3.2 List of high-level features

The high-level features to be considered in the RESPECT protocol design are described as follows.

1) Transport usage and management for signalling messages

   - Signalling messages are exchanged over the Secure WebSocket connection as described in clause 6.3.3.
   - The principle for the Secure WebSocket connection usage and management (e.g., keep alive) for RESPECT protocol needs to be considered to fulfil the functional requirements for transport of signalling message described in clause 6.3.2.3.
   - The principle of usage and management of Secure WebSocket connection is described in clause 6.4.4.

2) Media session control and management

   - The RESPECT protocol uses two types of sessions for reliable media session control to fulfil requirements described in clause 6.3.2.4; one is "control session", and the other is "media session".
   - The "control session" is a Secure WebSocket between two directory connected RESPECT endpoints managed with a state of authentication by using RESPECT protocol.
   - The "media session" is a concept for managing the media/data transported over U-Plane at C-Plane entities. This media session is identified by media session ID set in the signalling message.
   - The principles of these session are described in clause 6.4.5.2.2 and clause 6.4.5.2.3.

3) Transaction management

   - The RESPECT protocol is transaction-based protocol. To comply with RTC service requirements, transaction timeout feature is supported.
   - The principles of transaction management are described in clause 6.4.5.2.4.

4) Supported method

   - The following types of methods are supported to fulfil the functional requirements for media session control and management described in clause 6.3.2.4. The RESPECT defines the minimum set of method type and its extensibility and flexibility is achieved by the information elements in a request and response, or application specific method if needed.

     i) Authentication

        - This method is used by the RESPECT client to get authenticated by the RESPECT server. To enable the RESPECT client to send / receive signalling messages other than the signalling messages for authentication, the RESPECT client needs be authenticated by the RESPECT server.

     ii) Media session control (set up / update / disconnect)

        - The following methods are used by the RESPECT endpoint to control media session(s).

          a) Media session set up

          b) Media session update (modification)

          c) Media session disconnection

     iii) Information query

        - This method provides the alternative to the information queries instead of using RTC MSH and RTC AF via RTC-5 interface. The RESPECT client is allowed to send an information query request to the RESPECT endpoint in the network for getting information from the network.

     iv) application specific method

- Application specific method(s) is required to be applicable. The application specific method is required to be distinguished from the method defined in 3GPP specifications and guaranteed the uniqueness between any applications.

5) Feature negotiation

- To support the use of application specific features, a feature negotiation mechanism is required to be supported in the signalling message.
- The principles of feature negotiation are described in clause 6.4.5.2.7.

6) Identification of users and media resources

- The destination identities used for RESPECT are required to be defined for media session set up with appropriate media resource. Also, the originating user's identities are required to be defined to fulfil the functional requirements described in clause 6.3.2.4.
- The specification of these identities is defined in the subsequent clauses. The requirements specific to the originating user's identities are addressed in Key Issue #10 (Security considerations).

The high-level features to be considered in the RESPECT endpoint are described as follows.

1) WSF discovery

- A RESPECT client is required to be able to find a WSF where RESPECT messages are sent to. This mechanism is described in Solution #6 (WSF Discovery mechanism).

2) Message routing

- To enable authentication and media session setup, the WSF is required to resolve a next hop of a request in collaboration with the ASWF. The IWF is also required to support this feature.
- The procedures for this feature are described in the subsequent clauses.

3) QoS control

- The WSF is responsible for the QoS control of media sessions (i.e., U-Plane traffic). The WSF is required to interact with the 5GC (i.e., PCF or NEF) to reserve resources for a media session according to 3GPP TS 23.501 [4], 3GPP TS 23.502 [5], 3GPP TS 23.503 [5A]. The WSF is required to support the functionality to interact with MF to enforce IP packet flow control (e.g., Gate control, traffic policing, QoS packet marking).

NOTE: The WSF determines the QoS policy for the media session, based on the media session related information (e.g., User subscription, media type, SDP information) and operator policy.

4) IMS interworking

- An RTC network supports the interworking with IMS network. This feature is addressed in Solution #10 (Protocol-level interworking between RTC network and IMS network).

## 6.4.4 Transport of signalling massage

### 6.4.4.1 General

The signalling massage of RESPECT protocol is sent over Secure WebSocket connection specified as one of the transport protocol for C-Plane in 3GPP TS 26.113 [10]. The WebSocket URI is required to be consistent with the URI in clause 6.4.5.4 of this document.

For the purposes of this document, the following terminologies are used in this document as defined in clause 3.1.

- RESPECT client: A signalling agent supporting RESPECT protocol acting as a Secure WebSocket client.
- RESPECT server: A signalling agent supporting RESPECT protocol acting as a Secure WebSocket server.
- RESPECT endpoint: A signalling agent acting as RESPECT client or RESPECT server or both.

The Origin header field is not required to be set in an HTTP request. If an Origin header field is included in the HTTP request, RESPECT endpoint acting as a WebSocket server ignores the Origin header field.

#### 6.4.4.2 WebSocket connection establishment

The relationship of RESPECT client and server per interface is given as follows:

- UNI (Between a RESPECT client (UE) and a RESPECT server (WSF)):
  * A RESPECT client (UE) is required to initiate and establish a WebSocket connection with a RESPECT server (WSF) according to IETF RFC 6455 [27].
  * Only one WebSocket connection is established between a RESPECT client (UE) and a RESPECT server (WSF). When activating multiple RESPECT clients on a UE, each RESPECT client (UE) needs to establish one WebSocket connection.
- NNI (Between RESPECT endpoints (IWFs)):
  * WebSocket connection is required to be established between IWFs according to IETF RFC 6455 [27].
  * How to establish and keep the WebSocket connection(s) and how may WebSocket connections are required are determined based on inter-operator agreement.
- Internal interface in the operator network (Between RESPECT endpoints in the operator network):
  * WebSocket connection is required to be established between RESPECT endpoints in the operator network (i.e., between WSFs, between a WSF and an IWF) according to IETF RFC 6455 [27].
  * How to establish and keep the WebSocket connection are determined based on operator policy.

A RESPECT endpoint is allowed to send a RESPECT request on a WebSocket connection, regardless of whether the RESPECT endpoint initiated the WebSocket connection or not. A RESPECT endpoint is required to send a RESPECT response on the WebSocket connection where a RESPECT request was received.

A RESPECT endpoint is allowed to send multiple requests for different purpose (e.g., establish/modify different media session, use different service) in parallel on the single WebSocket connection.

If the RESPECT client re-establishes the WebSocket connection with the RESPECT server after the unexpected closure of WebSocket connection, the RESPECT client is required to process the WebSocket connection establishment procedure according to the closure reason, the information received from RESPECT server.

#### 6.4.4.3 WebSocket connection keep alive

The Ping frame and Pong frame specified in IETF RFC 6455 [27] are used for WebSocket connection keep alive. The RESPECT endpoint is required to support Ping frame and Pong frame.

The RESPECT server is required to send a Ping frame to the WebSocket client on the WebSocket connection. Upon receipt of a Ping frame, the RESPECT client is required to immediately send a pong frame to the RESPECT server which sent the ping frame.

The RESPECT client (UE) is allowed to send a Ping frame to the RESPECT server on the WebSocket connection. To prevent congestion in the network, the interval of sending Ping frame is required to be greater than 10 seconds.

#### 6.4.4.4 WebSocket connection closure

When a RESPECT client detects the failure of sending/receiving signalling message to/from the other RESPECT server (e.g., due to loss of IP connectivity), the RESPECT endpoint needs to close the WebSocket connection by sending a Close frame according to IETF RFC 6455 [27].

When a RESPECT server detects the following events, the RESPECT server needs to close the WebSocket connection according to IETF RFC 6455 [27].

- Expiration of authentication period

- Failure of application level keep alive
- Server internal error

#### 6.4.4.5 Sending a RESPECT message over WebSocket

When a RESPECT endpoint sending a request or response to another RESPECT endpoint, the RESPECT endpoint is required to send a request or response in a single data frame, in order to ease the parse of JSON as a RESPECT signalling message at the RESPECT endpoint.

#### 6.4.4.6 Error handling

##### 6.4.4.6.1 General

This clause describes error handling during the WebSocket connection set up.

##### 6.4.4.6.2 Protocol version error

When a RESPECT endpoint (WSF) does not support the protocol version specified in the WebSocket URI, the RESPECT endpoint (WSF) will respond to the HTTP GET request for WebSocket establishment from a RESPECT endpoint (UE) by sending an error response, such as 404 (Not Found) response, 30x response.

A RESPECT endpoint (UE) receiving the 30x response to the HTTP GET request for WebSocket establishment should retry the establishment by specifying the other protocol version into the WebSocket URI.

##### 6.4.4.6.3 Network congestion error

When receiving a 5xx response (e.g., 503 (Service Unavailable), 502 (Bad Gateway)) to the HTTP GET request for WebSocket establishment, the RESPECT endpoint (UE) is required to perform the following procedure since there is a possibility of congestion in the connecting network.

1) If the 5xx response contains a Retry-After header field, the RESPECT endpoint (UE) is required not to send a HTTP request in the period specified in a Retry-After header field.
2) If the 5xx response does not contain Retry-After header field, the RESPECT endpoint (UE) is recommended not to send a HTTP request in the random period greater than or equal to 0.4 times and less than 0.5 times of the expiration time used in the auth procedures of RESPECT protocol.

##### 6.4.4.6.4 Timeout error

When encountering the failure of UPGRADE on the established TCP connection in a specific period, the RESPECT endpoint (UE) is required to perform as if the 5xx response not containing a Retry-After header field was received.

### 6.4.5 RESPECT (signalling protocol)

#### 6.4.5.1 General

This clause describes the details of RESPECT specification.

AsyncAPI [63] could be used as Interface Definition Language (IDL) for the RESPECT protocol.

#### 6.4.5.2 Key features of the RESPECT protocol

##### 6.4.5.2.1 General

This clause describes key feature of the RESPECT protocol to aid with the readability of the RESPECT protocol specification. The details of protocol such as messages, procedures are described in the subsequent clauses.

The RESPECT is transaction-based signalling protocol. Each transaction consists of a request and a response to the request. For signalling format for RESPECT request and response message, the JavaScript Object Notation (JSON)

format is applied, and the RESPECT messages are exchanged over the control session established between two RESPECT endpoints. For the control session, WebSocket Secure is used in order securely to transport the RESPECT messages. Over the control session, RESPECT transactions are performed for authentication, establishment of media session, getting information from the RTC network.

As key features of the RESPECT protocol, the following is described in the subsequent clauses.

1) Control session management
2) Media session management
3) Transaction management
4) Simplified mechanism on SDP offer/answer
5) Feature negotiation

#### 6.4.5.2.2 Control session management

The "control session" is a Secure WebSocket between two directory connected RESPECT endpoints managed with a state of authentication by using RESPECT protocol. Figure 6.4.5.2.2.1 shows lifetime of control session.

After the successful establishment of a WebSocket connection between two RESPECT endpoints, the status of the control session transits to "Unauth" status.

Upon a successful authentication by using auth mechanism of RESPECT protocol by a RESPECT client, the status of the control session transits to "Authed" status. In case where any authentication using RESPECT protocol, the status of the control session transits to "Authed" immediately after the successful establishment of a WebSocket connection.

NOTE 1: For example, two RESPECT endpoints in same operator network, the authentication can be achieved by implicit manner (e.g., adaptation of ACL control) instead of authentication using RESPECT protocol.

Upon a successful transition to "Authed" status for a control session, the RESPECT endpoints can exchange a RESPECT message over the control session for media session setup, update, release. When expired the authentication, the status of the control session transits to "Unauth" status.

NOTE 2: In the auth mechanism of the RESPECT protocol, the RESPECT client receives the expiration time for authentication from the RESPECT server.

When the WebSocket connection is disconnected, the status of the control session transits to "Terminated" status unless a feature to keep a state of a control session in a grace period is applied on that control session. If this feature is applied on a control session, a state of a control session transits to "Moratorium" state. In this state, a state of a control session is inherited if a new WebSocket connection between two RESPECT endpoints is established within a grace period.

For details related to control session, see the signalling procedures, message definitions and call flow example described in the subsequent clauses.

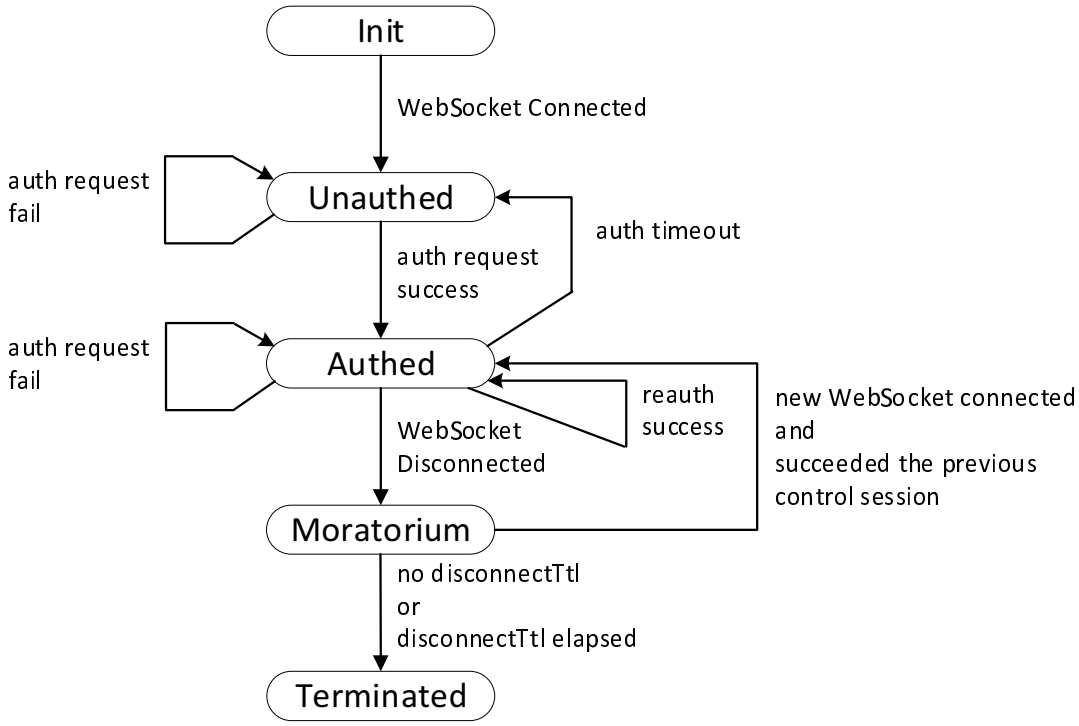


**Figure 6.4.5.2.2-1: Lifetime of control session**

### 6.4.5.2.3 Media session management

The "media session" is a concept for managing the media/data transported over U-Plane at C-Plane entities. The media/data setup by media session corresponds to the media/data processed in a "RTCPeerConnection" object defined in W3C WebRTC 1.0 [65].

This media session is identified at C-Plane entities by media session ID set in the signalling message. This media session ID is unique per each section on the signalling path and is generated by sender of media session setup. In the trapezoid model, three different media session ID is used on the signalling path for a media session; a media session ID between an originating UE and a WSF, a media session ID between two WSF and a media session ID between a WSF and a terminating UE.

The RESPECT endpoints in the RTC network manage the state of media session in the two perspectives; one is the state of whole media session, and the other is the state per media stream / STCP stream over data channel. To be more precise, these status is managed at a terminating RESPECT endpoint for a media session in the RTC network. The terminating RESPECT endpoints notifies these status of the adjacent RTC endpoints, and the adjacent RTC endpoints deliver these status to the proceeding or succeeding RTC endpoints. Therefore, all RTC endpoints involved in the media session can know the same status of the media session.

For the state of whole media session, the RESPECT endpoint in the RTC network manages "accepted", "connecting", "routed", "updateRequesting" and "updating" statuses. The meaning of these status is given as follows:

1) "accepted" indicates the media session setup request is reached to the terminating RESPECT endpoint in the RTC network, but the SDP offer/answer for the media session is not completed;

2) "connecting" indicates the media session setup request is reached to the terminating RESPECT endpoint in the RTC network and the SDP offer/answer for the media session is completed. The terminating RESPECT endpoint regards the SDP offer/answer is completed by sending a signalling message containing SDP answer.

3) "connected" indicates the U-Plane transport between a UE and the adjacent U-Plane entity in the RTC network is ready, but the media routing in the RTC network is not enabled;

4) "routed" indicates the U-Plane transport between a UE and the adjacent U-Plane entity in the RTC network is ready, but the media routing in the RTC network is enabled;

5) "updateRequesting" indicates the subsequent SDP offer/answer for the media session is not completed after the "connected" or "routed" state;

6) "updating" indicates the subsequent SDP offer/answer for the media session is completed after the "connected" or "routed" state, but there is at least one media stream or one SCTP stream whose status is not "connected".

For the state per media stream / STCP stream over data channel, the RESPECT endpoint in the network manages "connected" and "routed" statuses. The meaning of these status is same as the status mentioned above.

For details related to media session, see the signalling procedures, message definitions and call flow example described in the subsequent clauses.

#### 6.4.5.2.4 Transaction management

The RESPECT is transaction-based protocol. The transaction consists of a single request and a response corresponding to the request over the authenticated control session.

The pair of a request and a response in a transaction is identified by transaction ID. Transaction ID is generated by sender of a new request and the ID is set into the request. The same transaction ID is set into the response corresponding to the request. The transaction Id is unique on a control session between two RESPECT endpoints, and the transaction ID is managed at each RESPECT endpoint.

For media session control, the transactions for media session setup, update, disconnection are performed for a specific media session ID over a control session.

##### 6.4.5.2.4.1 Transaction timeout

The RESPECT defines transaction timer to comply with RTC service requirements.

For example, if a RESPECT endpoint in the network sends a media session setup request to a RESPECT endpoint of a UE acting as RESPECT endpoint client but no response is received for a long time, the network may not be able to a RTC service requirement due to this waiting time. Furthermore, the RESPECT endpoints over a signalling path for the media session needs to maintain server resource; this could be a security risk.

For transaction timer, the RESPECT defines the two timer; "T1" and "T2". T1 timer is a transaction timeout timer. A sender of a request needs to set T1 timer immediately after sending the request. Before expiration of T1 timer, the sender of the request needs to be prepared to receive a response to the request. Upon expiration of T1 timer, the sender of the request recognizes the timeout of the request and maintain transaction state before the expiration of "T2" timer.

In case the sender of the request is an intermediator of the request, the intermediator needs to send an error response indicating transaction timeout to the succeeding RESPECT endpoint. After sending the error response, if the intermediator receives;

- an error response from a proceeding RESPECT endpoint, then the intermediator discards the response;
- a successful response from a proceeding RESPECT endpoint, then the intermediator needs to disconnect the media session, if the request is for media session setup and update.

T2 timer is a timer for maintaining transaction state and T2 timer is longer than T1. A sender of a request needs to set T2 timer immediately after sending the request. Before expiration of T2 timer, the sender of the request needs to maintain the transaction state. Upon expiration of T2 timer, the sender of the request needs to free the transaction state.

Figure 6.4.5.2.4.1-1 shows a simple flow when a destination of a request does not respond to a request.

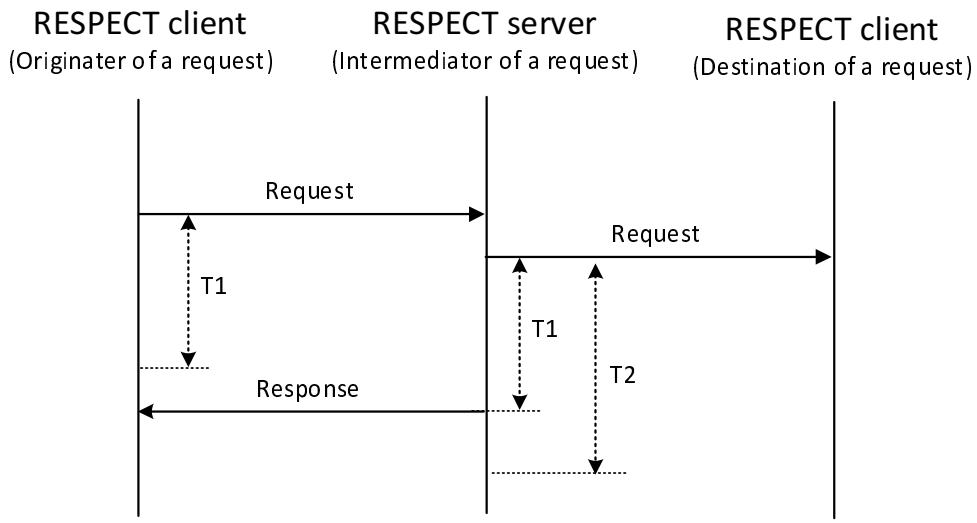


**Figure 6.4.5.2.4.1-1: A destination of a request does not respond**

For the value of T1 timer, round trip time of a transaction (2T) - time from sending a request at an originator of a request to receiving a response to the request at the originator of the request, is applied as a concept. This value may depend on the used access network, the number of intermediators, etc. For simplicity, the RESPECT protocol defines 10 seconds for the value of T1 timer.

For the value of T2 timer, 3T is applied considering the time for receiving an error response indicating transaction timeout from an intermediator to an originator of a request. The RESPECT protocol defined 15 seconds for the value of T2 timer.

The value of "T1" and "T2" is common for all RESPECT endpoint on a signalling path; therefore, an originator of a request is the first to recognize timeout.

#### 6.4.5.2.4.2 Retransmission

The RESPECT does not currently define a retransmission of a signalling message since a signalling message is transported over the reliable transport.

However, if a RESPECT endpoint receives a request which has a transaction ID previously received, the request is processed as if the re-transmission of the request was received, and the request is ignored at the RESPECT endpoint.

The same principle is applied to a response.

### 6.4.5.2.5 Simplified mechanism on SDP offer/answer

In the RESPECT protocol, the Session Description Protocol (SDP) defined in IETF RFC 8866 [53] is applied to negotiate media/data used, as it is required as WebRTC application as defined in W3C WebRTC 1.0 [65].

The RESPECT protocol explicitly specifies the type of SDP message (e.g., offer, answer, info) in the signalling message, so that a receiver of an SDP can determine what the receiver performs in an explicit manner, unlike SDP message transported over SIP. This information would be helpful for the processing of SDP at the RESPECT endpoint and for trouble shooting.

As types of SDP, the RESPECT protocol defines four types.

1) The "offer" indicates an SDP offer.

2) The "answer" indicates an SDP answer.

3) The "preOffer" indicates a tentative SDP offer. The "preOffer" type of SDP is allowed to be used only at an originating UE acting as RESPECT client, and how to handle this type of SDP is determined by a RESPECT endpoint in the network. For example, the RESPECT endpoint in the network will handle it as "offer" if this type of SDP is to be sent to a terminating UE acting as RESPECT client. Also, the RESPECT endpoint in the network will discard it and send "offer" type of SDP towards the originating UE if this type of SDP is to be sent to a

RESPECT endpoint in the network serving a VR space. Since this type of SDP is not "offer" nor "answer", the originating UE acting as RESPECT client is required not to set local description by using setLocalDescription() method defined in W3C WebRTC 1.0 [65].

4) The "info" indicates that this is information and not "offer" nor "answer".

As an SDP offer / answer, the RESPECT protocol adopts the following principle:

- Initial SDP offer and corresponding initial SDP answer containing a session description and media descriptions is transported in a signalling message. The SDP is formatted in an array and each description is formatted in a object of the array.

- As for the subsequent SDP offer and corresponding SDP answer, only the description(s) added or modified from the previously negotiated is transported in a signalling message. The SDP is constructed from the part of description(s) at both ends on the signalling path who need to handle the SDP offer/answer.

The background on adoptation of the above principle is as follows.

- In the WebRTC world, the multi-party real-time communication (e.g., VR space, online conference) are basically privided rather than peer-to-peer communication. Therefore, if an SFU is applied in the media function of the network, a signalling server (here, we call it WSF) needs to be send a new SDP offer to all the participants joining in s room, after generating an SDP offer reflecting the media descriptions for joining participant and leaving participant. As a result, both UEs and signalling server in a network need to handle huge number of "m=" line per paticipant.

- To detect change on the received subsequent SDP offer, the answerer needs to check if session-version in the "o=" line of the received SDP is changed from the previously negociaed SDP, then the answerer needs to check the modified descriptions by inspecting all the line of SDP. This process is realistic in a small SDP (like regacy voice call), but difficult in a big SDP containing fuge number of media descriptions.

- Therefore, the above principle for subsequest SDP offer/answer is useful for reducing the computing resource at the RTC endpoints.

For details related to simplified mechanism on SDP offer/answer and extension of SDP metadata, see the signalling procedures, message definitions and call flow example described in the subsequent clauses.

#### 6.4.5.2.7 Feature negotiation

The RESPECT protocol provides feature negotiation mechanism in a request-response exchange.

A RESPECT endpoint is allowed to indicate the supported feature(s) information to other RESPECT endpoint in a request or a response.

A RESPECT endpoint is allowed to require the use of the feature(s) for the service by including the required feature(s) information in the request. The RESPECT endpoint which receives the request requiring the use of the feature is allowed to accept or reject the required feature(s). If the RESPECT endpoint rejects the required feature(s) due to non-support of the feature, the RESPECT endpoint is required to indicate the feature(s) which is not supported by the RESPECT endpoint in the response corresponding to the received request.

A RESPECT endpoint is allowed to require the use of a feature(s) for the service by including the required feature(s) information in the response. A RESPECT endpoint is not allowed to require the feature(s) which is not indicated in the supported feature information of the corresponding request.

### 6.4.5.3 Protocol usage on UNI/NNI

#### 6.4.5.3.1 General

RESPECT uses control session and media session for media session control and management as described in clause 6.4.5.2.2 and clause 6.4.5.2.3. This clause describes the usage of these sessions at the UNI and NNI.

#### 6.4.5.3.2 UNI

The RESPECT client (UE) establishes only one control session with the RESPECT server (network). For the control session, Secure WebSocket are used as an underlying transport protocol. Every signalling message is terminated at the network. Then, no signalling messages are directly sent from UE to the RESPECT endpoint other than that UE connected to.

At the UNI, the network authenticates the UE connected to the network. Multiple UEs are able to use the single RTC user ID to connect with the network (however, operators are able to restrict the number of UEs allowed to be connected to the network based on the service policy). The authentication of the UE is performed on the control session.

Once the authentication is successfully completed, the UE and the network are able to send a media session setup request to the other. If the request is successfully proceeded, the media session is established over the UNI. Multiple media session is allowed to be setup using the single control session (however, operators are able to restrict the number of media sessions per control session, based on the service policy). Figure 6.4.5.3.2-1 shows the image of control sessions and media sessions over the UNI.

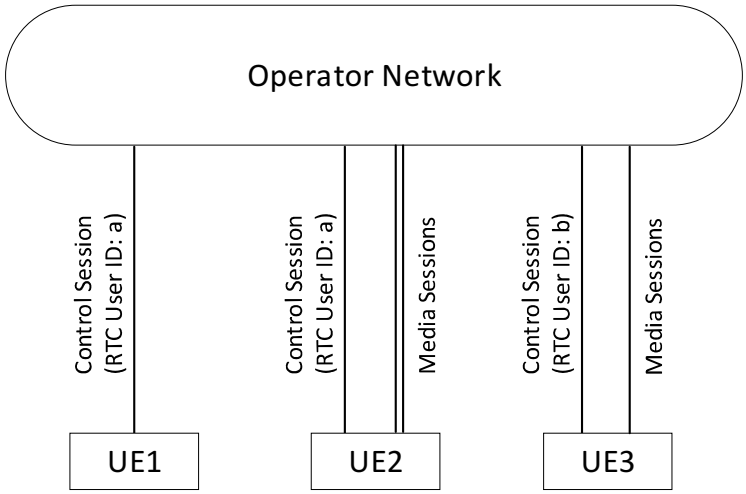


**Figure 6.4.5.3.2-1: Control Sessions and Media Session over UNI**

#### 6.4.5.3.3 NNI

At the NNI, the RESPECT client (in the other operator or SP network) is allowed to establish one or more control sessions with the RESPECT server (in the operator network). The operator is able to restrict the number of control session simultaneously connected to the operator network based on the bi-lateral agreement. For the control session, Secure WebSocket is used as an underlying transport protocol.

The control session of the UNI and the control session of the NNI do not have one-to-one correspondence.

At the NNI, there are cases that the RESPECT server in the operator network authenticates the RESPECT client of the other networks (the other operator or SP). This procedure for the authentication is same as UNI, however, the authentication could be skipped as mentioned in clause 6.4.5.2.2.

Multiple media session is allowed to be setup using the single control session (however, the operator of the RESPECT server is able to restrict such as the number of media sessions simultaneously established and/or bandwidth, based on the bi-lateral agreements). Figure 6.4.5.3.3-1 shows the image of control sessions and media sessions over the NNI.

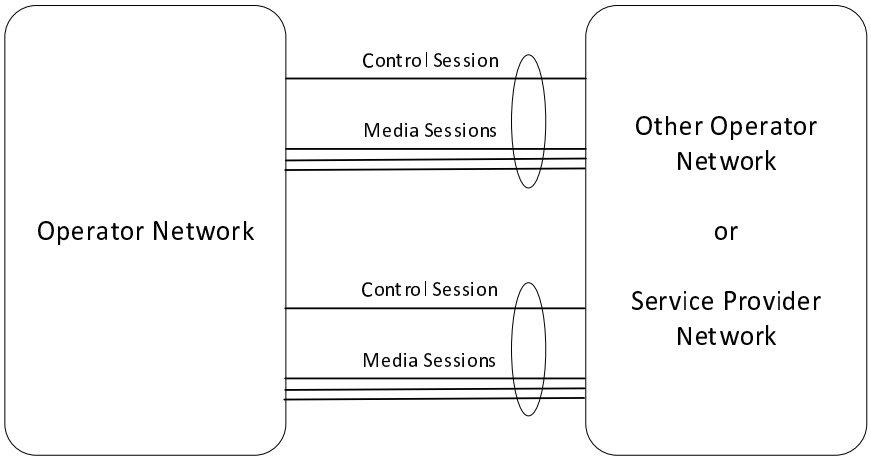


**Figure 6.4.5.3.3-1: Control Sessions and Media Session over NNI**

### 6.4.5.4 Protocol and version identification

The protocol name and the protocol version are required to be included in the WebSocket URI path as specified in 3GPP TS 26.113 [10] as follows:

{protocolRoot}/<protocolName>/<protocolVersion>

The WebSocket URI of the present version of RESPECT is required be set as follows:

- <protocolName> is set to "3gpp-respect"
- <protocolVersion> is set to "v1"

NOTE 1: {protocolRoot} is set as specified in 3GPP TS 26.113 [10] (i.e., "wss" schema is used).

The present version of RESPECT, the Sec-WebSocket-Protocol header field with "3gpp-respect.v1" subprotocol identifier is required to be included in the HTTP upgrade request.

NOTE 2: IANA registration is required for new subprotocol identifier in normative phase.

### 6.4.5.5 RESPECT messages

#### 6.4.5.5.1 General

RESPECT is defined as a text-based protocol and use the UTF-8 charset (IETF RFC 3629 [24A]). Each line in a RESPECT message is required to be terminated by carriage-return line-feed sequence (CRLF). The JavaScript Object Notation (JSON) format described in IETF RFC 8259 [36] is required to be used for encoding/decoding a payload of RESPECT message. Then default content type is required to be "application/json".

#### 6.4.5.5.2 Signalling message definition

##### 6.4.5.5.2.1 General

The RESPECT message is either a request or a response to the request. Request is a RESPECT message sent from a RESPECT endpoint to the other RESPECT endpoint, for the purpose of invoking a particular operation corresponding to an indicated method. Response is a RESPECT message sent from a RESPECT endpoint to the RESPECT endpoint which sent the request triggering the response, for indicating the result of the corresponding request prosessing. Both RESPECT client a RESPECT server is allowed to send requests.

All RESPECT message is required to include following information elements as the first level key of the message. These keys are specified as common key.

- Message type ("msgType")

- Method type ("method")
- Transaction ID ("transactionId")

A RESPECT message is allowed to include individual keys described in clause 6.4.5.5.4.3.

A RESPECT message is allowed to include application specific keys which are not specified for RESPECT. Application specific keys are required to be used as described in clause 6.4.5.5.4.4.

A RESPECT endpoint is allowed to ignore any keys which are not used by the RESPECT endpoint, unless the key is required to be processed.

The length of key name is required to be less than/equal to 64 octets. The keys are allowed to appear in any order in RESPECT message. The string of key name and value in RESPECT messages are required to be case-sensitive unless the key name and value are specified in IETF RFC 8259 [36] or RESPECT specification.

NOTE: The purpose of supporting application specific key is to enable RESPECT endpoint to use application specific capability between directly connected RESPECT endpoints. Application specific key is not intended for end-to-end operation.

#### 6.4.5.5.2.2 Request

Message type ("msgType") key of the message is set to "request".

If the RESPECT endpoint is pending status (i.e., the RESPECT endpoint has not received a response to the request which the RESPECT endpoint sent, the RESPECT endpoint is allowed to send new request (not resending of the request in pending status). For UNI, the maximum number of requests in pending status is specified by the operator policy. For NNI, the maximum number of requests in pending status is specified by the bilateral agreement between operators.

A pair of request and response which has same transaction ID ("transactionId") is called transaction.

#### 6.4.5.5.2.3 Response

Message type ("msgType") key of the message is set to "response". Method type ("method") key and Transaction ID ("transactionId") key are set to same value of the keys in the corresponding request.

Result of the request processing ("success") key is required to be set into the response. When the Result of the request processing ("success") key is set to "false", Error details ("problemDetails") key is required to be set into the response. When "success" key is set to "true", the response is called successful response. When "success" key is set to "false", the response is called error response.

The RESPECT is not required to send responses in sequential order of receiving request. The RESPECT endpoint which sent requests is required to be able to receive response in any order.

A single response is required to be sent to a single request. If the RESPECT endpoint receives multiple response which has same Transaction ID ("transactionId") key to the request, the RESPECT endpoint is required to process only the first response.

If the Transaction ID ("transactionId") of the received response is not match with the request in pending status, the response is required to be ignored.

### 6.4.5.5.3 Supported methods

#### 6.4.5.5.3.1 General

RESPECT supports the methods as shown in Table 6.4.5.5.3.1-1. Each method consists of a single request and a response to the request. RESPECT endpoint is required to support all methods described in Table 6.4.5.5.3.1-1.

**Table 6.4.5.5.3.1-1: Supported methods description**

| Method | Description |
|---|---|
| auth | A method for requesting authentication. |
| msetup | A method for requesting initiation of media session set up. |
| mupdate | A method for requesting media session status update (modification). |
| mdisc | A method for requesting media session disconnection. |
| getinfo | A method for retreaving information. |

#### 6.4.5.5.3.2 Authentication method ("auth")

Authentication method ("auth") is used by RESPECT client (UE) to request authentication of the RESPECT client (UE) to the RESPECT server (WSF) over the UNI.

When the authentication request is successfully processed, RESPECT client (UE) is authenticated and the binding information (the combination of RTC user ID and control session) is registered to the database in the network.

Applicable keys in the request / response of Authentication method ("auth") are described in clause D.1.1.

#### 6.4.5.5.3.3 Media session set up method ("msetup")

Media session set up method ("msetup") is used for requesting media session set up.

A RESPECT client (UE) which sends Media session set up request over the UNI is called originating RTC UE. A RESPECT client (UE) which receives Media session set up request over the UNI is called terminating RTC UE. A network which sends Media session set up request over the NNI as a RESPECT endpoint is called as originating RTC network. A network which receives Media session set up request over the NNI as a RESPECT endpoint is called as terminating RTC network.

Applicable keys in the request / response of Media session set up method ("msetup") are described in clause D.1.2.

#### 6.4.5.5.3.4 Media session update method ("mupdate")

Media session update method ("mupdate") is used for requesting modification of an existing media session.

The RESPECT endpoint which receives Media session update ("mupdate") request is required to check the received keys which need to be processed for the requested modification. The following keys are used for modification of the media session.

- Offer/answer ("offerDesc"/"answerDesc")
- Media session state ("mediaSessionState")
- Originating ID ("preferredOid" / "AssertedOid") / Terminating ID ("preferredTid" / "assertedTid")

Application specific key is also applicable for the target of Media session update method ("mupdate").

If the RESPECT endpoint sent/received Media session setup request ("msetup") or Media session update request ("mupdate") and the RESPECT endpoint has not received/sent a response to the corresponding request, the media session (identified by Media session ID) is in the pending status. The RESPECT endpoint is not allowed to send a new Media session update request for the media session in pending status. If, a race condition is caused by crossover of requests, the RESPECT endpoint is required to ignore the Media session update request ("mupdate") and send error response to the request.

If the RESPECT endpoint receives an error response to the Media session update request ("mupdate"), the media session update is failed and the existing media session is not modified.

Applicable keys in the request / response of Media session update method ("mupdate") are described in clause D.1.3.

#### 6.4.5.5.3.5 Media session disconnect method ("mdisc")

Media session disconnect method ("mdisc") is used for requesting release of the media session.

The RESPECT endpoint is allowed to send Media session disconnect request ("mdisc") to the media session in pending status, if Media session ID ("mediaSessionId") is assigned for the media session. The followings are example situation.

- The RTC operator network is allowed to send a Media session disconnect request ("mdisc") for media session disconnection when the RTC operator network sent a Media session set up request ("msetup") or Media session update request ("mupdate") including Media session ID ("mediaSessionId") key to a RESPECT endpoint and has not received a response to the request from the RESPECT endpoint.

- The RESPECT client (UE) is allowed to send a Media session disconnect request ("mdisc") for media session disconnection when the RESPECT endpoint sent a Media session update request ("mupdate") including Media session ID ("mediaSessionId") key to a RESPECT server (WSF) and has not received a response to the request from the RESPECT server (WSF).

Applicable keys in the request / response of Media session disconnect method ("mdisc") are described in clause D.1.4.

#### 6.4.5.5.3.6 Information query method ("getinfo")

Information query method ("getinfo") is used for getting information (e.g., STUN/TURN server address) from the operator network. The queried information is indicated by Requested information list ("resourcesReq") key.

This method provides the alternative to the information queries on RTC-5 interface.

Applicable keys in the request / response of Information query method ("getinfo") are described in clause D.1.6.

### 6.4.5.5.4 Keys (information elements) included in RESPECT messages

#### 6.4.5.5.4.1 General

This clause defines the information elements included in a signalling message as a JSON key. Information Elements are categorised as following key types.

- Common key
  An information element which is required to be set into all RESPECT messages.

- Individual key
  An information element which is required to be set into a RESPECT message based on the individual requirement (e.g., type of message, type of method, selected capability, etc.).

- Application specific key
  An information element which is specific to an application. RESPECT allows to use application specific method and application specific keys for flexibility.

NOTE: As a convention, data types in the present specification are written with an upper-case letter in the beginning. Parameters are written with a lower-case letter in the beginning. As an exception, data types that are also defined in AsyncAPI [63] can use a lower-case case letter in the beginning for consistency.

#### 6.4.5.5.4.2 Common key

##### 6.4.5.5.4.2.1 General

Common key is an information element which is required to be set into all RESPECT messages. This clause describes common keys.

##### 6.4.5.5.4.2.2 Message type ("msgType")

This key indicates the message type of RESPECT massage. If the message is request, then "msgType" key is set to "request". Otherwise (i.e., the message is response), "msgType" key is set to "response".

The data type of "msgType" key is "enum". The applicable values are "request" or "response". If the RESPECT endpoint receives a RESPECT message includes other value, the RESPECT endpoint is required to discord the received message.

#### 6.4.5.5.4.2.3 Method type ("method")

This key indicates the method type of RESPECT message. The method supported by RESPECT is defined in Table 6.4.5.5.3.1-1 of this document.

The data type of "method" key is "string". The value is required to be set to the value of "Method" column of Table 6.4.5.5.3.1-1.

The supported methods are possibly extended in future releases. However, the method of RESPECT message is not allowed to include "." in method name.

RESPECT allows to send application specific methods for immersive RTC applications. The application specific method name is required to include reverse order of the internet domain which owned by the operator or the SP who provide the RTC service, before the application specific method (e.g., com.example.specificMethod). This rule enables to avoid confliction of method name.

#### 6.4.5.5.4.2.4 Transaction ID ("transactionId")

This key indicates the transaction ID of the RESPECT message. The pair of a request and a response corresponding to the request is identified as transaction by transaction identifier (ID).

The RESPECT endpoint which sends new request is required to generate a transaction ID and set the transaction ID to the "transactionId" key of the request.

The data type of "transactionId" key is required to be 64-bit unsigned integer and unique for all transactions on the WebSocket connection. Transaction ID is not allowed to be reused for another transaction on the WebSocket connection. To avoid collision of Transaction ID among different transactions, RESPECT endpoint is required to generate a Transaction ID as follows:

i) at the UNI

- the RESPECT client (UE) is required to generate even-numbered transaction ID. Transaction ID for an initial request on the control session is required to be set to "0".
- the RESPECT server (WSF in the network) is required to generate odd-numbered transaction ID. Transaction ID for an initial request on a control session is required to be set to "1".

ii) at the NNI

- the RESPECT client is required to generate even-numbered transaction ID. Transaction ID for an initial request on the control session is required to be set to "0".
- the RESPECT server is required to generate odd-numbered transaction ID. Transaction ID for an initial request on a control session is required to be set to "1".

iii) Transaction ID is incremented by "2", when the transaction ID is issued for a new transaction on the WebSocket connection.

iv) If the bit-field of transaction ID crosses 64-bit boundary, the value is wraparound to initial value (i.e., "0" or "1")

### 6.4.5.5.4.3 Individual key

#### 6.4.5.5.4.3.1 General

Individual key is an information element which is required to be set into a RESPECT message based on the individual requirement (e.g., type of message, type of method, selected capability, etc.). This clause describes individual keys.

#### 6.4.5.5.4.3.2 Result of the request processing ("success")

This key indicates the result of the request processing. The data type of "success" key is Boolean. When the request is successfully processed, the "success" key is set to "true", otherwise, set to "false". When "success" key is set to "true", the response is called successful response. When "success" key is set to "false", the response is called error response.

#### 6.4.5.5.4.3.3 Error details ("problemDetails")

This key indicates the detailed information of the failure reason.

Error responses are required to include "problemDetails" key which indicates the factor of the error. "mdisc" request is allowed to include "problemDetails" key to indicate the reason for disconnection of the session.

The data type of "problemDetails" key is "object" according to Problem Details JSON Object (IETF RFC 7807 [32A]). "type" sub-key is required to be set. The data type of "type" sub-key is "string" Other sub-keys are allowed to be set.

Details of the applicable string for "type" sub-key is described in clause 6.4.5.5.5.

#### 6.4.5.5.4.3.4 Required extensional capability ("requiredExtension")

This key indicates the extended feature(s) which is required to process the RESPECT message at the received RESPECT endpoint.

When this key is included in the request, the indicated feature is required to be applied for processing the request. When this key is included in the response, the indicated feature is required to be applied for processing the response. When the request includes the "requiredExtension" key, the response other than the error response which includes the "problemDetails" key set to Bad Extension is implicitly treated as the indicated feature in the corresponding request is required, even if the response does not include the "requiredExtension" key.

The data type of "requiredExtension" key is "array". Only "string" data type values indicating extended feature are allowed to be set into "requirdExtension" key. If the array of "requiredExtension" key does not has the element, "requiredExtension" key is not allowed to be set in the RESPECT message.

#### 6.4.5.5.4.3.5 Unsupported extensional capability ("unsupportedExtension")

This key indicates the extended feature(s) which is not supported by the RESPECT endpoint.

When the RESPECT endpoint receives the "requiredExtension" key including extended feature(s) which the RESPECT endpoint does not support, the RESPECT endpoint includes the "unsupportedExtension" key including unsupported extended feature(s) in the response corresponding to the request.

The data type of "unsupportedExtension" key is "array". Only "string" data type values indicating extended feature are allowed to be set into "unsupportedExtension" key. If the array of "unsupportedExtension" key does not has the element, "unsupportedExtension" key is not allowed to be set in the RESPECT message.

#### 6.4.5.5.4.3.6 Supported extensional capability ("supportedExtension")

This key indicates the extended feature(s) which is supported by the RESPECT endpoint.

The data type of "supportedExtension" key is "array". Only "string" data type values indicating extended feature are allowed to be set into "supportedExtension" key. If the array of "supportedExtension" key does not has the element, "supportedExtension" key is not allowed to be set in the RESPECT message.

#### 6.4.5.5.4.3.7 Retry restriction timer ("retryAfter")

RESPECT endpoints in the operator network at the UNI, RESPECT endpoints in the operator network and RESPECT endpoints SP network at the NNI are allowed to include the "Retry-After" key in the top of the error response which includes the "problemDetails" key set to Bad Gateway, Service Unavailable or Server Time-out. The value of the "Retry-After" indicates how long the RESPECT endpoints is required to wait before sending a RESPECT message. The RESPECT endpoint which received the response which include "Retry-After" is not allowed to send any new RESPECT message on the control session (excluding a pending response to a received request.).

AT the UNI, RESPECT endpoint (on UEs) is not allowed to send the "Retry-After" key in the response.

The data type of "Retry-After" key is 32-bit "number". The value means seconds.

#### 6.4.5.5.4.3.8 Target of redirection ("location")

This key indicates the preferred target resource for redirecting the request.

When RESPECT endpoints (in the operator network) send an error response which includes the "problemDetails" key set to 307 (Temporary Redirect) or 308 (Permanent Redirect), the response is required to include "location" key.

The data type of "location" key is "string". The value target RTC resource ID for redirecting the request.

#### 6.4.5.5.4.3.9 RTC user ID ("rtdUserId")

RTC user ID is set into this key.

The data type of "rtcUserId" key is "string". The format of the value is URI format defined in IETF RFC 3986 [24B].

The URI scheme is "3gpp-respect-v1" for RESPECT version 1. The host part is set to the internet domain which owned by the operator or the SP who provide the RTC service to the user.

If this key is set to "auth" request, the RESPECT client set the RTC user ID which is used to authentication of the RESPECT client.

#### 6.4.5.5.4.3.10 Authentication type ("authType")

This key indicates the type of authentication in the "auth" request.

The data type of "authType" key is "string". The value is required to be set to one of "Basic", "Digest" and "Bearer", according to HTTP "auth-scheme" (IETF RFC 9110 [54]). The value is case-insensitive.

NOTE: The values are possibly extended in future releases.

#### 6.4.5.5.4.3.11 Authentication information ("authorization")

This key indicates the authentication information in the "auth" request.

The data type of "authorization" key is "string". The value is required to be set according to HTTP "credential" (IETF RFC 9110 [54]). The value is case-insensitive.

Therefore, "auth-scheme" is set to the same value of "authType" key in the request and the following "token68" or "auth-params" is set to the value according to the type of authentication of "authType" key (e.g., JWT).

#### 6.4.5.5.4.3.12 Authentication and media session retention timer ("disconnectTtl")

This key indicates the duration which the network keeps the authentication status and media session state related to the RESPECT endpoints, if the WebSocket connection is disconnected.

The data type of "disconnectTtl" key is "number" and the value is unsigned 32-bit integer.

The RESPECT endpoint (UE) is allowed to include this key in the "auth" request to indicate requesting use of the Authentication and media session retention timer. The value is set to the duration (seconds) which the RESPECT endpoint (UE) wished to keep the authentication and media session, when the WebSocket connection is disconnected.

The RESPECT endpoint (AS) is required to include the applied duration in the response to the "auth" request. If the requested value in the "auth" request exceeds the maximum duration of the operator policy, the RESPECT endpoint (AS) include the maximum value to the "disconnectTtl" key in the response to the "auth" request. If "0" is set to the "disconnectTtl" key in the response to the "auth" request, it indicates that Authentication and media session retention timer is not applied.

#### 6.4.5.5.4.3.13 Credential for authentication restoration ("webrtcReauthCredential")

This key indicates the credential for re-authentication when the WebSocket connection is disconnected unexpectedly. The data type of "webrtcReauthCredential" key is "string". The value is the token issued by the RESPECT endpoint (AS).

The RESPECT endpoint (UE) receives the "webrtcReauthCredential" key in the "auth" response, if the authentication restoration is applicable. The RESPECT endpoint (UE) is allowed to include the "webrtcReauthCredential" key in the "auth" request for re-authentication after the re-connection of WebSocket connection. If the re-authentication is successfully proceeded, the control session state before the WebSocket disconnection is restored for the new control session.

#### 6.4.5.5.4.3.14 Authentication challenge ("wwwAuthenticate")

This key is used for indicating the information related to authentication from the RESPECT server to the RESPECT client.

In the case of Digest authentication (i.e., "Digest" is set to "authType" key of the "auth" reqeust), RESPECT server is required to request the RESPECT client to send the "auth" request again, based on the authentication information provided by eiRTCW server. This key is included in the 401 error response for indicating authentication information from RESPECT server to RESPECT client.

The data type of "wwwAuthenticate" key is "object". The value is encoded according to HTTP WWW-Authenticate header field (IETF RFC 9110 [54]). The "authScheme" sub-key of "wwwAuthenticate" key is corresponding to "scheme" of WWW-Authenticate header field.

#### 6.4.5.5.4.3.15 Duration of the authentication ("expires")

This key indicates the expiration time duration that the RESPECT client is authenticated. When the timer is expired, the RESPECT client is de-authenticated and transitions to unauthenticated status. To keep being authenticated, the RESPECT client is required to send "auth" request and be re-authenticated during the indicated expiration time duration in the "expires" key.

This key is set into the "auth" response when the authentication is successfully processed.

The data type of "expires" key is "number" and the value is unsigned 64-bit integer. The value means seconds.

#### 6.4.5.5.4.3.16 Destination ID ("dId")

This key indicates a destination of a media session setup request. Destination ID ("dId") key is required to be included in the "msetup" request and is allowed to be included in the "mupdate" request. The data type of the "dId" key is "object".

For the "dId" key, any one of the following fields is allowed to be set. Regarding "ds" field, only a originating RESPECT endpoint (UE) is allowed to be set in the "msetup" request towards an RTC network.

- "uri"
- "tn"
- "ds"

The data type of "uri", "tn" and "ds" fields are "string". The value of each fields is set as follows.

- "uri": either RTC resource ID or RTC user ID
- "tn": telephone number (global number digits excluding "+")
- "ds": dialstring

The applicable URI scheme for RTC resource ID is "3gpp-respect-v1" for native RTC resources. The host part is set to the internet domain which owned by an operator or an SP who provides the RTC service to the user. The userinfo part is assigned by the operator or the SP who provides the RTC service to the user.

#### 6.4.5.5.4.3.17 Media session ID ("mediaSessionId")

This key indicates the media session ID used for identification of a media session over a control session. This key is always set in the RESPECT message related to media session (i.e., request and response of "msetup", "mupdate" and "mdisc"). Uniqueness of the media session ID is guaranteed per control session where the media session is established.

The data type of this key is "string". The length of this key is required to be less than/equal to 128 octets.

When sending an "msetup" request over an established control session, the RESPECT endpoints need to newly generates the media session ID and set the generated media session ID into "mediaSessionId" key.

When receiving the "msetup" request for the media session, the RESPECT endpoints need to set the same media session ID as with that of the request into "mediaSessionId" key of an "msetup" response.

When sending a subsequent RESPECT request (i.e., "mupdate" and "mdisc" request), the RESPECT endpoints need to set the media session ID which is targeted media session of the request.

When receiving the subsequent RESPECT request for the media session, the RESPECT endpoints need to set the same media session ID as with that of the request into "mediaSessionId" key of a response.

#### 6.4.5.5.4.3.18 Media session state ("mediaSessionState")

This key indicates the overall state of the media session (identified by the "mediaSessionId" key).

The terminating RESPECT endpoint (AS) is required to manage the state of the media session state and indicate the state to other RESPECT endpoints of the media session.

Terminating RESPECT endpoint (AS) refers to following RESPECT endpoints based on the destination indicated by the "dId" key:

- "dId" key is set to RTC resource ID:

  The WSF which the RTC resource ID is assigned to.

- "dId" key is set to RTC user ID:

  The WSF which the RESPECT endpoint (UE) is connected to.

- "dId" key is set to telephone number (i.e., IMS UE):

  The IWF which the IBCF of the IMS network where the IMS UE is registered to.

The data type of "mediaSessionState" key is "string". The value of "mediaSessionState" key is "enum" and only following values are applicable.

- "accepted"
- "connecting"
- "connected"
- "routed"
- "updating"

"accepted" indicates the media session is in following state.

- The "msetup" request is reached to the terminating RESPECT endpoint (AS) of the "dId" key.
- The offer/answer for the media session is not completed.

"connecting" indicates the media session is in following state.

- The "msetup" request is reached to the terminating RESPECT endpoint (AS) of the "dId" key.
- The offer/answer for the media session is completed (the state of the message which sends answer is treated as offer/answer is completed).
- At least one of the "connected" field of the "state" of the "mc" sub-key or "dc" subkey in the "mediaInfo" key is not set to "true".

"connected" indicates the media session is in following state.

- The offer/answer for the media session is completed (the state of the message which sends answer is treated as offer/answer is completed).
- All the "connected" field of the "state" of the "mc" sub-key or "dc" subkey in the "mediaInfo" key is set to "true".
- At least one of the "routed" field of the "state" of the "mc" sub-key or "dc" subkey in the "mediaInfo" key is not set to "true".

"routed" indicates the media session is in following state.

- The offer/answer for the media session is completed (the state of the message which sends answer is treated as offer/answer is completed).
- All the "connected" field of the "state" of the "mc" sub-key or "dc" subkey in the "mediaInfo" key is set to "true".
- All the "routed" field of the "state" of the "mc" sub-key or "dc" subkey in the "mediaInfo" key is set to "true".

"updateRequesting" indicates the media session is in following state.

- The offer/answer for the media session setup was completed.
- The offer/answer for the media session is not completed.

"updating" indicates the media session is in following state.

- The offer/answer for the media session setup was completed.
- The offer/answer for the media session update is completed (the state of the message which sends answer is treated as offer/answer is completed).
- At least one of the "connected" field of the "state" of the "mc" sub-key or "dc" subkey in the "mediaInfo" key is set to "true".

#### 6.4.5.5.4.3.19 Media Information ("mediaInfo")

This key indicates the following information related to the media session identified the media session ID.

- SDP description
- Metadata of the media channel and data channel

The data type of "mediaInfo" key is "object". The "mediaInfo" key has following 5 sub-keys.

1) "type"
2) "sdp"
3) "mc"
4) "dc"
5) "participantDesc"

The details of the above sub-keys are followings.

(1) "type" sub-key:

This sub-key indicates the requested treatment of the information in the "mediaInfo" key. The data type of the "type" key is "string" and only following values are applicable.

a) "offer":
This indicates the "mediaInfo" key includes an SDP description as an initial offer or a subsequent offer specified in IETF RFC 8829 [47].

b) "preOffer":
This indicates that the "mediaInfo" key includes an SDP description as a preoffer.

c) "answer":
This indicates that the "mediaInfo" key includes an SDP description as an initial answer or a subsequent answer specified in IETF RFC 8829 [47].

d) "info":
This indicates that the "mediaInfo" key includes media related metadata, however, does not include SDP description.

(2) "sdp" sub-key:

This sub-key indicates the SDP description and the label of the SDP description. The SDP description can be a complete SDP description or a part of SDP description. The data type of "sdp" sub-key is "object". The "sdp" sub-key has following sub-keys.

1) "part"

2) "label"

"part" sub-key includes SDP description. The data type of "part" sub-key is "array" and includes following fields.

- "index":
  This field includes the index number of the "lines" field. The data type of "index" field is unsinged 32-bit "number". Index number "0" indicate the "lines" is session-level section. Index number "1" and subsequent number indicates media description per" m=" line. The number is required to be incremented sequentially from "1" for media description.

- "lines":
  This field includes the session level section and the media description of the SDP description. The data type of "lines" field is "array[string]". Strings included in the array is the SDP description per line without CRLF.

  <example>

  "lines": ["v=0","o=...", "s=-", "c=...", "t=0 0"]

"label" sub-key includes the unique ID of the media session. The ID is expected to be used by APIs related to the medina handling. Then the ID included in "label" sub-key is required to be unique per RTC resource.

The data type of "label" sub-key is "string".

If a "sdp" sub-key is included in the RESPECT message sent from the network connected with CP and for initial offer/preoffer/answer, the "sdp" sub-key is required to include the "label". "label" sub-key is not allowed to included other cases.

NOTE 1: The label of the media session can be the corresponding media session ID. However, media session ID is unique per control session (i.e., If WSF use the label, the WSF needs to care about that the media session ID is different between control sessions).

(3) "mc" sub-key:

This sub-key indicates the metadate related to media channel. "The data type of "mc" sub-key is "object".

When the "part" sub-key includes the media description for media channel, the corresponding information is required to be included. This key is allowed to be included in the "mediaInfo" key, even if the corresponding "part" sub-key of "sdp" sub-key is not included in the "mediaInfo" key.

This key has the following sub-key.

1) "metadata"

"metadata" sub-key includes the information related to media channel. The data type of "metadata" sub-key is "array[object]" and includes following information.

- "index":
  This field indicates dependency with the corresponding media description described in the "part" sub-key of "sdp" sub-key. Then the index number of "index" field is set to the index number of the "index" field of "part" sub-key of "sdp" sub-key corresponding to the metadata. This field is not allowed to be used for media description for data channel. The data type of "index" field is unsinged 32-bit "number".

- "actType":
  This field indicates the intention of the media description specified by "index" key This field is not allowed to be

used for session description. The data type "actType" is "string" and only following values are applicable depends on the "type" sub-key of "mediaInfo" key.

<values for offer/preOffer>

- "add" : indicates the addition of the media channel or resume of inactive media channel.
- "del" : indicates the deletion of the media channel
- "mod" : indicates the modification of the media channel

<values for answer>

- "aly" : the requested offer (add/del/mod) is accepted by the answerer.
- "dcl" : the requested offer (add) is declined by the answerer.

NOTE 2: If the whole the offer is refused, sends error response to the "msetup"/"mupdate" request.

- "groupLabel":
  This field indicates the group label which expected to be used by APIs related to the media handling. The data type of "groupLabel" is string. The string is not allowed to include ".".
  The group label is used as category. Then there are multiple media descriptions which has same group label in the media session.
  The group label is deleted if the media channel is deleted (by setting "actType" to "del"). If the media channel is re-added, a group label different from the previous label is allowed to be assigned.

  This field is applicable for:

  - The metadata is media description for media channel.
  - The metadata is for offer (initial offer/subsequent offer/preoffer) which "actType" is set to "add".

- "label":
  This field indicates the unique ID of the media channel or session description. The ID for session description indicates the media session, not individual media channels.
  The data type of "label" sub-key is "string". The ID is expected to be used by APIs related to the medina handling. Then the ID included in "label" sub-key is required to be unique per RTC resource.
  If a "sdp" sub-key is included in the RESPECT message sent from the network connected with CP and "actType" sub-key is set to "add", the "label" sub-key is required to be included in the "metadata". "label" sub-key is not allowed to included other cases.
  The "label" is deleted when the media channel is deleted.

- "state":
  This field indicates the communication state of the media channel. Only the terminating RESPECT endpoint (AS) is allowed to set this field. The data type of this field is "object" and has following fields.

  - "connected":
    This field indicates whether the media channel is connected between UE and network (i.e., MF or TGF) or not. The data type of "connected" field is "boolean". "true" means the media channel is connected between UE and network or update of the media channel is successfully completed. "false" means the media channel is not yet connected or updated between UE and network. Note that the SRTP packet can be arrived at UE or network, even if the "connected" key is set to "false".

  - "routed":
    This field indicates whether the media routing is completed or not on the connected media channel. The data type of "routed" field is "boolean". "true" means the media routing is completed for all media channels related to the media session at the RTC resource or the UE. "false" means the media routing is not yet completed for all media channels.

- "handlingPref":
  This sub-key indicates the user preference for the media channel. This sub-key is allowed to be set only when the media description is for offer/preoffer and "actType" is set to "add"/"mod". The data type of "handlingPref" sub-key is "object" and has following fields.

- "index":
  This field indicates dependency with the corresponding media description described in the"part" sub-key of "sdp" sub-key. Then the index number of "index" field is set to the index number of the "index" field of "part" sub-key of "sdp" sub-key corresponding to the metadata. The data type of "index" field is unsinged 32-bit "number".

- "connectedToDevice":
  This field indicates the expected device which the media channel is connected to. The data type of "connectedToDevice" is "string" and only following values are applicable. The values other than follwoings are ignored.

  - "audioIn": Audio input device such as microphone.

  - "audioOut": Audio output device such as speaker or headphone.

  - "videoIn": Video input device such as camera

  - "display": Video output for specific area on the display

  - "videoOut": Video output for full screen on the display

- "preferredStyle":
  This field indicates the specific characteristics of the connected media. The data type of "preferredStyle" is "string" and following values are specified for the case "connectedToDevice" field is seto to "display".

  - "thumbnail"

  - "mainview"

  - "screenshare"

- "prticipandId":
  This field indicates the participant ID. The participant ID of the "participantDesc" sub-key of the "mediaInfo" key is included in the "participantId" field of the "handlingPref" sub-key, when the media is associated with the participant ID. The data type of "participandId" field is "string".

(4) "dc" sub-key:

This sub-key indicates the metadate related to data channel. "The data type of "dc" sub-key is "object".

When the "part" sub-key includes the media description for data channel, the corresponding information is included. This key is allowed to be included in the "mediaInfo" key, if the corresponding "part" sub-key of "sdp" sub-key is not included in the "mediaInfo" key.

This key has the following sub-key.

1) "sdpIndex"

2) "metadata"

"sdpIndex" sub-key indicates the dependency with the corresponding media description for data channel described in the "part" sub-key of "sdp" sub-key. The data type of "index" field is unsinged 32-bit "number"

"metadata" sub-key includes the information related to the data channel. The data type of "metadata" sub-key is "array[object]" and includes following information.

- "index":
  This field indicates stream identifier of the corresponding data stream. The data type of "index" field is unsinged 16-bit "number".

- "actType":
  This field indicates the handling of the data stream specified by "index" key. The data type "actType" is "string" and only following values are applicable depends on the "type" sub-key of "mediaInfo" key.

  <values for offer/preOffer>

- "add" : indicates the addition of the data stream to the data channel.
- "del" : indicates the deletion of the data stream from the data channel

<values for answer>

- "aly" : the requested offer (add/del) is accepted by the answerer.
- "dcl" : the requested offer (add) is declined by the answerer. As a result, the data stream is not added.

NOTE 2: If the requested handling is "del", "dcl" is not applicable for the media stream.

- "groupLabel":
  This field indicates the group label which expected to be used by APIs related to the data channel stream handling. The data type of "groupLabel" is string. The string is not allowed to include ".".
  The group label is used as category. Then there are multiple data channel stream which has same group label in the media session.
  The group label is deleted if the media channel is deleted (by setting "actType" to "del"). If the media channel is re-added, a group label different from the previous label is allowed to be assigned.

  This field is applicable for:

  - The metadata is media description for data channel.
  - The metadata is for offer (initial offer/subsequent offer/preoffer) which "actType" is set to "add".

- "label":
  This field indicates the unique ID of the data channel stream.
  The data type of "label" sub-key is "string". The ID is expected to be used by APIs related to the medina handling. Then the ID included in "label" sub-key is required to be unique per RTC resource.
  If a "sdp" sub-key is included in the RESPECT message sent from the network connected with CP and "actType" sub-key is set to "add", the "label" sub-key is required to be included in the "metadata". "label" sub-key is not allowed to included other cases.

- "state":
  This field indicates the communication state of the data channel stream. Only the terminating RESPECT endpoint (AS) is allowed to set this field. The data type of this field is "object" and has following fields.

  - "connected":
    This field indicates whether the data channel stream is connected between UE and network (i.e., MF or TGF) or not. The data type of "connected" field is "boolean". "true" means the data channel stream is connected between UE and network. "false" means the data channel stream is not yet connected between UE and network. Note that the SRTP packet can be arrived at UE or network, even if the "connected" key is set to "false".

  - "routed":
    This field indicates whether the data routing is completed or not for the connected data channel stream. The data type of "routed" field is "boolean". "true" means the data routing is completed for all data channel streams related to the media session at the RTC resource or the UE. "false" means the data routing is not yet completed for all data channel streams.

- "subprotocol":
  This field is equivalent to "a=dcmap subprotocol-opt" specified in IETF RFC 8864 [52]. The data type of "subprotocol" field is "string". This field is only applicable for the "mediaInfo" which the "type" sub-key is set to "offer"/"preoffer" and "actType" sub-key of the "dc" sub-key is set to "add".

- "ordered":
  This field is equivalent to "a=dcmap ordering-opt" specified in IETF RFC 8864 [52]. The data type of "ordered" field is "boolean". This field is only applicable for the "mediaInfo" which the "type" sub-key is set to "offer"/"preoffer" and "actType" sub-key of the "dc" sub-key is set to "add".

- "maxretr":
  This field is equivalent to "a=dcmap maxretr-opt" specified in IETF RFC 8864 [52]. The data type of "maxretr" field is unsigned 32-bit "number". This field is only applicable for the "mediaInfo" which the "type" sub-key is set to "offer"/"preoffer" and "actType" sub-key of the "dc" sub-key is set to "add".

- "maxtime":
  This field is equivalent to "a=dcmap maxtime-opt" specified in IETF RFC 8864 [52]. The data type of "maxtime" field is unsigned 32-bit "number". This field is only applicable for the "mediaInfo" which the "type" sub-key is set to "offer"/"preoffer" and "actType" sub-key of the "dc" sub-key is set to "add".

- "priority":
  This field is equivalent to "a=dcmap priority-opt" specified in IETF RFC 8864 [52]. The data type of "priority" field is unsigned 16-bit "number". This field is only applicable for the "mediaInfo" which the "type" sub-key is set to "offer"/"preoffer" and "actType" sub-key of the "dc" sub-key is set to "add".

(5) "participantDesc" sub-key:

This sub-key indicates the information of the participant in the media session. The included information is deference of the participant information from previous status. "participantDesc" key is allowed to be included in the when the SDP information is not included in the "mediaInfo" key. In that case, the "type" of the "mediaInfo" key is set to "info". This key is allowed to be included only for the RESPECT message is sent from a network. The data type of "participantDesc" sub-key is "array[object]" and includes following information.

- "actType":
  This field indicates the handling of the data stream specified by "index" key. The data type "actType" is "string" and only following values are applicable.

  - "add" : indicates the addition of the participant information.

  - "del" : indicates the deletion of the the participant information

  - "mod" : indicates the modification of the participant information

- "participantId":
  This field indicates the anonymized RTC user ID. The data type of "participantId" field is "string".

- "displayText":
  This field indicates the string which is represented as the RTC user name on the display. The data type of "displayText" is "string". This field is only applicable when the "actType" field of the "participantDesc" sub-key is set to "add" or "mod".

- "displayImage":
  This field indicates the URI of the picture/image which is shown as the RTC user icon on the display. The data type of "displayImage" is "string". This field is only applicable when the "actType" field of the "participantDesc" sub-key is set to "add" or "mod".

- "oId":
  This field is equivalent to the "oId" key. This field is set in the "participantDesc" sub-key only when the "actType" sub-key of the "participantDesc" sub-key is set to "add" and the RTC user Id of the RTC user is allowed to be indicated to the destination.

- "userState":
  This field indicates the status of the participant. The data type of "userState" sub-key is "string" and only following values are applicable.

  - "joiningIn" : the participant is joining in the media session.

  - "alearting" : the participant is in alerting status.

  - "joined" : the participant has joined in the media session.

  - "leaving" : the participant is left from the media session and proceeding the leaving.

#### 6.4.5.5.4.3.20 Originating ID ("oId")

This key is used for conveying the identifier(s) of an originating RTC user to the target RESPECT endpoint of the request. Originating ID ("oId") key is allowed to be included in the "msetup" and "mupdate" request/response. The data type of the "oId" key is "object". The "oId" key has following 4 sub-keys:

- "user"
- "network"
- "privacy"
- "passport"

The "user" sub-key is used for conveying an originating ID provided by an originating RTC user. The data type of "user" sub-key is "object".

The "network" sub-key is used for conveying the network-asserted originating ID provided by a RESPECT endpoint (AS). A RESPECT endpoint (AS) needs to delete this sub-key before sending a RESPECT message containing this sub-key to an entity outside the trust domain. The data type of "network" sub-key is "object".

Both the "user" and "network" sub-keys have following fields:

- "uri"
- "tn"
- "displayName"

The data type of "uri", "tn" and "displayName" fields are "string". The value of each field is set as follows.

"uri": RTC user ID

"tn": telephone number (global number digits excluding "+")

"displayName": display name of the RTC user

The "privacy" sub-key is used for indicating the privacy setting of an originating ID requested by an RTC user. The data type of "privacy" sub-key is "array(string)". When this sub-key is not contained in the RESPECT message from a RESPECT endpoint (UE), then RESPECT endpoints (AS) will forward "user" and "network" sub-key towards a RESPECT endpoint (UE). The following value is applicable to indicate privacy setting.

"id": This value indicates the "network" sub-key containing a network-asserted originating ID is required to be removed when the RESPECT message is sent to a RESPECT endpoint (UE). If this value is set into the "privacy" sub-key, a RESPECT endpoint (UE) is not allowed to set "user" sub-key in the "oId" key.

The "passport" sub-key is used for conveying a signature of originating ID provided by an originating RTC network, to verify an originating ID in a terminating RTC network. A RESPECT endpoint (AS) needs to delete this sub-key before sending a RESPECT message containing this sub-key to an entity outside the trust domain. The data type of "passport" sub-key is "object". The "passport" sub-key has following fields.

- "identity"
- "info"
- "alg"
- "ppt"

The "identity" sub-key contains a signature of the PASSporT generated as specified in IETF RFC 8225 [x6] and IETF RFC 8588 [x7]. Only full form PASSporT is allowed to be included in the "identity" sub-key. When creating the PASSporT at an originating RESPECT endpoint (AS), PASSporT header and PASSporT payload are set as follows, then a signature is generated using these header and payload according to IETF RFC 8825 [x6].

<PASSporT header>

- The "typ" is required to be set to "passport" as specified in IETF RFC 8225 [35].
- The "alg" is required to be set to the cryptographic algorithm for the signature part. In this version of the RESPECT, the "alg" is required to be set to "ES256".
- The "x5u" is required to be set to the URI referring to the resource for the X.509 public key certificate corresponding to the key used to generate the PASSporT signature.

- The "ppt" is required to be set to "shaken" as specified in IETF RFC 8588 [43].

<PASSporT payload>

- The "attest" is required to be set to the attestation level as specified in IETF RFC 8588 [43]. In this version of the RESPECT, the "attest" claim is required to be set to "A" since every RTC user ID is authenticated by an operator or a SP.
- The "dest" is required to be set to the same value of the "dId" key of the "msetup" request.
- The "iat" is required to be set to the date and time of issuance of the PASSporT signature as specified in IETF RFC 8225 [x7].
- The "orig" is required to be set to the same value of the "tn" or "uri" of the "network" sub-key in the "oId" key of the request.
- The "origid" is required to be set to the UUID as specified in IETF RFC 8588 [43]. How to generate the UUID and the granularity of the "origid" is determined by an operator policy.
- The "mky" is not allowed to be used.

The "info" sub-key is required to be included in the "passport" sub-key. This sub-key is used for indicating the URI referring to the resource for the X.509 public key certificate corresponding to the key used to generate the PASSporT signature as same as the "info" parameter specified IETF RFC 8224 [34].

The "alg" sub-key is allowed to be included in the "passport" sub-key. This sub-key is used for indicating the encryption algorithms of used to generate the PASSporT signature as same as the "alg" parameter specified IETF RFC 8224 [34].

The "ppt" sub-key is allowed to be included in the "passport" sub-key. This sub-key is used for indicating the required PASSporT extension needed to be supported for verification as same as the "alg" parameter specified IETF RFC 8224 [34].

#### 6.4.5.5.4.3.21 Requested information list ("resourcesReq") / Information list ("resourcesRes")

"getinfo" request is allowed to include "resourcesReq" key. The data type of "resourcesReq" key is "array". Only "string" data which indicates the item name of the requested information is allowed to be set into the allay.

"getinfo" response is allowed to include "resourcesRes" key. The data type of "resourcesRes" key is "object". The sub-key of the "resourcesRes" key consists of a key which is the item name of the requested information and a value representing the information. If the information is not available, the sub-key corresponding to the information is not included in the "resourceRes" key.

The applicable information is defined in clause D.2.5. The string representing the item name of the information is required to start from "/". The application specific item name of the information is allowed to be used. To avoid the confliction with other item names, the application specific item name is required to include reverse order of the internet domain which owned by the operator or the SP who provide the RTC service, before the application specific item name (e.g., /com.example/net/conf/appSpecificItem). This rule enables to avoid confliction of key name. The internet domain part of the item name is required to be lower-case letter. The data type of application specific item is allowed to be any data type.

#### 6.4.5.5.4.3.22 Updating key list ("updatingKeys")

This key indicates the keys which are requested to be updated in the "mupdate" request. The data type of "updatingKeys" key is "array" Only "string" data type values indicating key is allowed to be set into the array of "updatingKeys" key.

#### 6.4.5.5.4.3.23 Updated key list ("updatedKeys")

This key indicates the keys which are updated by the "mupdate" request. The data type of "updatedKeys" key is "array" Only "string" data type values indicating key is allowed to be set into the array of "updatedKeys" key.

#### 6.4.5.5.4.3.24 Called party ID ("cId")

This key indicates the RTC user ID or RTC resource ID which is used as destination ID. Called party ID ("cId") key is allowed to be included in the "msetup" request sent from a RESPECT endpoint (AS). The data type of "cId" key is "object".

For the "dId" key, any one of the following fields is allowed to be set.

- "uri"
- "tn"

The data type of "uri" and "tn" fields are "string". The value of each fields is set as follows.

- "uri": either RTC resource ID or RTC user ID
- "tn": telephone number (global number digits excluding "+")

The applicable URI scheme for RTC resource ID is "3gpp-respect-v1" for native RTC resources. The host part is set to the internet domain which owned by an operator or an SP who provides the RTC service to the user. The userinfo part is assigned by the operator or the SP who provides the RTC service to the user.

#### 6.4.5.5.4.3.25 User data ("userData")

This key is able to be used for conveying user specified data in free format. The data type of "userData" key is "object". This key is allowed to be included in "msetup", "mupdate", "mdisc" request/response. Note that this key is delivered to the endpoint of the media session (e.g., from originating UE to terminating UE).

### 6.4.5.5.4.4 Application specific key

#### 6.4.5.5.4.4.1 General

RESPECT allows to use application specific keys for flexibility.

The application specific key name is required to include reverse order of the internet domain which owned by the operator or the SP who provide the RTC service, before the application specific key (e.g., com.example.specificKey). This rule enables to avoid confliction of key name.

The data type of application specific Key is specified by the application.

## 6.4.5.5.5 Response code for error response

### 6.4.5.5.5.1 General

This clause describes the applicable URL string for "type" sub-key of "problemDetails" key.

The "type" sub-key indicates the URL string of the error/problem details. The URL is required to be following format.

**<scheme>://<error-type>/<error-details>**

- <scheme> is required to be "3gpp-respect"
- <error-type> is required to be set as follows:
  - "error": This context indicates the method or the setup/modification of media session is failed to process by a reason indicated by <error-details>. "error" is not allowed to be used for indicating the error caused by timeout.
  - "timeout" : This context indicates the method or the setup/modification of media session is failed is not successfully proceeded before a time expires. The detailed time is expected to be explained in <error-details>
- <error-details> is required to be set to the following context based on the reason of the error:
  - "method-unsupported":
    This context indicates that the method of the request is not supported at the destination RESPECT endpoint.

- "feature-unsupported":
  This context indicates that the feature required in the request is not supported at the destination RESPECT endpoint.

- "feature-required":
  This context indicates that a required feature(s) is not included in the request. The request/response indicating this context is requited to include "requiredFeature" key contains the value(s) indicating required feature(s).

- "mediaSession-id-not-found":
  This context indicates that the media session which specified by the "mediaSessionId" ID is not existing.

- "mediaSession-offer-required":
  This context indicates that the request is required to include SDP offer in the "msetup" request or "mupdate" request. The RESPECT endpoint is allowed to generate another "msetup"/"mupdate" request including SDP offer.

- "mediaSession-offer-rejected":
  This context indicates that the SDP offer included in the request is rejected by the destination RESPECT endpoint.

- "destination-not-found":
  This context indicates that the destination which specified by "dId" key in the request is not found.

- "destination-rejected":
  This context indicates that the media session setup to the destination specified in "dId" key is rejected by the destination.

- "auth-failed":
  This context indicates that the "auth" request is failed. The reason for the failure is not provided. If the "retryAfter" key is included in the response, the RESPECT endpoint is not allowed to send an "auth" request during the duration indicated in the "problemDetails" key.

- "congested":
  This context indicates that the RESPECT request is failed due to congestion of a C-Plane path. The RESPECT endpoint which receives the response including this "type" sub-key is not allowed to send any request during the duration indicated in the "retryAfter" in the "problemDetails" key.

#### 6.4.5.5.6 Originating ID and verification using signature verification and attestation information

##### 6.4.5.5.6.1 General

This clause describes the following feature of RESPECT.

- Handling of the originating ID
- Originating ID verification using signature verification and attestation information

##### 6.4.5.5.6.2 Handling of originating ID

###### 6.4.5.5.6.2.1 General

This clause describes the procedures for handling of originating ID.

###### 6.4.5.5.6.2.2 User-provided originating ID

When initiating a media session, the RESPECT endpoint (UE) is allowed to include an originating ID in the "user" sub-key of the "oId" key in the "msetup" request.

When receiving the "msetup" request, the RESPECT endpoint (AS) receives the "msetup" request, the RESPECT endpoint (AS) is required to copy the "user" sub-key of the "oId" key in the received "msetup" request into the "user" sub-key of the "oId" key in the sending "msetup" request.

#### 6.4.5.5.6.2.3 Network-asserted originating ID

The RESPECT endpoint (UE) is not allowed to include an originating ID in the "network" sub-key of the "oId" key in the "msetup" request.

When receiving the "msetup" request from a RESPECT endpoint (UE), the RESPECT endpoint (AS) retrieves an ID assigned to the RESPECT endpoint (UE). Before sending the "msetup" request to the entity within the trust domain, the RESPECT endpoint (AS) is allowed to include the retrieved IDs in the "network" sub-key of the "oId" key as a network-asserted ID.

When receiving the "network" sub-key from the preceding RESPECT endpoint, the RESPECT endpoint (AS) is required not to include the "network" sub-key in the "msetup" request if the request is sent to the entity outside the trust domain.

NOTE: RESPECT endpoint (AS) is expected to be able to retrieve the originating ID (e.g., RTC user Identity of the originating RESPECT endpoint (UE)) from the ASWF.

#### 6.4.5.5.6.2.4 Privacy

The RESPECT endpoint (UE) is allowed to include privacy indication information in the "privacy" sub-key of the "oId" key in the "msetup" request when the RESPECT endpoint (UE) initiates the "msetup" request. The privacy indication information is set as described in clause 6.4.5.5.4.3.20.

When receiving the "privacy" sub-key of the "oId" key in the "msetup" request, the RESPECT endpoint (AS) is required to copy the "privacy" sub-key of the "oId" key in the received "msetup" request to the "privacy" sub-key of the "oId" key in the "msetup" request which sent to another RESPECT endpoint in the trust domain.

When receiving the "privacy" sub-key set to "id" from the preceding RESPECT endpoint, the RESPECT endpoint (AS) is required not to include the "network" and "privacy" sub-keys in the "msetup" request, if the request is sent to the RESPECT endpoint (UE).

When receiving the "privacy" sub-key from the preceding RESPECT endpoint, the RESPECT endpoint (AS) is required not to include the "privacy" sub-key in the "msetup" request if the request is sent to the entity outside the trust domain.

### 6.4.5.5.6.3 Originating ID verification using signature verification and attestation information

#### 6.4.5.5.6.3.1 General

The RESPECT protocol supports the Originating ID verification using signature verification and attestation information. This clause describes the procedure for the originating ID verification using signature verification and attestation information.

#### 6.4.5.5.6.3.2 Signing for the originating ID

The RESPECT endpoint (AS) in the originating RTC network is allowed to include PASSporT in the "identity" field of the "passport" sub-key of the "oId" key in the "msetup" request which sent to the RESPECT endpoint in the trust domain. The PASSporT is generated as described in clause 6.4.5.5.4.3.20.

#### 6.4.5.5.6.3.3 Verification of the originating ID

The RESPECT endpoint (AS) in the terminating RTC network is allowed to use the PASSporT in the "identity" field of the "passport" sub-key of the "oId" key in the received "msetup" request for verification of originating ID in the "network" sub-key.

When sending the "msetup" request to the entity outside the trust domain, the RESPECT endpoint (AS) is required not to include "passport" sub-key in the request.

## 6.4.5.6 General call flow and procedure

### 6.4.5.6.1 General

This clause describes the general call flows and procedures for RESPECT listed in Table 6.4.5.6.1-1.

**Table 6.4.5.6.1-1: RESPECT Call flows**

| No. | Title | related CS # | Signalling path | clause | message example |
|---|---|---|---|---|---|
| 1 | Authentication | CS #3, CS #4 | UE1 - WSF | 6.4.5.6.2 | B.2 |
| 2 | Media session setup and disconnection for the operator self-contained RTC resource | CS #3 | UE1 - WSF | 6.4.5.6.3 | B.3 |
| 3 | Media session setup and disconnection for the RTC resource provided by other operator | CS #4 | UE1 - WSF1 - IWF1 - IWF2 - WSF2 | 6.4.5.6.4 | B.4 |
| 4 | Media session setup and disconnection between UEs within a single operator network | CS #3 | UE1 - WSF - UE2 | 6.4.5.6.5 | B.5 |
| 5 | Media session setup and disconnection between UEs over inter-operator networks | CS #4 | UE1 - WSF1 - IWF1 - IWF2 - WSF2 -UE2 | 6.4.5.6.6 | B.6 |

<Premises of the message flow>

- The RTC user established a secure WebSocket connection (i.e., control session) with the WSF in the connected operator network.

In this clause, RESPECT endpoint is described as follows.

- RESPECT endpoint (UE): UE, originating UE, terminating UE
- RESPECT endpoint (AS) indicating WSF: WSF, IWF
- RESPECT endpoint (AS) indicating IWF: IWF

In this clause, the preceding control session means that the control session which the RESPECT endpoint (AS) received a request message from another RESPECT endpoint (AS) and the succeeding control session means the control session which the RESPECT endpoint (AS) sends the request to another RESPECT endpoint. Then the RESPECT endpoint (AS) receives the response message corresponding to the request message on the succeeding control session and sends the response message on the preceding control session which the corresponding request message is received.

In this clause, the response message includes "success" key is set to "true" is described as follows:

- success response

In this clause, the response message includes "success" key is set to "false" is described as follows:

- error response

### 6.4.5.6.2 Authentication

When the UE established a secure WebSocket connection with the WSF, the UE is required to be authenticated by the WSF in the operator network to use RESPECT for media session control. This clause describes the general message flow and procedure for authentication.

<Premises of the message flow>

- The UE has the JSON Web Signature (JWS) credential through preceding authentication with the operator network or an external identity provider (IdP).

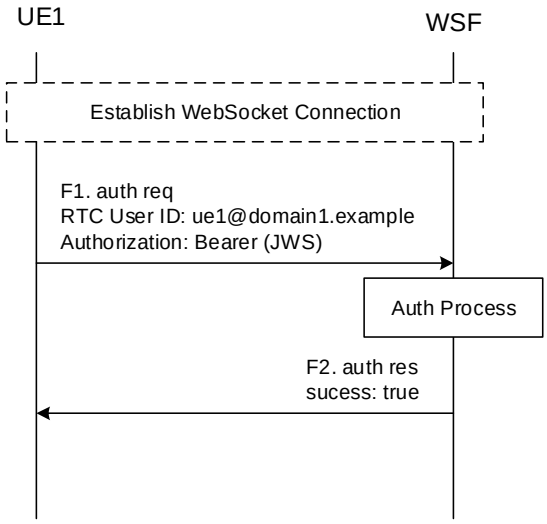


**Figure 6.4.5.6.2-1: Authentication**

1. After the establishment of the secure WebSocket connection with the WSF in the operator network, the UE sends "auth" request for requesting authentication to the WSF over the secure WebSocket connection. The "auth" request is required to include the information used for the authentication as follows.

   a) The RTC user ID requested to be authenticated (i.e., "rtcUserId" key).

   b) The authentication type and credential for the authentication. In this flow, token based bearer authentication is applied (i.e., "authType" key and "authorization" key).

2. The WSF processes the authentication of the RTC user ID using the received token. If the authentication is successful, the WSF:

   i. registers the binding information to the database (ASWF) in the operator network and

   ii. sends "auth" success response to the UE. The "auth" success response is required to include the following information.

      a) The duration that the authentication is valid (i.e., "expires" key). The duration is determined by the operator policy.

After the successful authentication, the UE is allowed to send RESPECT messages other than "auth" request. Also the UE is able to called from other UEs since the WSF/IWF can identify the UE by using the registered binding information is in the database.

When the authentication period expires, the authentication and the registered binding information is deleted. Then, the UE is not able to set up new media session and the existing media session is disconnected.

To avoid the deletion of the authentication and registered binding information, the UE is required to send the authentication request before the previous authentication is expired, after the successful authentication.

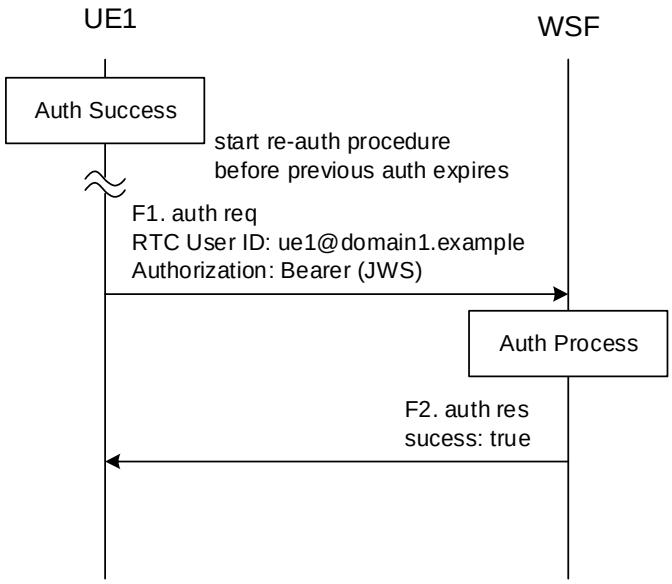


**Figure 6.4.5.6.2-2: Re-authentication**

1. The UE sends "auth" request for requesting re-authentication to the WSF over the secure WebSocket connection. The "auth" request is required to include the information used for the authentication as same as the previous authentication. If there are any updates on the information used for the authentication, the latest information is required to be used.

2. The WSF processes the authentication of the RTC user ID using the received token. If the authentication is successful, the WSF is required to

   i. update the registered binding information of the database (ASWF) in the operator network and

   ii. send "auth" success response to the UE as same as the previous authentication.

#### 6.4.5.6.3 Media session setup and disconnection for the operator self-contained RTC resource

This clause describes the general call flow and procedure for media session setup and disconnection with the RTC resource in the connected operator network.

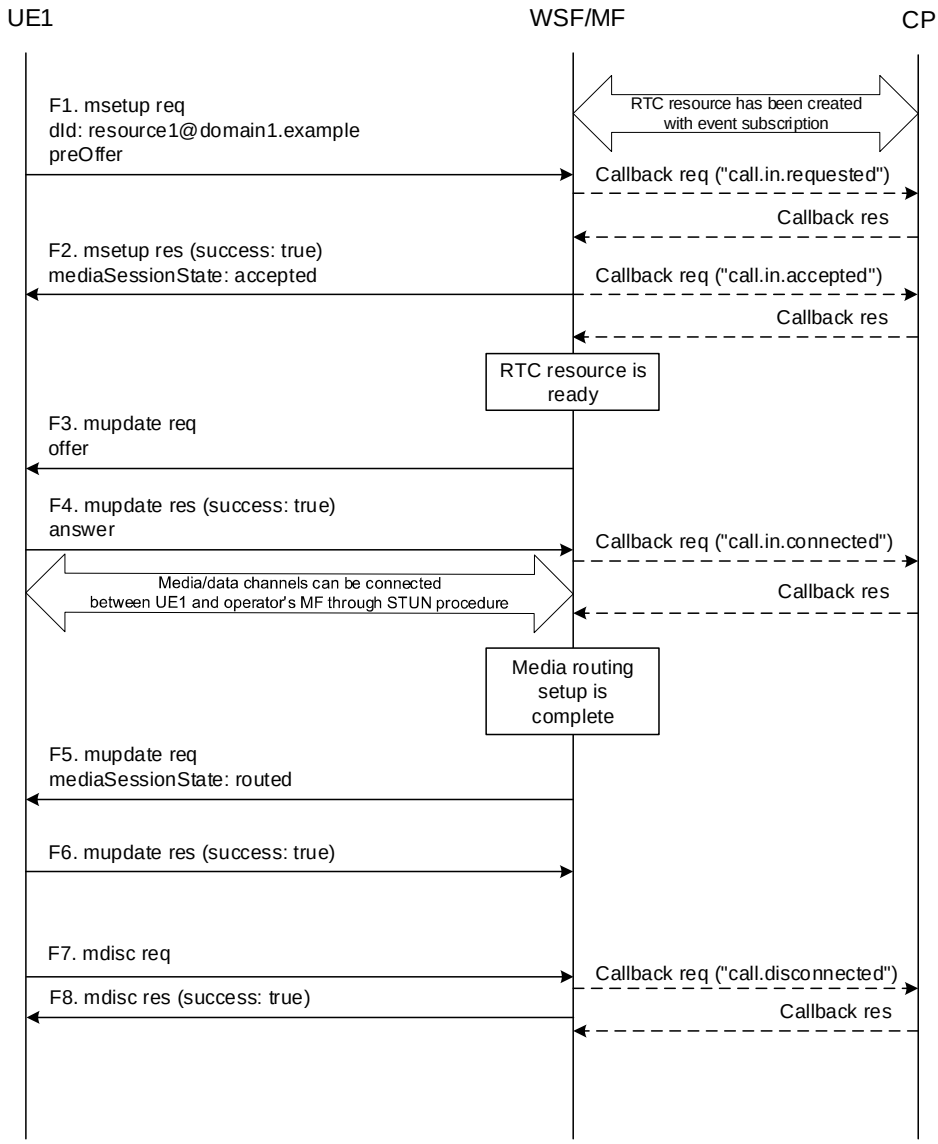


**Figure 6.4.5.6.3-1: Media session setup and disconnection between UE1 and RTC resource provided by the operator network**

NOTE: Dashed arrows in the Figure 6.4.5.6.3-1 show the service control API interactions and outside the scope of this solution.

1. UE1 sends "msetup" request for requesting a media session setup to the WSF over the secure WebSocket connection. The "msetup" request is required to include following information.

   a) RTC resource ID of the RTC resource where the media session is connected to (i.e., "dId" key).

   b) Media session ID of the media session which is requested to be set up (i.e., "mediaSessionId" key). This Media session id is generated by the UE and is required to be unique on the control session.

   c) Media information including "preOffer" description of the UE for the media session (i.e., "mediaInfo" key).

2. The WSF detects the RTC resource ID of the destination in the received "msetup" request is valid and the RTC resource is assigned to the WSF. If the UE (RTC user ID) is allowed to connect to the media resource:

   i. the WSF sends the "msetup" success response to the UE for indicating the result of the "msetup" request and media session state immediately. The "msetup" success response includes following information.

a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key). The media session ID in the response is same ID included in the "msetup" request.

b) Current media session state (i.e., "mediaSessionState" key) to notify the media session state transitions to "accepted" since the WSF is acting as terminating RESPECT endpoint (AS).

c) Media Information including the participant description (i.e., "participantDesc" sub-key of the "meidaInfo" key), if available.

ii. The WSF interact with the RTC resource (i.e., MF) to reserve the U-plane resource for the requested media session set up and obtain the SDP information.

3. Upon the U-Plane resource becomes available, the WSF sends "mupdate" request to the UE for the offer/answer negotiation of the media session. The "mupdate" request includes following information.

a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key).

b) Media information including the complete SDP offer description. The SDP offer description is generated based on the information received from RTC resource (i.e., "mediaInfo" key).

c) The information which intended to be updated (i.e., "updatingKeys" key)

4. If the received "mupdate" request is valid and included SDP offer is acceptable for the UE, the UE sends "mupdate" success response to the WSF for sending SDP answer to the received SDP offer. The "mupdate" response includes following information.

a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key). The media session ID in the response is same ID included in the "mupdate" request.

b) Media information including complete "answer" description of the UE for the media session (i.e., "mediaInfo" key).

c) The updated information by the "mupdate" method (i.e., "updatedKeys" key). In this case, the information included in the "mediaInfo" key is updated.

5. If the offer/answer for the media session is successfully completed, the WSF allocate the U-Plane media resource to the MF. Upon the media routing for the media session is successfully configured, the WSF send the "mupdate" request to the UE for indicating the media session setup is successfully completed. The "mupdate" response includes following information.

a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key).

b) Current media session state (i.e., "mediaSessionState" key) since the media session state transitions from "accepted" to "routed".

c) Media information including metadata of the updated participant information (i.e., "participantDesc" sub-key" of the "mediaInfo" key).

d) The information which intended to be updated (i.e., "updatingKeys" key)

6. If the received "mupdate" request is valid, the UE store the updated information indicated in the "updatingKeys" key and sends "mupdate" success response to the WSF for indicating the result of "mupdate" request. The "mupdate" response includes following information.

a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key). The media session ID in the response is same ID included in the "mupdate" request.

b) The updated information by the "mupdate" method (i.e., "updatedKeys" key). In this case, the information included in the "mediaSessionState" key and the "mediaInfo" key are updated.

7. The UE sends the "mdisc" request to disconnect the media session. After sending the "mdisc" request, the UE is allowed to disconnect the media session before receiving "mdisc" success response. The "mdisc" request includes following information.

a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key).

8. If the received "mdisc" request is valid, the WSF instruct the RTC resource to release the U-Plane resource for the media session and sends the "mdisc" success response to disconnect the media session. The "mdisc" request includes following information.

   a) Media session ID indicating the target media session of the method (i.e., "mediaSessionid" key). The media session ID in the response is same ID included in the "mdisc" request.

#### 6.4.5.6.4 Media session setup and disconnection for the RTC resource provided by other operator

This clause describes the general call flow and procedure for media session setup and disconnection with the RTC resource in the connected operator network.

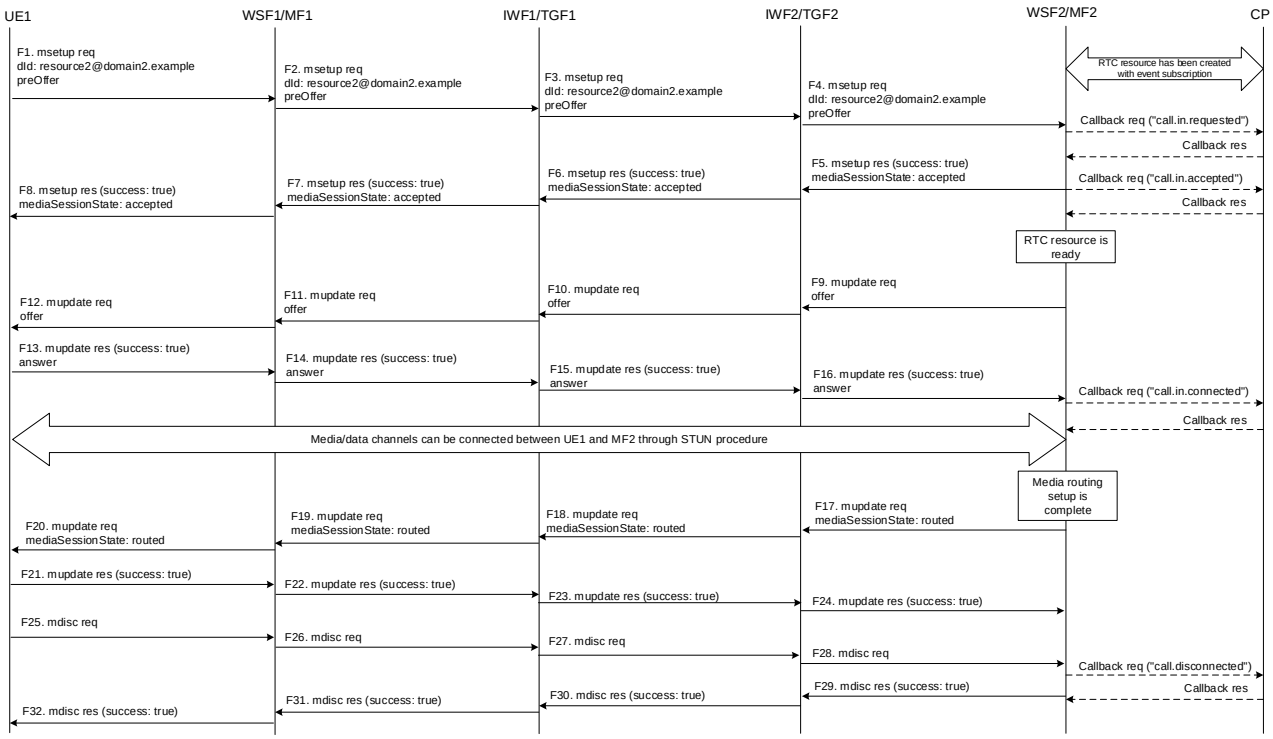


**Figure 6.4.5.6.4-1: Media session setup and disconnection for the RTC resource provided by other operator**

NOTE 1: Dashed arrows in the Figure 6.4.5.6.4-1 show the service control API interactions and outside the scope of this solution.

1. This process is same as process 1 of clause 6.4.5.6.3.

2. WSF1 detects the destination of the "msetup" request is in other network by the domain of the URI and proceeds the following steps to send the "msetup" request to the destination.

   i. WSF1 retrieves the information of IWF1 which is acting as the exit point of the operator network and connected to the IWF in the network where RTC resource is assigned to.

NOTE 2: How to retrieve the IWF1 information is FFS.

   ii. WSF1 sends the "msetup" request to the retrieved IWF1. The "msetup" request is constructed as follows.

      a) Media session ID is newly generated by the WSF1 since the uniqueness of the media session ID is guaranteed per control session. WSF1 is required to store the relationship between the received media session ID on the preceding control session and the generated media session ID on the succeeding media session. This relationship is necessary to identify the appropriate RESPECT endpoint where a received

request message is sent to, when the RESPECT endpoint (AS) received a request message for requesting update/disconnect the media session.

b) The individual keys in the "msetup" request from UE1 is copied to the "msetup" request sent to IWF1.

3. IWF1 detects the destination of the "msetup" request is in other network by domain of the URI and proceeds following steps to send "msetup" request to the destination.

i. IWF1 retrieves the information of IWF2 which is acting as the entry point of the network where RTC resource is assigned to.

NOTE 3: How to retrieve the IWF2 information is FFS.

ii. IWF1 sends the "msetup" request to the retrieved IWF2 in the network where the RTC resource is assigned to. The "msetup" request is constructed as follows.

a) Media session ID is newly generated and stored by IWF1 as same as process 2-ii-a) in this clause.

b) The individual keys in the "msetup" request from WSF1 is basically copied to the "msetup" request sent to the IWF2. IWF1 acts as a gateway (exit point) of the network, then IWF1 is allowed to modify the values in the individual keys based on the operator policy. (e.g., hiding of the operator network specific information, modification of SDP description to involve TGF1 in the media path.)

If TGF1 is applied as media gateway, IWF1 instructs TGF1 to reserve U-Plane resource and receives the media related information from TGF1.

4. IWF2 detects the destination of the "msetup" request is in the network by domain of the URI and proceeds following steps to send "msetup" request to the destination.

i. IWF2 retrieves the information of WSF2 which the RTC resource is assigned to.

NOTE 3: How to retrieve the WSF2 information is FFS.

ii. IWF2 sends the "msetup" request to the retrieved WSF2 in the network where RTC resource is assigned to. The "msetup" request is constructed as follows. IWF2 is required to store the relationship between the received media session ID and the generated media session ID.

a) Media session ID is newly generated and stored by IWF2 as same as process 2-ii-a) in this clause.

b) The individual keys in the "msetup" request from IWF1 is basically copied to the "msetup" request sent to IWF2. IWF2 acts as a gateway (entry point) of the network, then IWF2 is allowed to modify the values in the individual keys based on the operator policy. (e.g., screening of the received information, modification of SDP description to involve TGF2 in the media path.)

If TGF2 is applied as media gateway, IWF2 instructs TGF2 to reserve U-Plane resource and receives the media related information from TGF2.

5. This process is same as process 2 of clause 6.4.5.6.3. To refer the process, "UE1" is replaced with "IWF2", and "WSF1" is replaced with "WSF2".

6. IWF2 checks the transaction ID of the "msetup" success response and sends the "msetup" success response to the IWF1 which sent the corresponding "msetup" request identified by the transaction ID. The "msetup" success response is constructed as follows:

a) The media session ID in the response is same media session ID included in the corresponding "msetup" request on the control session.

b) The individual keys in the "msetup" success response from WSF2 is basically copied to the "msetup" success response sent to IWF1. IWF2 acts as a gateway (exit point) of the network, then IWF2 is allowed to modify the values in the individual keys based on the operator policy. (e.g., hiding of the operator network specific information.)

7. IWF1 checks the transaction ID of the "msetup" success response and sends the "msetup" success response to the WSF1 which sent the corresponding "msetup" request identified by the transaction ID. The "msetup" response is constructed as follows:

a) The media session ID in the response is same media session ID included in the corresponding "msetup" request on the control session.

b) The individual keys in the "msetup" success response from IWF2 is basically copied to the "msetup" success response sent to WSF1. IWF1 acts as a gateway (entry point) of the network, then IWF1 is allowed to modify the values in the individual keys based on the operator policy. (e.g., screening of the received information)

8. WSF1 checks the transaction ID of the "msetup" success response and sends the "msetup" success response to the UE1 which sent the corresponding "msetup" request identified by the transaction ID. The "msetup" response is constructed as follows:

   a) The media session ID in the response is same media session ID included in the corresponding "msetup" request on the control session.

   b) The individual keys in the "msetup" success response from IWF1 is copied to the "msetup" success response sent to UE1.

   UE1 stores the received information and mupdates the media session related information (i.e., media session state and media session information) indicated by the media session ID in the received "msetup" success response.

9. This process is same as process 3 of clause 6.4.5.6.3. To refer the process, "UE1" is replaced with "IWF2", and "WSF1" is replaced with "WSF2".

10. IWF2 checks the media session ID of the "mupdate" request and sends the "mupdate" request to IWF1 which has the control session corresponding to the media session identified by the media session ID. The "mupdate" request is constructed as follows:

    a) The media session ID in the "mupdate" request on the succeeding control session is set to the media session ID of the media session which corresponding to the media session indicated in the received "mupdate" request on the preceding control session.

    b) The individual keys in the "mupdate" request from WSF2 is basically copied to the "mupdate" request sent to IWF1. IWF2 acts as a gateway (exit point) of the network, then IWF2 is allowed to modify the values in the individual keys based on the operator policy. (e.g., hiding of the operator network specific information. modification of SDP description to involve TGF2 in the media path.)

11. IWF1 checks the media session ID of the "mupdate" request and sends the "mupdate" request to WSF1 which has the control session corresponding to the media session identified by the media session ID. The "mupdate" request is constructed as follows:

    a) The media session ID in the "mupdate" request on the succeeding control session is set as same as process 10-a) in this clause.

    b) The individual keys in the "mupdate" request from IWF2 is basically copied to the "mupdate" request sent to WSF1. IWF1 acts as a gateway (entry point) of the network, then IWF1 is allowed to modify the values in the individual keys based on the operator policy. (e.g., screening of the received information, modification of SDP description to involve TGF1 in the media path.)

12. WSF1 checks the media session ID of the "mupdate" request and sends the "mupdate" request to UE1 which has the control session corresponding to the media session identified by the media session ID. The "mupdate" request is constructed as follows:

    a) The media session ID in the "mupdate" request on the succeeding control session is set as same as process 10-a) in this clause.

    b) The individual keys in the "mupdate" request from IWF1 is basically copied to the "mupdate" request sent to UE1.

    UE1 stores the received information and updates the media session related information (i.e., media session state and media session information) indicated by the media session ID in the received "mupdate" request.

13. This process is same as process 4 of clause 6.4.5.6.3.

14. WSF1 checks the transaction ID of the "mupdate" success response and sends the "mupdate" success response to the IWF1 which sent the corresponding "mupdate" request identified by the transaction ID. The "mupdate" response is constructed as follows:

    a) The media session ID in the response on the preceding control session is set to the media session ID of the media session which corresponding to the media session indicated in the received "mupdate" success response on the succeeding control session.

    b) The individual keys in the "msetup" success response from UE1 is copied to the "mupdate" success response sent to IWF1.

15. IWF1 checks the transaction ID of the "mupdate" success response and sends the "mupdate" success response to the IWF2 which sent the corresponding "mupdate" request identified by the transaction ID. The "mupdate" response is constructed as follows:

    a) The media session ID in the response on the preceding control session is set to the media session ID of the media session is set as same as process 14-a) in this clause.

    b) The individual keys in the "msetup" success response from WSF1 is basically copied to the "mupdate" success response sent to IWF2. IWF1 acts as a gateway (exit point) of the network, then IWF1 is allowed to modify the values in the individual keys based on the operator policy. (e.g., hiding of the operator network specific information, modification of SDP description to involve TGF1 in the media path.)

    If TGF1 is applied as media gateway, IWF1 instructs the TGF1 to allocates the U-Plane resource and receives the media related information from TGF1.

16. IWF2 checks the transaction ID of the "mupdate" success response and sends the "mupdate" success response to the WSF2 which sent the corresponding "mupdate" request identified by the transaction ID. The "mupdate" response is constructed as follows:

    a) The media session ID in the response on the preceding control session is set to the media session ID of the media session is set as same as process 14-a) in this clause.

    b) The individual keys in the "msetup" success response from IWF1 is basically copied to the "mupdate" success response sent to WSF2. IWF2 acts as a gateway (entry point) of the network, then IWF2 is allowed to modify the values in the individual keys based on the operator policy. (e.g., screening of the received information, modification of SDP description to involve TGF2 in the media path.)

    If TGF2 is applied as media gateway, IWF2 instructs the TGF2 to allocates the U-Plane resource and receives the media related information from TGF2.

17. This process is same as process 5 of clause 6.4.5.6.3.

18. IWF2 checks the media session ID of the "mupdate" request and sends the "mupdate" request to IWF1 which has the control session corresponding to the media session identified by the media session ID as same as process 10.

19. IWF1 checks the media session ID of the "mupdate" request and sends the "mupdate" request to WSF1 which has the control session corresponding to the media session identified by the media session ID as same as process 11.

20. WSF1 checks the media session ID of the "mupdate" request and sends the "mupdate" request to UE1 which has the control session corresponding to the media session identified by the media session ID as same as process 12.

    UE1 stores the received information and updates the media session related information (i.e., media session state and media session information) indicated by the media session ID in the received "mupdate" request.

25. This process is same as process 7 of clause 6.4.5.6.3.

26. WSF1 checks the media session ID of the "mdisc" request and sends the "mdisc" request to IWF1 which has the control session corresponding to the media session identified by the media session ID as same as process 2.

27. IWF1 checks the media session ID of the "mdisc" request and sends the "mdisc" request to IWF2 which has the control session corresponding to the media session identified by the media session ID as same as process 3.

28. IWF2 checks the media session ID of the "mdisc" request and sends the "mdisc" request to WSF2 which has the control session corresponding to the media session identified by the media session ID as same as process 4.

29. This process is same as process 8 of clause 6.4.5.6.3.

30. IWF2 checks the media session ID of the "mdisc" request and sends the "mdisc" request to IWF1 which has the control session corresponding to the media session identified by the media session ID as same as process 6.

31. IWF1 checks the media session ID of the "mdisc" request and sends the "mdisc" request to WSF1 which has the control session corresponding to the media session identified by the media session ID as same as process 7.

32. WSF1 checks the media session ID of the "mdisc" request and sends the "mdisc" request to UE1 which has the control session corresponding to the media session identified by the media session ID as same as process 8.

### 6.4.5.6.5 Media session setup and disconnection between UEs within a single operator network

This clause describes the general call flow and procedure for media session setup and disconnection with the RTC resource in the connected operator network.

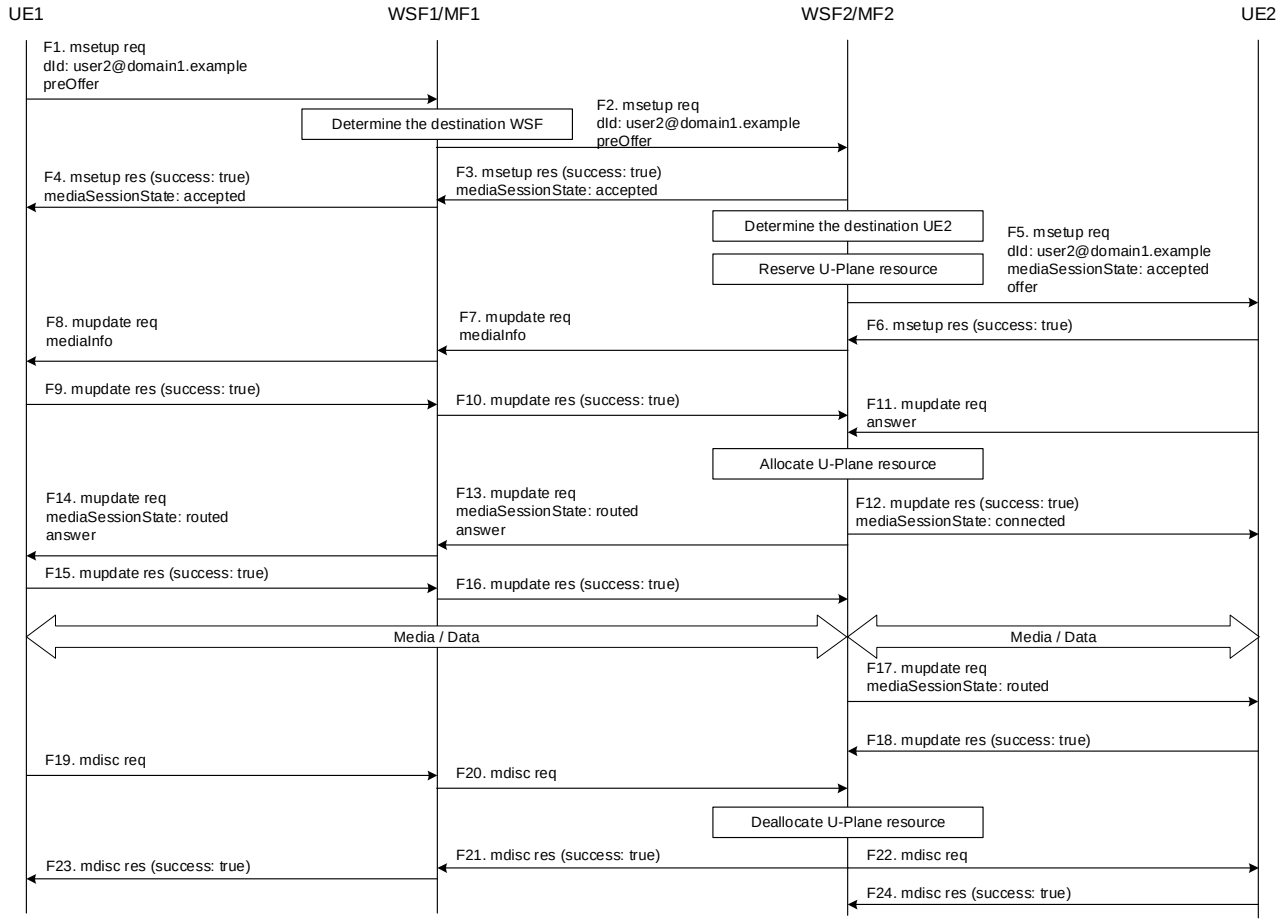


**Figure 6.4.5.6.5-1: Media session setup and disconnection between UE1 and UE2 within a single operator network**

1. UE1 sends an "msetup request for requesting a media session setup to WSF1 over the control session (i.e., secure WebSocket connection). The "msetup" request includes following information.

   a) RTC user ID of the RESPECT endpoint (i.e., UE2, in this case) which the media session is connected to (i.e., "dId" key).

   b) Media session ID of the media session which is requested to be set up (i.e., "mediaSessionId" key). This Media session id is generated by the UE and is required to be unique on the control session.

c) Media information including "preOffer" description of the UE for the media session (i.e., "mediaInfo" key).

2. WSF1 detects the destination of the "msetup" request is in the network by the domain of the URI and proceeds the following steps to send the "msetup" request to the destination.

   i. WSF1 retrieves the information of WSF2 which UE2 is connected to.

NOTE 1: WSF2 is expected to be able to use a database which stores the binding information.

NOTE 2: If the UE2 is connected to WSF1, WSF acts as WSF2 of this call flow.

   ii. WSF1 sends the "msetup" request to the retrieved WSF2 as same as process 2-ii of clause 6.4.5.6.4.

3. WSF2 detects the destination of the "msetup" request is the RESPECT endpoint (UE) connected to WSF2 via the valid control session. WSF2 proceed the following steps.

   i. WSF2 stores the received information in the "msetup" request.

   ii. WSF2 instruct MF2 to reserve U-Plane resource for the media session based on the received "preOffer" and the constructed initial SDP offer for sending "msetup" request to UE2.

   iii. WSF2 sends "msetup" success response to WSF1 for indicating the result of the "msetup" request and media session state immediately. The "msetup" success response includes following information.

      a) Media session ID in the response is set to same ID included in the "msetup" request.

      b) Current media session state (i.e., "mediaSessionState" key) to notify the media session state transitions to "accepted" since the WSF is acting as terminating RESPECT endpoint (AS).

      c) Media Information including the participant description (i.e., "participantDesc" sub-key of the "meidaInfo" key), if available.

4. WSF1 checks the transaction ID of the "msetup" success response and sends the "msetup" success response to the UE1 which sent the corresponding "msetup" request identified by the transaction ID. The "msetup" response is constructed as follows:

   a) The media session ID in the response is same media session ID included in the corresponding "msetup" request on the control session.

   b) The individual keys in the "msetup" success response from IWF1 is copied to the "msetup" success response sent to UE1.

   UE1 stores the received information and updates the media session related information (i.e., media session state and media session information) indicated by the media session ID in the received "msetup" success response.

5. WSF2 sends an "msetup" request to UE2. The "msetup" request includes following information.

   a) Media session ID which is newly generated and stored by WSF2 as same as process 2-ii-a) in clause 6.4.5.6.4.

   b) Current media session state (i.e., "mediaSessionState" key) to notify the media session state transitions to "accepted" since the WSF is acting as terminating RESPECT endpoint (AS).

   c) Media Information including the complete SDP description for initial offer (which is constructed at step 3-ii in this clause) and participant description (i.e., "sdp" sub-key and "partiscipantDesc" sub-key of the "mediaInfo" key).

   UE2 stores the received information and updates the media session related information (i.e., media session ID, media session state and media information) in the received "msetup" request.

6. UE2 sends an "msetup" success response to the received "msetup" request, to immediately notify that the "msetup" request is successfully proceeded at UE2. The "msetup" request incudes following information

   a) Media session ID in the response is same ID included in the corresponding "msetup" request (i.e., "mediaSessionid" key).

7. Upon receiving the "msetup" success response from UE2, WSF2 sends "mupdate" request towards UE1 to notify that the user state of UE2 is updated to "joininIn". The "request" request includes following information.

   a) Media session ID which indicating the target media session of the method (i.e., "mediaSessionId" key).

   b) Media Information including participant description related to the updated information (i.e., "partiscipantDesc" sub-key of the "mediaInfo" key).

   c) The individual keys which intended to be updated (i.e., "updatingKeys" keys).

8. WSF1 checks the media session ID of the "mupdate" request and sends the "mupdate" request to UE1 which has the control session corresponding to the media session identified by the media session ID. To notify that the user state of UE2 is updated to "joininIn". The "request" request includes following information.

   a) The media session ID in the "mupdate" request on the succeeding control session is set as same as process 12-a) in clause 6.4.5.6.4 (i.e., "mediaSessionId" key).

   b) Other individual keys in the received "mupdate" is basically copied to the "mupdate" request sent to UE1

9. If the received "mupdate" request is valid, UE1 updates the stored information indicated by the "updatingKeys" key in the received "mupdate" request and sends "mupdate" success response as follows.

   a) Media session ID in the response is same ID included in the corresponding "mupdate" request (i.e., "mediaSessionid" key).

   b) The updated information by the "mupdate" method (i.e., "updatedKeys" key). In this case, the information in the "mediaSessionState" key is updated.

10. This process is same as process 14 of clause 6.4.5.6.4. To refer the process, "IWF1" is replaced with "WSF2".

11. UE2 sends an "mupdate" request to complete offer/answer for the media session, when UE2 becomes to be able to connect the media session (e.g., SDP answer description is available and the user is able to join the call). The "mupdate" request includes following information.

    a) Media session ID which indicating the target media session of the method (i.e., "mediaSessionId" key).

    b) Media Information including complete SDP description for initial answer and participant description related to the updated information (i.e., "partiscipantDesc" sub-key of the "mediaInfo" key).

    c) The individual keys which intended to be updated (i.e., "updatingKeys" keys).

12. If the received "mupdate" request is valid, WSF2 updates the stored information indicated by the "updatingKeys" key in the received "mupdate" request. If the offer/answer for the media session is successfully completed, the WSF allocate the U-Plane media resource to the MF2. Then WSF2 sends the "mupdate" success response as follows.

    a) Media session ID in the response is same ID included in the corresponding "mupdate" request (i.e., "mediaSessionid" key).

    b) Current media session state (i.e., "mediaSessionState" key) to notify the media session state transitions to "connected" since the offer/answer is completed and the media session is connected between WSF2 and UE2.

    c) Media Information including participant description related to the updated information (i.e., "partiscipantDesc" sub-key of the "mediaInfo" key).

    d) The updated information by the "mupdate" method (i.e., "updatedKeys" key). In this case, the information in the "mediaSessionState" key and "mediaInfo" key are updated.

13. WSF2 sends an "mupdate" request to UE1 via WSF1, to complete the offer/answer for the media session setup. The "mupdate" request includes following information.

    a) Media session ID which indicating the target media session of the method (i.e., "mediaSessionId" key).

    b) Media session state (i.e., "mediaSessionState" key) to notify the media session state transitions to "routed" if the "mupdate" method is successfully completed.

c) Media Information including complete SDP description for initial answer (which is generated based on the SDP answer received from UE2 and MF2 U-plane resource) and participant description related to the updated information (i.e., "partiscipantDesc" sub-key of the "mediaInfo" key).

d) The individual keys which intended to be updated (i.e., "updatingKeys" keys").

14. This process is same as process 8 of this clause.

15. If the received "mupdate" request is valid, UE1 updates the stored information indicated by the "updatingKeys" key in the received "mupdate" request and sends "mupdate" success response as follows.

    a) Media session ID in the response is same ID included in the corresponding "mupdate" request (i.e., "mediaSessionid" key).

    b) The updated information by the "mupdate" method (i.e., "updatedKeys" key). In this case, the information in the "mediaSessionState" key is updated.

    UE1 establishes the media session using the SDP included in the sent preoffer and the received answer.

16. This process is same as process 10 of this clause.

17 WSF2 sends an "mupdate" request to UE2 to notify the media session state transition. The "mupdate" request includes the following information.

    a) Media session ID which indicating the target media session of the method (i.e., "mediaSessionId" key).

    b) Media session state (i.e., "mediaSessionState" key) to notify the media session state transitions to "routed".

    c) The individual keys which intended to be updated (i.e., "updatingKeys" keys").

18. If the received "mupdate" request is valid, UE2 updates the stored information indicated by the "updatingKeys" key in the received "mupdate" request and sends "mupdate" success response as follows.

    a) Media session ID in the response is same ID included in the corresponding "mupdate" request (i.e., "mediaSessionid" key).

    b) The updated information by the "mupdate" method (i.e., "updatedKeys" key). In this case, the information in the "mediaSessionState" key is updated.

19. This process is same as process 8 of clause 6.4.5.6.3.

20. This process is same as process 26 of clause 6.4.5.6.4. To refer the process, "IWF1" is replaced with "WSF2"

21. This process is same as process 29 of clause 6.4.5.6.4. To refer the process, "IWF2" is replaced with "WSF1"

22. WSF2 sends an "mdisc" request to UE2 to disconnect the media session. The "mdisc" request includes the following information.

    a) Media session ID which indicating the target media session of the method (i.e., "mediaSessionId" key).

23. This process is same as process 32 of clause 6.4.5.6.4.

24. If the received "mdisc" request is valid, UE2 disconnects the media session indicated by "mediaSessionId" key and sends "mdisc" success response as follows.

    a) Media session ID in the response is same ID included in the corresponding "mupdate" request (i.e., "mediaSessionid" key).

#### 6.4.5.6.6 Media session setup and disconnection between UEs over inter-operator networks

This clause describes the general call flow and procedures for media session setup and disconnection with the RTC user in the operator network other than the connected operator network.

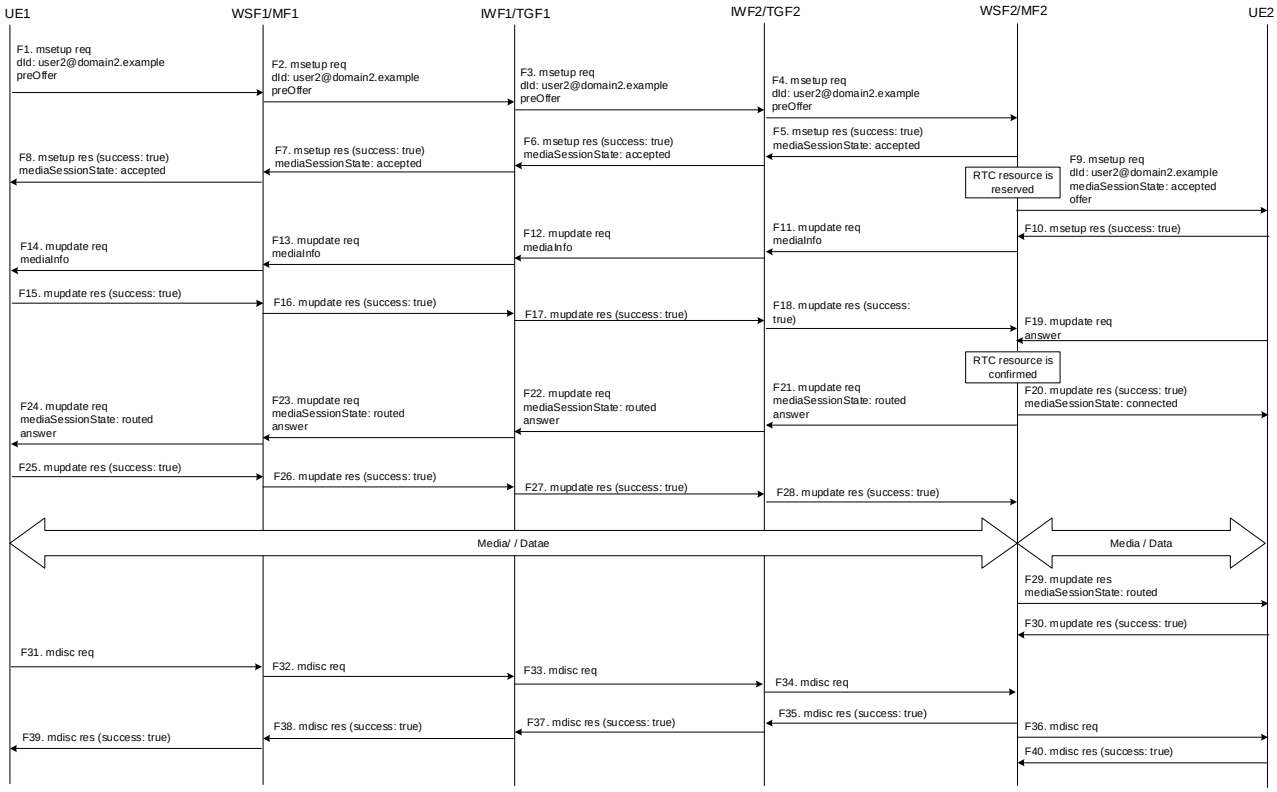


**Figure 6.4.5.6.6-1: Media session setup and disconnection between UEs over inter-operator networks**

1. This process is same as process 1 of clause 6.4.5.6.5.
2. This process is same as process 2 of clause 6.4.5.6.4.
3. This process is same as process 3 of clause 6.4.5.6.4.
4. This process is same as process 4 of clause 6.4.5.6.4.
5. This process is same as process 5 of clause 6.4.5.6.4.
6. This process is same as process 6 of clause 6.4.5.6.4.
7. This process is same as process 7 of clause 6.4.5.6.4.
8. This process is same as process 8 of clause 6.4.5.6.4.
9. This process is same as process 5 of clause 6.4.5.6.5.
10. This process is same as process 6 of clause 6.4.5.6.5.
11. This process is same as process 7 of clause 6.4.5.6.5.
12. This process is same as process 10 of clause 6.4.5.6.4.
13. This process is same as process 11 of clause 6.4.5.6.4.
14. This process is same as process 8 of clause 6.4.5.6.5.
15. This process is same as process 9 of clause 6.4.5.6.5.
16. This process is same as process 14 of clause 6.4.5.6.4.
17. This process is same as process 15 of clause 6.4.5.6.4.
18. This process is same as process 10 of clause 6.4.5.6.5.

19. This process is same as process 11 of clause 6.4.5.6.5.
20. This process is same as process 12 of clause 6.4.5.6.5.
21. This process is same as process 13 of clause 6.4.5.6.5.
22. This process is same as process 10 of clause 6.4.5.6.4.
23. This process is same as process 11 of clause 6.4.5.6.4.
24. This process is same as process 14 of clause 6.4.5.6.5.
25. This process is same as process 15 of clause 6.4.5.6.5.
26. This process is same as process 14 of clause 6.4.5.6.4.
27. This process is same as process 15 of clause 6.4.5.6.4.
28. This process is same as process 16 of clause 6.4.5.6.5.
29. This process is same as process 17 of clause 6.4.5.6.5.
30. This process is same as process 18 of clause 6.4.5.6.5.
31. This process is same as process 7 of clause 6.4.5.6.3.
32. This process is same as process 26 of clause 6.4.5.6.4.
33. This process is same as process 27 of clause 6.4.5.6.4.
34. This process is same as process 20 of clause 6.4.5.6.5.
35. This process is same as process 21 of clause 6.4.5.6.5.
36. This process is same as process 22 of clause 6.4.5.6.5.
37. This process is same as process 30 of clause 6.4.5.6.4.
38. This process is same as process 31 of clause 6.4.5.6.4.
39. This process is same as process 23 of clause 6.4.5.6.5.
3. This process is same as process 24 of clause 6.4.5.6.5.

## 6.4.6 SDP

### 6.4.6.1 General

RESPECT applies SDP for describing multimedia sessions. This clause describes the usage of SDP for RESPECT.

The SDP description in the RESPECT message is required to comply with SDP specification (IET RFC 8866 [53]) and JSEP specification (IETF RFC 8829 [47]).

### 6.4.6.2 Session-Level Section

#### 6.4.6.2.1 General

This clause describes the usage of session-level section of the session description.

"v=" line (protocol version), "o=" line (origin), "s=" line (session name), "a=group" line (group attribute) which is set to "BUNDLE", "a=ice-ufrag" line and "a=ice-pwd" line is allowed to be included as Session-Level Section of the session description.

Other lines are ignored if the received network does not allow to use the line(s) at the UNI. The support of other lines are based on by-lateral agreements at the NNI.

#### 6.4.6.2.2 Protocol Version ("v=")

The "v=" line (protocol version) is required to be set to "0" in accordance with IETF RFC 8866 [53].

#### 6.4.6.2.3 Origin ("o=")

The "o=" line (origin) is required to be set as described in IETF RFC 8866 [53] and IETF RFC 8829 [47].

"username" part is required to be set to "-" as described in clause 5.2 of IETF RFC 8866 [53].

"sess-id" is required to be set to the UTC timestamp as described in clause 5.2 of IETF RFC 8866 [53] or a 64-bit quantity with the highest bit set to zero and the remaining 63 bits being cryptographically random as described in clause 5.2.1 of IETF RFC 8829 [47].

"nettype", "addrtype" and "unicast-address" are recommended to set to "IN IP4 0.0.0.0" as described in IETF RFC 8829 [47].

#### 6.4.6.2.4 Session Name ("s=")

The "s=" line (session name) is required to be set to " " (single space) or "-" as described in clause 5.3 of IETF RFC 8866 [53].

#### 6.4.6.2.5 Time Active ("t=")

The "t=" line (time active) is recommended to be set to "0 0"as described in clause 5.2.1 of IETF RFC 8829 [47].

#### 6.4.6.2.6 Group Attribute ("a=group")

The "a=group" line (group attribute) which is set to "BUNDLE" is required to include the mid identifiers of each "m=" section as described in SDP grouping framework specification (IETF RFC 5888 [25]) and BUNDLE mechanism (IETF RFC 8859 [51] and IETF RFC 9143 [59]).

All media descriptions in the SDP description are required to be treated as the target of the BUNDLE mechanism.

#### 6.4.6.2.7 "ice-ufrag" and "ice-pwd" attributes

The "a=ufrag" line is required to contain fragment of ICE username as described in IETF RFC 8839 [50]. The "a=ice-pwd" line is required to contain ICE password as described in IETF RFC 8839 [50]. These values of the attributes are used for ICE mechanism as specified in IETF RFC 8445 [39].

### 6.4.6.3 Media description

#### 6.4.6.3.1 General

For audio or video stream, "m=" is set as follows:

- <media> is set to "audio" or "video".
- <proto> is set to "UDP/TLS/RTP/SRVPF" (IETF RFC 5764 [24D]).

For Data channel. "m=" is set as follows:

- <media> is set to "application"
- <proto> is set to "UDP/DTLS/SCTP" as described in IETF RFC 8841 [50A].
- <fmt> is recommended to be set to "webrtc-datachannel" as described in IETF RFC 8841 [50A]. However, RESPECT endpoint is required to be identify the media type from <media> and <proto>, even if the <fmt> is not set to "webrtc-datachannel".

The RESPECT endpoint (UE) sets <port> to "9" (indicating there are no ICE candidate at that time), if the media is not invalid.

The RESPECT endpoint in the network sets <port> to the transport port to which the media/data stream is received or sets <port> to "0" and include "a=bundle-only" line.

If the <port>of the "m=" line in the SDP description for offer is set to "0", the <port>of the corresponding "m=" line in the SDP description for answer is required to be set to "0".

#### 6.4.6.3.2.2 Audio and video

Media description for audio/video stream is required to be set per media source, according to IETF RFC 8829 [47].

#### 6.4.6.3.2.3 Data channel

Only one media description for Data Channel is allowed to be set in the SDP description.

#### 6.4.6.3.2.4 Disabling and (re-)enabling the media description

The RESPECT endpoint (AS) in the network is allowed to disable the media description by sending the SDP offer including a media description which does not include "a=bundle-only" line and sets the port number of the "m=" line to "0", as described in IETF RFC 3264 [21A].

The RESPECT endpoint (AS) is able to use the disabled media description for other purpose by sending the SDP offer including a media description which includes "a=bundle-only" line or sets the port number of the "m=" line to other than "0".

#### 6.4.6.3.2.5 Partial non-use of Media description

RESPECT endpoint (UE) is allowed to disable the specific media description by setting the port number of the "m=" line to "0" in the SDP answer (which does not include "a=bundle-only" line). However, The RESPECT endpoint (UE) is not allowed to enable the disabled media description.

The RESPECT endpoint is able to use "a-inactive" line for temporary suspending of the media description.

### 6.4.6.3.3 Connection Information ("c=")

"c=" line is set to the next line after "m=" line.

### 6.4.6.3.4 Media Stream Identification Attribute ("a=mid")

"a=mid" line is set into the media description. The label in the "a=mid" is referred by "a=group:BUNDLE" line in the session-level section. It is recommended to set the sequential number which start from "0" (i.e., "0", "1", "2" ...).

The label in "a-mid" line is also set to the RTP SDES header extension. This enables to identify the dependency between the RTP packet and the media description.

### 6.4.6.3.5 "candidate" Attribute ("a=candidate")

This attribute is set into SDP description according to IETF RFC 8445 [39].

### 6.4.6.3.6 "ice-lite" Attribute ("a=ice-lite")

This attribute is set into SDP description to indicate the RESPECT endpoint is ice lite implementation. In that case, an reachable IP address is set into the SDP description".

### 6.4.6.3.7 Attribute ("a=ice-options")

If the RESPECT endpoint is full implementation for ICE, this attribute is set to "ice2" or "trikle" depends on the supported ICE implementation.

#### 6.4.6.3.8 "ice-ufrag" and "ice-pwd" attributes ("a=ice-ufrag"/"a=ice-pwd")

"a=ice-ufrag" and "a=ice-pwd" line are able to be specified into media description. If these lines are set into both session-level section and media description, the lines in the media description take precedence.

#### 6.4.6.3.9 Attribute ("a=extmap")

This attribute indicates the RTP header extension for the media. RESPECT endpoint is required to support "urn:ietf:params:rtp-hdrext:sdes:mid" extension and set this extension to the media description.

#### 6.4.6.3.10 Attribute ("a=bundle-only")

If this attribute is set into the SDP media description for offer, it indicates that the corresponding media description is used only for BUNDLE mechanism. Regardless of this attribute is included in the SDP media description for offer or not, BUNDLE mechanism is required to be applied for any media description.

#### 6.4.6.3.11 Attribute ("a=rtcp-mux-only")

This attribute is used in an offer to indicate exclusive support of RTP/RTCP multiplexing for the RTP-based media associated with the SDP media description ("m=" line). Regardless of this attribute is included in the SDP media description for offer, or not, RESPECT endpoint is required to uses a single port for RTP and RTCP.

#### 6.4.6.3.12 Attribute ("a=rtcp-mux")

This attribute indicates the RESPECT endpoint uses a single port for RTP and RTCP. This attribute is required to be set into the media description.

#### 6.4.6.3.13 Attribute ("a=msid")

"a=msid" line is allowed to be set into the media description.

#### 6.4.6.3.14 Attribute ("a=ssrc")

"a=ssrc", "a=ssrc cname" and "a=ssrc msid" line are allowed to be set into the media description.

#### 6.4.6.3.15 Attribute ("a=sendrecv" / "a=sendonly" / "a=recvonly" / "a=inactive")

Audio stream and video stream are able to set these attributes to specify the direction of the media stream.

#### 6.4.6.3.16 Attribute ("a=setup")

This attribute is used for specifying the direction of the DTLS negotiation, it the media description is for Data Channel. Every offer is required to be set to "actpass". The direction of the DTLS negotiation is determined by the value included in the corresponding answer.

#### 6.4.6.3.17 Attribute ("a=fingerprint")

This attribute is user for indicating the fingerprint of the public key of the RESPECT endpoint for DTLS negotiation. This attribute needs to be set to all media description applies BUNDLE mechanism.

#### 6.4.6.3.18 Attribute ("a=rtpmap" / "a=fmtp")

This attribute specifies the information such as codec, corresponding the payload type in the "m=" line.

#### 6.4.6.3.19 Attribute ("a=dcmap")

This attribute specifies the stream number of the data stream of the Data Channel. This attribute is also used for set such as the label of the data stream. This attribute is specified in the offer by the network. The RESPECT endpoint (UE) is required to set the specified value in the answer, if included in the offer.

#### 6.4.6.3.20 Attribute ("a=sctp-port")

When the media type of the media description is "application" for Data Channel, "a=sctp-port" attribute is required to be specified.

#### 6.4.6.3.21 Attribute ("a=max-message-size")

When the media type of the media description is "application" for Data Channel, "a=max-message-size" attribute is specified by the network. The RESPECT endpoint (UE) is required to handle the attribute, if specified by the network.

#### 6.4.6.3.22 Attribute ("a=rtcp-rsize")

When the media type of the media description is "video", "a=rtcp-rsize" attribute is specified by the network. The RESPECT endpoint (UE) is required to handle the attribute, if specified by the network.

### 6.4.7 Solution evaluation

This solution proposes RESPECT as a signalling protocol which support the collaboration scenario 4 and collaboration scenario 3 specified in 3GPP TS 26.506 [12]. RESPECT is developed as the signalling protocol applicable to the RTC functional architecture and supports WebRTC technologies-based media handling by the signalling message.

Then, it is proposed to apply RESPECT as base of stage 3 work for specifying RTC signalling protocol supporting collaboration scenario 4 and collaboration scenario 3.

## 6.5 Solution #4: Functional requirements for U-Plane

### 6.5.1 Solution description

This solution addresses Key Issue #4.

This clause identifies requirements for U-Plane needed for WebRTC-based immersive RTC session management supporting inter-operator connection (i.e., collaboration scenario 4 in TS 26.506 [12]) based on the architecture described in clause 6.2. Figure 6.5.1-1 shows the U-Plane reference points on the derivative architecture. RTC-4m and RTC-Ym are focussed reference points of this document as described in clause 6.2.7.

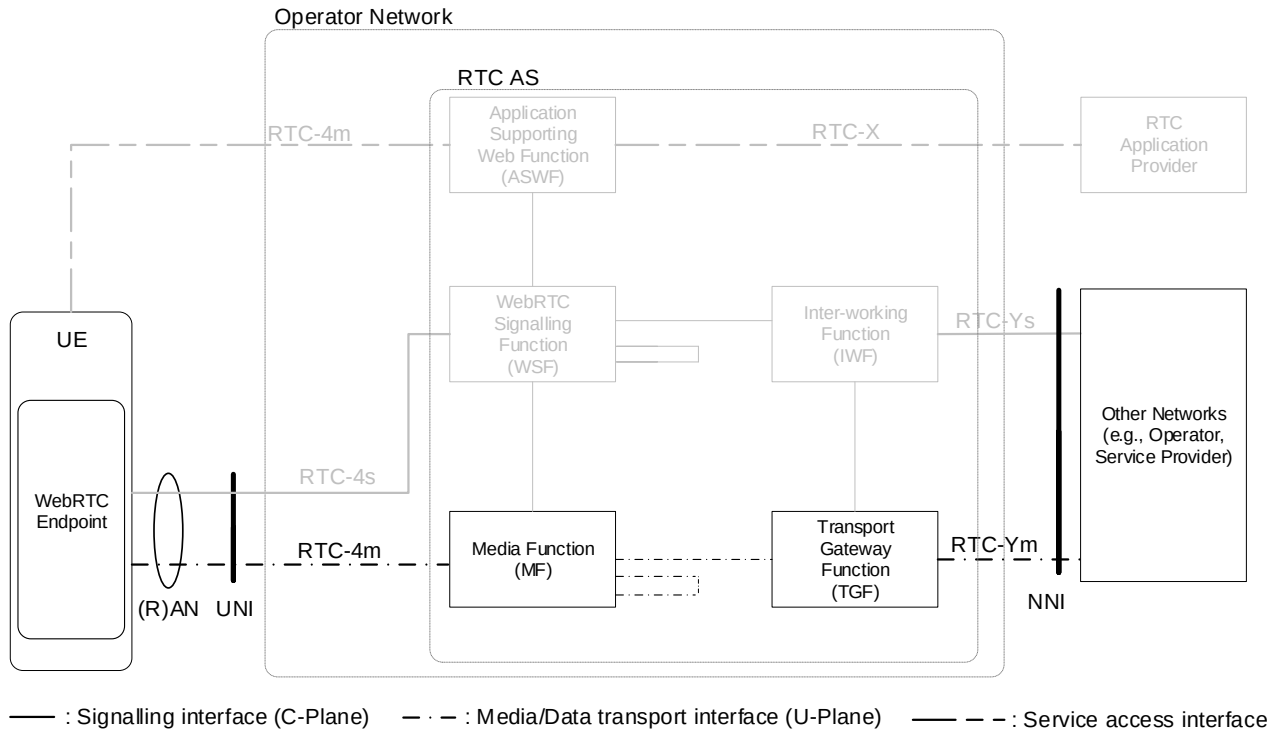


**Figure 6.5.1-1: Reference points for U-Plane**

## 6.5.2 Functional requirements for U-Plane interface

This clause identifies the functional requirements of RTC-4m and RTC-Ym reference point as U-Plane interface.

The derivative architecture supports collaboration scenario 3 and 4 defined in 3GPP TS 26.506 [12]. The requirements of RTC-4m on the derivative architecture are compliant with the requirements of RTC-4m specified in 3GPP TS 26.506 [12]. RTC-Ym is a new U-plane interface for WebRTC media transport between different operator's network or between an operator and service provider network. On the viewpoint of service interoperability, the requirements of RTC-Ym are required to be same as RTC-4m.

Functional requirements applied to both the RTC-4m and RTC-Ym are to transport:

- Media data transmitted over RTP;
- Application data transmitted using data channel; and
- Media related meta-data transmitted using data channel

NOTE: As RTC-Ym is the interface between the networks operating by two different operators (or an operator and a service provider) where the different policy/application can be adopted/provided; therefore, a bilateral agreement may be required.

## 6.5.3 Protocol stack

RTC-4m and RTC-Ym on the derivative architecture are U-Plane interfaces for WebRTC media transport. Then the protocol stack of RTC-4m and RTC-Ym conforms to the protocols specified in RFC 8835 [48]. This protocol stack is also applied for U-Plane interface in 3GPP TS 26.113 [10] which specified "enabler for Immersive Real-time Communication".

Detailed protocol stack for the U-Plane interface studied in this document is defined by selecting the protocol from the protocol stack specified in clause 13.1 of 3GPP TS 26.113 [10] in the corresponding normative work for stage 3 specification.

NOTE: The specification other than protocols (e.g., codec) is not referred.

### 6.5.4 Solution evaluation

The U-Plane functional requirements proposed in clause 6.5.2 and the U-Plane protocol stack proposed in clause 6.5.3 are appropriate for WebRTC media transport and aligned with 3GPP SA4 RTC specifications (i.e., 3GPP TS 26.506 [12] and 3GPP TS 26.113 [10]). Therefore, these are proposed as the U-Plane requirements in this document.

## 6.6 Solution #5: Service control API

### 6.6.1 Solution Description

This solution addresses Key Issue #5.

In this solution, procedures and APIs required for the following functions described in clause 5.6.3 and additional notification process between the CP and operator network are identified. These procedures and APIs are used for instructions and notifications between the service logic managers and the service logic enforcers.

- CRUD of RTC ID resource and associated properties
- User connection control using asserted identity
- Media data forwarding control (MDFC)

In the extension of the RTC architecture described in clause 6.2 of this document, the ASWF has the functionalities of conference session management and capability exposure to CP. Therefore, these API are provided through RTC-X between ASWF and CP, as illustrated in the Figure 6.6.1-1.

NOTE: RTC-X is expected to be added to the RTC architecture specified in 3GPP TS 26.506 [12]. However, the RTC architecture shares common elements with the media delivery architecture, as specified in 3GPP TS 26.510 [13]. Therefore, it is necessary to identify possible impacts of RTC-X against the common elements before normative works.

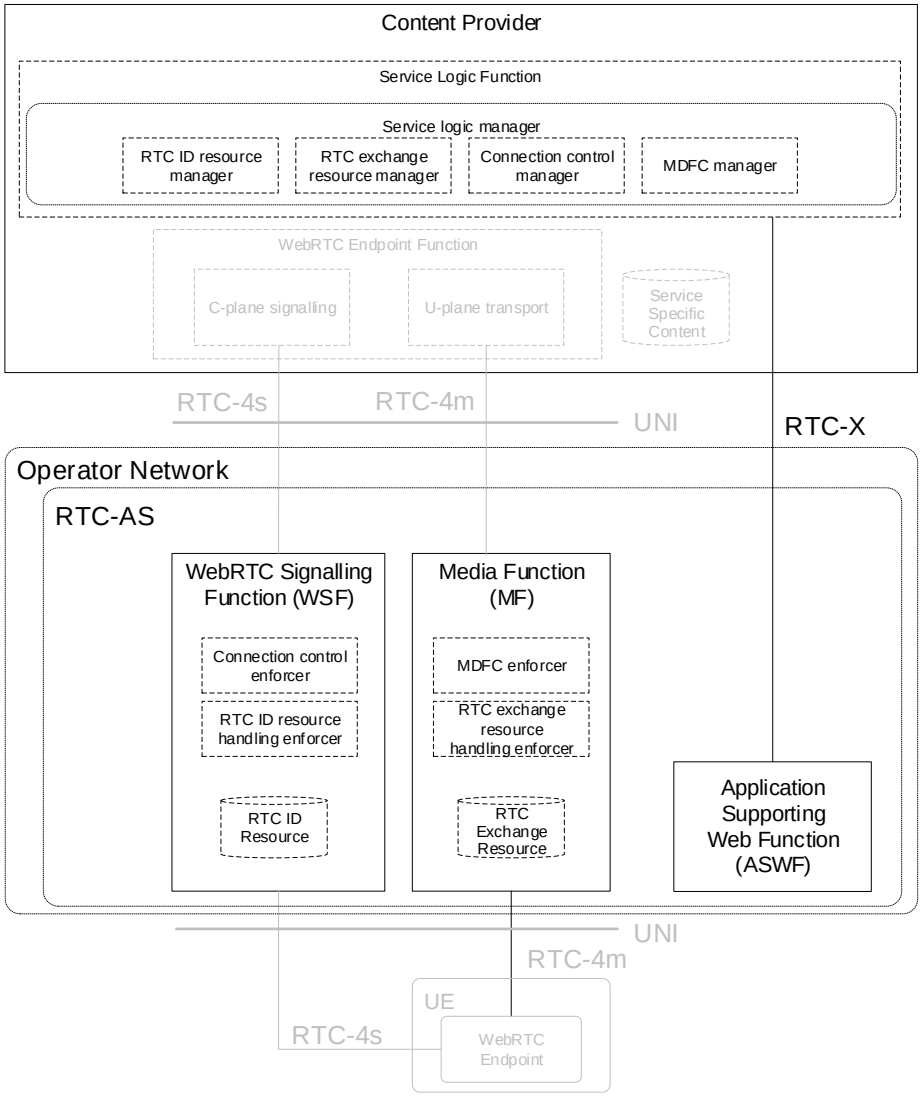


**Figure 6.6.1-1: Reference point representation of service control**

## 6.6.2 Procedures for service control

### 6.6.2.1 General

This clause identifies the procedures for the functions mentioned in clause 5.6.3.

These procedures refer to instructions and notifications performed over RTC-X between the service logic managers in CP and the service logic enforcers in the operator network.

The CRUD of RTC ID Resource are procedures for manipulation of resources. In these procedures, requests are always initiated proactively by the CP's service logic manager, and performed by the operator network's service logic enforcer.

In contrast, user connection control is performed in a callback procedure. The callback procedure is triggered by a new attempt of media session setup establishment from UE in an operator network and the operator network requests instruction from the CP along with the notification.

Regarding MDFC, it can be divided into two procedures. First, static rules associated with VR spaces or conference rooms are instructed from the CP to the operator network. Second, in cases where exceptional rule cannot be handled by pre-configured rules, the operator requests instructions from the CP with a notification through a callback procedure.

Other notifications of state changes in RTC ID resources can be performed as callback procedures, informing about the occurrence of events that are previously subscribed to.

Based on the above, all the procedure for service control in this document categorized into the following two types:

- **CP-oriented procedure**: procedures refer to the instructions by the CP's service logic manager to the operator network's service logic enforcer.
- **Callback procedure**: procedures refer to the notifications from an operator network's service logic enforcer to a CP's service logic manager regarding the occurrence of events CP has subscribed to (e.g., new attempt of media session establishment from UE). In the response to callback, CP could instruct the operator network's service logic enforcer to perform a certain process associated with the event.

### 6.6.2.2 CP-oriented procedure

#### 6.6.2.2.1 General

CP-oriented procedures provide instructions from the CP to the operator network. The procedures are listed in Table 6.6.2.2.1-1.

**Table 6.6.2.2.1-1: CP-oriented procedures**

| Procedures | Description | Section Defined |
|---|---|---|
| Create RTC ID resource | This procedure is used by CP to register the RTC ID resource and associated properties. | Clause 6.6.3.2 |
| Read RTC ID resource | This procedure is used by CP to retrieve the status of the RTC ID resource and associated properties. | |
| Update RTC ID resource | This procedure is used by CP to update the RTC ID resource and associated properties. | |
| Delete RTC ID resource | This procedure is used by CP to delete the RTC ID resource and associated properties. | |

#### 6.6.2.2.2 RTC ID resource management

##### 6.6.2.2.2.1 General

This procedure group provides the functionality for CP's RTC ID resource manager to instruct RTC ID resource management performed by operator network's RTC ID resource enforcer. It consists of four procedures providing the functionality of CRUD of RTC ID resource and associated properties.

The creation and deletion of VR space / conference room provided by CP are performed as registration and de-registration of RTC ID resources to operator network. An RTC resource ID, that is a property of an RTC ID resource used to identify the RTC ID resource, is represented as a URI. This URI consists of user part and host part (e.g., userinfo@host). The host part represents a CP providing services, and the user part represents individual VR space / conference room in the specific CP's service.

In addition, the associated configurations for individual VR space / conference room are executed by CRUD of properties associated with an RTC ID resource. The information of properties managed by this procedure could include:

- Expiration time: This property is used to set expiration time for a RTC resource ID. In the case of a conference service, it may be necessary to set an expiration time for the meeting. This expiration time is typically set when making a reservation for the meeting. It is commonly used to send notifications to participants at the start time, and to provide information to participants about the remaining time before the end time.
- Acceptance of new connections: This property is used to restrict new connections to a RTC resource ID. Depending on the RTC service, there may be a need to disallow any new connections to an RTC ID resource. For example, after the expiration of a meeting, a new participant can be disallowed to enter the meeting, while participants joined before the expiration of the meeting are allowed to continue communications.
- Permission list: This property is used to restrict session participation to specific UEs assigned certain IDs, enabling private sessions. When flexible connection control is required instead of static settings, the "user connection control using asserted identity" procedure in clause 6.6.2.5 will be used.
- Event subscription: This property is information about the type of subscribed events and the related callback URI(s). These events may include the new connection attempt from a UE to a specific RTC ID resource, and

completion of RTC exchange resource after a CP's RTC ID resource deletion request. Since these events are related to RTC ID resources, it is thought to be natural for CP to configure event subscription settings associated with this procedure. In accordance with this event subscription, callback procedures described in clause 6.6.2.3 are performed.

- Media data forwarding control configuration: This property is information about media data forwarding control. The role of the UE in the context of MDFC, labelling of media/data transmitted and received by the UE, and the routing rules for those media/data are configured by associating this property with the RTC ID resource.

#### 6.6.2.2.2.2 Create RTC ID resource

In this procedure, the RTC ID resource manager in the CP instructs the RTC ID resource enforcer in the operator network to register a new RTC ID resource and reflect the configurations based on the associated properties.

The register request from CP use HTTP POST method and may include properties associated with RTC resource ID such as RTC ID resource, the expiration time of RTC exchange resources, the acceptance of new connections, permission lists, event subscriptions and MDFC rules as described in clause 6.6.2.2.2.1.

After successful creation, the operator network will send 201 (Created) response including the created RTC ID resource URI in the Location header of accept response. This URI allows CP to identify the target resources for subsequent Update/Read/Delete procedures.

#### 6.6.2.2.2.3 Read RTC ID resource

In this procedure, the RTC ID resource manager in the CP instructs the RTC ID resource enforcer in the operator network to retrieve the configurations related to the registered RTC ID resource.

The read request from CP uses the HTTP GET method for this purpose.

After successful information retrieval, the operator network will respond with a 200 (OK) response, including the properties of the specified RTC ID resource.

#### 6.6.2.2.2.4 Update RTC ID resource

In this procedure, the RTC ID resource manager in the CP instructs the RTC ID resource enforcer in the operator network to update the configurations related to the registered RTC ID resource.

The update request from CP uses the HTTP PATCH method for this purpose and may include changed properties associated with the RTC ID resource.

After successful RTC ID resource update, the operator network will respond with a 200 (OK) response.

#### 6.6.2.2.2.5 Delete RTC ID resource

In this procedure, the RTC ID resource manager in the CP instructs the RTC ID resource enforcer in the operator network to delete the registered RTC ID resource and related configurations.

CP uses HTTP DELETE method for this purpose.

Upon accepting the request, the operator network will respond with a 202 (Accepted) response. The completion of the RTC ID resource deletion is notified through the callback procedure described in clause 6.6.2.3.2.

### 6.6.2.3 Callback procedure

#### 6.6.2.3.1 General

Callback procedures provide notifications from operator network to the CP for events occurrence corresponding to subscribed callback triggers. When notifying events, if additional instructions are required, the CP issues instructions in the response of this procedure. For each callback trigger, callback procedures are listed in Table 6.6.2.3.1-1.

The Callback URI registered or updated in RTC ID resource management procedures described in clause 6.6.2.2 is used as the destination of requests. The operator network uses HTTP POST method for this purpose. Upon receiving the notification, the CP responds with a 200 (OK) response.

Common information elements for callback request procedures could include RTC resource ID, callback trigger, and a sequence ID to identify the request.

**Table 6.6.2.3.1-1: Callback procedures**

| Procedures | Description | Section Defined |
|---|---|---|
| Notification of RTC ID resource deletion | This procedure is used by the OP to notify that the deletion of the RTC ID resource, as requested by the CP, has been completed. | Clause 6.6.3.3 |
| Notification of forced RTC ID resource deletion | This procedure is used by the OP to notify that the RTC ID resource has been forcefully deleted. | Clause 6.6.3.4 |
| Notification of RTC ID resource suspended | This procedure is used by the OP to notify that the RTC ID resource is temporarily unavailable. | Clause 6.6.3.5 |
| Notification of RTC ID resource resumed | This procedure is used by the OP to notify that the suspended RTC ID resource is available again. | Clause 6.6.3.6 |
| Notification of user call in requested | This procedure is used by the OP to notify that a new connection has been requested from the UE. | Clause 6.6.3.7 |
| Notification of user call in accepted | This procedure is used by the OP to notify that a new connection request from the UE has been accepted by operator network. | Clause 6.6.3.8 |
| Notification of user call in connected | This procedure is used by the OP to notify that a media session has been established as a result of a new connection request from the UE. | Clause 6.6.3.9 |
| Notification of user call disconnected | This procedure is used by the OP to notify that the media session of the UE has been disconnected. | Clause 6.6.3.10 |
| Notification of media routing query | This procedure is used by the OP to notify that additional media/data is added during the establishment or update of a UE's media session and the OP requests instructions from the CP. | Clause 6.6.3.11 |

### 6.6.2.3.2 Notification of RTC ID resource deletion

In this procedure, the RTC ID resource enforcer in the operator network notifies the CP's RTC ID resource manager about the completion of a RTC ID resource including the release of RTC exchange resource associated with the RTC ID resource.

It is expected that even if the "Delete RTC ID resource" request is acceptable for operator network, there may be situations where RTC exchange resources in RTC exchange enforcer (MF) associated with the RTC ID resource cannot be immediately released while UEs are actively participating in conference rooms. This procedure is necessary to indicate the completion of both RTC ID resource and the associated RTC exchange resource deletion after all active sessions have ended and corresponding RTC exchange resources have been released.

### 6.6.2.3.3 Notification of forced RTC ID resource deletion

In this procedure, the RTC ID resource enforcer in the operator network notifies the CP's RTC ID resource manager that the RTC ID resource has been forcefully deleted.

In cases where the RTC ID resource becomes unavailable independently of CP requests and is not expected to recover in the short term (e.g., due to network failures), this procedure is used for notification. After this notification, it is assumed that the CP cannot perform status retrieval, update, or deletion related to the respective resource.

### 6.6.2.3.4 Notification of RTC ID resource suspended

In this procedure, the RTC ID resource enforcer in the operator network notifies the CP's RTC ID resource manager that the RTC ID resource is temporarily unavailable.

When this procedure is performed, new connections to the corresponding RTC ID resource are not possible, and existing connections are expected to either be disconnected or in the process of being disconnected. Upon receiving this

notification, the CP is only capable of performing the deletion process described in clause 6.6.2.2.2.5 for the RTC ID resource.

#### 6.6.2.3.5 Notification of RTC ID resource resumed

In this procedure, the RTC ID resource enforcer in the operator network notifies the CP's RTC ID resource manager that the RTC ID resource has recovered from the suspended state previously notified in clause 6.6.2.3.4.

For the corresponding RTC ID resource, new connections are possible after recovery, but it is not guaranteed that the state will be the same as it was before entering the suspended state.

#### 6.6.2.3.6 Notification of user call in requested

In this procedure, the connection control enforcer in the operator network notifies the connection control manager in the CP that a UE has attempted a new media session setup. The CP provides instructions regarding the permissibility of this UE's connection in the response of this procedure, offering the functionality of User connection control using asserted identity.

As described in clause 5.6.3.3, the CP performs connection control based on its own user's subscriptions data and its own service requirements when a UE requests the media session setup to the registered RTC ID resource. Therefore, when UE requests a media session setup, the operator queries the CP to determine whether UE's connection is permissible. Since UE's media session setup requests are made through RTC-4s to WSF, the connection control enforcer in WSF handles the connection control.

The information elements for the request in this procedure would additionally include the CP-provided ID indicating the originating UE and media information in the media session setup request. CP can determine the validity of UE's media session setup request based on the CP-provided ID and subscription data managed by CP.

The response in this procedure would involve the acceptance of connection request by the UE. Furthermore, along with the determination of connection permissibility, CP may indicate the role of the UE in the context of MDFC. As a property of the RTC ID resource, it is possible to statically configure which UE roles are associated with specific RTC user IDs. However, if a connection attempt is made with an RTC user ID that has not been pre-configured, the response of this procedure is used to convey information about connection permissibility and the role of UE.

#### 6.6.2.3.7 Notification of user call in accepted

In this procedure, the connection control enforcer in the operator network notifies the CP's connection control manager that a UE has attempted to connect, and the connection request has been accepted. The CP provides instructions regarding the permissibility of this UE's connection in the response of this procedure, offering the functionality of User connection control using asserted identity.

The content of the request-response for this procedure follows clause 6.6.2.3.6, excluding the callback trigger.

#### 6.6.2.3.8 Notification of user call in connected

In this procedure, the connection control enforcer in the operator network notifies the CP's connection control manager that a UE has attempted to connect, and the media session has been established.

#### 6.6.2.3.9 Notification of user call disconnected

In this procedure, the connection control enforcer in the operator network notifies the CP's connection control manager that the established media session has been disconnected.

#### 6.6.2.3.10 Notification of media routing query

In this procedure, the media data forwarding control enforcer in the operator network notifies the CP's media data forwarding control manager that a UE has attempted to connect or update media session, and there is additional media that cannot be handled by pre-configured rules. The CP provides instructions regarding the requested media/data connections in the response of this procedure.

As described in clause 5.6.3.5, the CP controls how individual RTC media/data are connected to the endpoints of specific UE or WebRTC Endpoint Functions in response to a media session setup request or media update for a registered RTC ID resource from UE or WebRTC Endpoint Functions.

In many RTC services, participating UEs can be classified into several groups that share same MDFC patterns based on their roles (e.g., hosts and guests in meetings, broadcaster and audiences in streaming). By describing MDFC patterns as presets for each role, efficient representation of MDFC can be achieved. Additionally, by refining the classification of roles, the MDFC can be attributed to the "copy to all" pattern, such as forwarding the RTC media/data to all terminals except the sender, resembling the common SFU (Selective Forwarding Unit) forwarding process.

Therefore, in MDFC, the preset roles for UE, labelling of media/data, and routing rules are configured as associated properties at the time of RTC ID resource registration. Additionally, the assignment of UE roles during connection is either pre-configured as a property of the RTC ID resource or instructed by the CP during the determination of connection permissibility as described in clause 6.6.2.3.6.

In exceptional cases, such as private one-to-one RTC media/data forwarding, individual queries to the MDFC manager are necessary. When such specific inquiries are required, the MDFC enforcer uses this procedure to query the MDFC manager.

As additional information elements in the request of this procedure, CP-provided ID indicating the originating UE, the role of the UE, the number of UEs with the same role, and media information included in the media session request are expected to be included. The additional information in the response of this procedure conforms to clause 6.6.2.3.6.

## 6.6.3 Service Control APIs

### 6.6.3.1 General aspects of service control API

#### 6.6.3.1.1 Usage of HTTP

For service control API, ASWF and CP are expected to support HTTP/2 (IETF RFC 9113 [57]).

#### 6.6.3.1.2 Content type

The bodies of HTTP request and successful HTTP responses are required to be encoded in JSON format (IETF RFC 8259 [36]). The MIME media type that is required to be used within the related Content-Type header field is "application/json", unless described otherwise in the API definition.

When the "Problem Details" JSON object (IETF RFC 7807 [32A]) is used to indicate error details in an HTTP response, the Content-Type is set to "application/problem+json".

#### 6.6.3.1.3 URI structure

##### 6.6.3.1.3.1 Resource URI structure

API URIs for service control APIs are expected to be:

**{apiRoot}**/<**apiVersion**>/<**apiName**>/<**apiSpecificResourceUriPart**>/

"apiRoot" is configured following clause 4.4.1 of 3GPP TS 29.501 [17]. "apiVersion" is dependent on APIs but expected to be "v1" in the first version. "apiName" is also dependent on APIs. "apiSpecificSuffixes" can be set depending on resource definitions of each API.

##### 6.6.3.1.3.2 Custom operations URI structure

The custom operation is the operations that is non-idempotent and not a CREATE on a collection as described in Annex C of 3GPP TS 29.501 [17].

The URI of a custom operation which is associated with a resource is required to have the following structure:

**{apiRoot}/<apiVersion>/<apiName>/<apiSpecificResourceUriPart>/<custOpName>**

Custom operations can also be associated with the service instead of a resource. The URI of a custom operation which is not associated with a resource is required to have the following structure:

**{apiRoot}/<apiVersion>/<apiName>/<custOpName>**

In the above URI structures, "apiRoot", "apiName", "apiVersion" and "apiSpecificResourceUriPart" are as defined in clause 6.6.3.1.3.1 and "custOpName" represents the name of the custom operation as defined in clause 5.1.3.2 of 3GPP TS 29.501 [17].

#### 6.6.3.1.4 Error handling

Response bodies for error handling are described in table 6.6.3.1.5-1. The HTTP client is expected to handle the described status codes and 'ProblemDetails' JSON (with the "application/problem+json" set in the Content-Type header) appropriately.

**Table 6.6.3.1.4-1: Response bodies supported for responses to all requests**

| | Data type | Cardinality | Response Codes | Remarks | Applied Methods |
|---|---|---|---|---|---|
| **Response body** | ProblemDetails | 1 | 403 Forbidden | This represents the case when the server is able to understand the request but unable to fulfil the request due to errors. | GET, POST, PATCH, DELETE |
| | ProblemDetails | 1 | 404 Not Found | The resource URI was incorrect. | GET, POST, PATCH, DELETE |
| | ProblemDetails | 1 | 500 Internal Server Error | The server encountered an unexpected condition that prevented it from fulfilling the request. | GET, POST, PATCH, DELETE |
| | ProblemDetails | 1 | 503 Service Unavailable | The server is unable to handle the request. | GET, POST, PATCH, DELETE |

#### 6.6.3.1.5 HTTP headers

This API follows the HTTP Semantics (IETF RFC 9110 [54]), and HTTP headers are set accordingly.

**Table 6.6.3.1.5-1: HTTP headers used in service control API**

| Name | Reference | Description |
|---|---|---|
| Content-Type | IETF RFC 9110 [54] | When setting JSON in the message body of an HTTP response, "application/json" is used and when setting Problem Details "application/problem+json" is used. If the message body is not set, this header is not used. |
| Location | IETF RFC 9110 [54] | This header is used in 201 (Created) response upon successful RTC ID resource registration. The URI set in this header is used as a path parameter during RTC ID resource state retrieval, state modification, and deletion. |

### 6.6.3.2 RTC ID resource management API

#### 6.6.3.2.1 API URI

The HTTP requests from CP's RTC ID resource manager have the structure of the request URI as indicated in clause 6.6.3.1.3. However, the following is applied for the base path for this operation:

- <apiVersion>: "v1"

- <apiName>: "rtc-service-control"

#### 6.6.3.2.2 Resources

##### 6.6.3.2.2.1 Overview

This clause describes the structure for the resource URIs and the resources and methods used for the service.

Figure 6.6.3.2.2.1-1 depicts the resource URIs structure for the RTC ID resource management API.

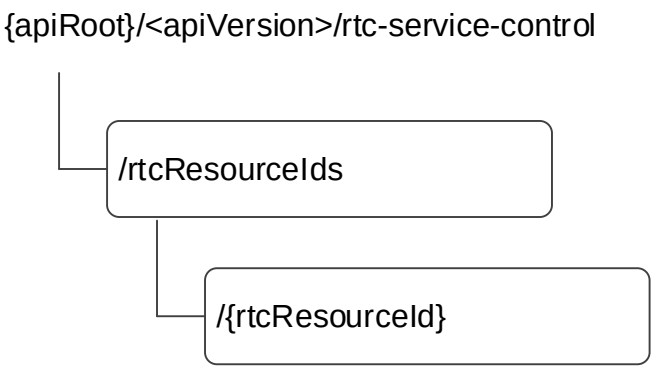


**Figure 6.6.3.2.2.1-1: Resource URI structure of the CRUD of RTC ID resource and related configuration operation**

Table 6.6.3.2.2.1-1 provides an overview of the resources and applicable HTTP methods.

**Table 6.6.3.2.2.1-1: Resources and methods overview**

| Resource name | Resource URI | HTTP method or custom operation | Description |
|---|---|---|---|
| RTC ID resources | /rtcResourceIds | POST | Create an RTC ID resource |
| Individual RTC ID resource | /rtcResources/{rtcResourceId} | DELETE | Delete the RTC ID resource specified by {rtcResourceId} |
| | | PATCH | Update the status of the RTC ID resource specified by {rtcResourceId} |
| | | GET | Get the status of the RTC ID resource specified by {rtcResourceId} |

##### 6.6.3.2.2.2 Resource: RTC ID resources

###### 6.6.3.2.2.2.1 Description

RTC ID resources represents the all RTC ID resources registered to RTC ID resource enforcer.

###### 6.6.3.2.2.2.2 Resource Definition

Resource URI : {apiRoot}/<apiVersion>/rtc-service-control/rtcResourceIds

This resource is required to support the resource URI variable defined in table 6.6.3.2.2.2.2-1.

**Table 6.6.3.2.2.2.2-1: Resource URI variables for this resource**

| Name | Data Type | Definition |
|---|---|---|
| apiRoot | string | See clause 6.6.3.1.3 |

#### 6.6.3.2.2.3 Resource Standard Methods

##### 6.6.3.2.2.3.1 POST

This method is required to support the URI query parameters described in table 6.6.3.2.2.3.1-1.

**Table 6.6.3.2.2.3.1-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.2.2.3.1-2 and the response data structure and response codes described in table 6.6.3.2.2.3.1-3.

**Table 6.6.3.2.2.3.1-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcResourceRegReq | M | 1 | Information for RTC ID resource to register (e.g., RTC resource ID, callback URI) |

**Table 6.6.3.2.2.3.1-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| n/a | | | 201 Created | RTC ID resource in the corresponding request is successfully created.<br>The URI of the created resource is required to be returned in the "Location" HTTP header. |

**Table 6.6.3.2.2.3.1-4: Headers supported by the 201 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| Location | String | M | 1 | Contains the URI of the newly created resource. |

### 6.6.3.2.2.3 Resource: Individual RTC ID resource

#### 6.6.3.2.2.3.1 Description

The individual RTC ID resource represents the individual RTC ID resource registered to RTC ID resource enforcer.

#### 6.6.3.2.2.3.2 Resource Definition

Resource URI : {apiRoot}/<apiVersion>/rtc-service-control/rtcResourceIds/{rtcResourceId}

This resource is required to support the resource URI variables defined in table 6.6.3.2.2.3.2-1.

**Table 6.6.3.2.2.3.2-1: Resource URI variables for this resource**

| Name | Data Type | Definition |
|---|---|---|
| apiRoot | string | See clause 6.6.3.1.3 |
| rtcResourceId | string | String identifying an individual RTC ID resource |

#### 6.6.3.2.2.3.3 Resource Standard Methods

##### 6.6.3.2.2.3.3.1 DELETE

This method is required to support the URI query parameters described in table 6.6.3.2.2.3.3.1-1.

**Table 6.6.3.2.2.3.3.1-1: URI query parameters supported by the DELETE method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.2.2.3.3.1-2 and the response data structure and response codes described in table 6.6.3.2.2.3.3.1-3.

**Table 6.6.3.2.2.3.3.1-2: Data structures supported by the DELETE Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| n/a | | | |

**Table 6.6.3.2.2.3.3.1-3: Data structures supported by the DELETE Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| n/a | | | 202 Accepted | RTC ID resource deletion is accepted. The completion of resource deletion is expected to be notified by callback operation described in clause 6.6.3.3. |

**Table 6.6.3.2.2.3.3.1-4: Headers supported by the 202 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

##### 6.6.3.2.2.3.3.2 PATCH

This method is required to support the URI query parameters described in table 6.6.3.2.2.3.3.2-1.

**Table 6.6.3.2.2.3.3.2-1: URI query parameters supported by the PATCH method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.2.2.3.3.2-2 and the response data structure and response codes described in table 6.6.3.2.2.3.3.2-3.

**Table 6.6.3.2.2.3.3.2-2: Data structures supported by the PATCH Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcResourceModReq | M | 1 | Update of RTC ID resource |

**Table 6.6.3.2.2.3.3.2-3: Data structures supported by the PATCH Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| n/a | | | 200 OK | Update of RTC ID resource is successfully completed. |

**Table 6.6.3.2.2.3.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

#### 6.6.3.2.2.3.3.3 GET

This method is required to support the URI query parameters described in table 6.6.3.2.2.3.3.3-1.

**Table 6.6.3.2.2.3.3.3-1: URI query parameters supported by the GET method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.2.2.3.3.3-2 and the response data structure and response codes described in table 6.6.3.2.2.3.3.3-3.

**Table 6.6.3.2.2.3.3.3-2: Data structures supported by the GET Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| n/a | | | |

**Table 6.6.3.2.2.3.3.3-3: Data structures supported by the GET Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcResourceStatRes | M | 1 | 200 OK | Requested RTC ID resource is successfully retrieved. |

**Table 6.6.3.2.2.3.3.3-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

### 6.6.3.2.3 Data model

#### 6.6.3.2.3.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.2.3.1-1 depicts the data types defined specifically for the RTC ID resource management API service.

**Table 6.6.3.2.3.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| rtcResourceRegReq | Clause 6.6.3.2.3.2.1 | RTC resource create request body | |
| rtcResourceModReq | Clause 6.6.3.2.3.2.2 | RTC resource update request body | |
| rtcResourceStatRes | Clause 6.6.3.2.3.2.3 | RTC resource read response body | |
| callbackInformation | Clause 6.6.3.2.3.2.4 | Event subscription for callbacks | |
| timeRange | Clause 6.6.3.2.3.2.5 | Time range | |
| mediaRouting | Clause 6.6.3.2.3.2.6 | MDFC rule | |
| rtcUserStatus | Clause 6.6.3.2.3.2.7 | Connected RTC user status | |
| ueRoleElem | Clause 6.6.3.2.3.2.8 | UE Role information | |
| accessControl | Clause 6.6.3.2.3.2.9 | Access control information | |
| anonymityControl | Clause 6.6.3.2.3.2.10 | ID anonymity control information | |
| mc | Clause 6.6.3.2.3.2.11 | Labels for media channel group | |
| dc | Clause 6.6.3.2.3.2.12 | Labels for data channel group | |
| mcGroupLabelElem | Clause 6.6.3.2.3.2.13 | Media channel group definition | |
| mediaDesc | Clause 6.6.3.2.3.2.14 | Media description | |
| codecElem | Clause 6.6.3.2.3.2.15 | Codec information | |
| mediaAttributeElem | Clause 6.6.3.2.3.2.16 | Media attribute | |
| extmapElem | Clause 6.6.3.2.3.2.17 | RTP header extension | |
| setupControl | Clause 6.6.3.2.3.2.18 | Media negotiation control information | |
| iniOffer | Clause 6.6.3.2.3.2.19 | Initial offer control information | |
| iniAnswer | Clause 6.6.3.2.3.2.20 | Initial answer control information | |
| subOffer | Clause 6.6.3.2.3.2.21 | Subsequent offer control information | |
| subAnswer | Clause 6.6.3.2.3.2.22 | Subsequent answer control information | |
| dcGroupLabelElem | Clause 6.6.3.2.3.2.23 | Data channel group definition | |
| routingRuleElem | Clause 6.6.3.2.3.2.24 | Rule entry of media routing | |
| routingElem | Clause 6.6.3.2.3.2.25 | Media routing pipeline element | |

Table 6.6.3.2.3.1-2 describes data types re-used by the RTC ID resource management API service.

**Table 6.6.3.2.3.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| n/a | | | |

6.6.3.2.3.2 Structured data types

6.6.3.2.3.2.1 Type: rtcResourceRegReq

**Table 6.6.3.2.3.2.1-1: Definition of type rtcResourceRegReq**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| rtcResourceId | string | M | 1 | RTC resource ID for the RTC ID resource to be created, based on the instruction by the RTC ID resource manager. | |
| callback | callbackInformation | O | 0..1 | Callback information associated with the RTC ID resource. | |
| validityPeriod | timeRange | O | 0..1 | The period where the RTC ID resource is available. | |
| newConnectionEnabled | boolean | O | 0..1 | Set to true indicate new connection can be established. Otherwise set to false. | |
| mediaRouting | mediaRouting | O | 0..1 | Configuration for media data forwarding control. If not set, all media and data are forwarded to all other endpoints. | |

6.6.3.2.3.2.2 Type: rtcResourceModReq

**Table 6.6.3.2.3.2.2-1: Definition of type rtcResourceModReq**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| callback | callbackInformation | O | 0..1 | Callback information associated with the RTC ID resource. | |
| validityPeriod | timeRange | O | 0..1 | The period where the RTC ID resource is available. | |
| newConnectionEnabled | boolean | O | 0..1 | Set to true indicate new connection can be established. Otherwise set to false. | |
| mediaRouting | mediaRouting | O | 0..1 | Configuration for MDFC. If not set, all media and data are forwarded to all other endpoints. | |

6.6.3.2.3.2.3 Type: rtcResourceStatRes

**Table 6.6.3.2.3.2.3-1: Definition of type rtcResourceStatRes**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| callback | callbackInformation | O | 0..1 | Callback information associated with the RTC ID resource. | |
| validityPeriod | timeRange | O | 0..1 | The period where the RTC ID resource is available. | |
| newConnectionEnabled | boolean | O | 0..1 | Set to true indicate new connection to the RTC ID resource can be established. Otherwise set to false. | |
| mediaRouting | mediaRouting | O | 0..1 | Configuration for MDFC. If not set, all media and data are forwarded to all other endpoints. | |
| rtcUserStatusList | array(rtcUserStatus) | M | 0..N | Status list of connected RTC users. | |

6.6.3.2.3.2.4 Type: callbackInformation

**Table 6.6.3.2.3.2.4-1: Definition of type callbackInformation**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| callbackUri | string | M | 1 | Callback URI assigned to the RTC ID resource. | |
| triggerList | array(string) | M | 0..N | Subscribed event triggers. Callback triggers are described in Table 6.6.3.2.3.2.4-2. | |
| bundleDuration | uint32 | O | 0..1 | Set the duration in seconds for bundling callbacks. If set to zero in update request, the callback bundling is released. | |

**Table 6.6.3.2.3.2.4-2: Definition of callback trigger**

| Callback trigger name | Description |
|---|---|
| "call.in.requested" | Callback is triggered when a new media session setup request to the related RTC ID resource is attempted. |
| "call.in.accepted" | Callback is triggered when a new media session setup request to the related RTC ID resource is accepted. |
| "call.in.connected" | Callback is triggered when a new media session setup request to the related RTC ID resource is connected and all media routings are completed. |
| "call.out.accepted" | Callback is triggered when a new instruction from CP that initiate call out session from the related RTC ID resource is accepted. |
| "call.out.connected" | Callback is triggered when a new instruction from CP that initiate call out session from the related RTC ID resource is completed. |
| "call.disconnected" | Callback is triggered when an existing media session is disconnected. |
| "call.media.requested" | Callback is triggered when operator network request CP's instruction regarding routing rules for added media. |
| "resource.deleted" | Callback is triggered when RTC ID resource is deleted in response to a resource delete request. |
| "resource.deletedForced" | Callback is triggered when RTC ID resource is deleted without a resource delete request. (e.g., network failure) |
| "resource.suspended" | Callback is triggered when RTC ID resource is temporarily unavailable. |
| "resource.resumed" | Callback is triggered when RTC ID resource is recovered from suspended state. |

6.6.3.2.3.2.5 Type: timeRange

**Table 6.6.3.2.3.2.5-1: Definition of type timeRange**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| startTime | string | M | 1 | String formatted time that indicates the absolute start time of the time range. | |
| endTime | string | M | 1 | String formatted time that indicates the absolute end time of the time range. | |

6.6.3.2.3.2.6 Type: mediaRouting

**Table 6.6.3.2.3.2.6-1: Definition of type mediaRouting**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| ueRole | array(ueRoleElem) | M | 0..N | UE's role presets for MDFC. | |
| mcGroupLabel | array(mcGroupLabelElem) | O | 0..N | Label definition for media channel group label. | |
| dcGroupLabel | array(dsGroupLabelElem) | O | 0..N | Label definition for data channel group label. | |
| routingRule | array(routingRuleElem) | O | 0..N | Routing rules for individual media and data. If not set, all media is forwarded to all other endpoints. | |
| routingMode | string | O | 0..1 | If "callback" is set, all media routing is controlled by CP's instruction in callback. If "auto" is set, the operator network performs media routing control and callback for requesting CP's instruction will not be triggered. If not set, "auto" is applied. | |

6.6.3.2.3.2.7 Type: rtcUserStatus

**Table 6.6.3.2.3.2.7-1: Definition of type rtcUserStatus**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| rtcUserId | string | M | 1 | RTC User ID for UE or WebRTC Endpoint Function. | |
| roleName | string | M | 1 | Role name associated with the RTC User ID. This role is related to the routing rules for media data forwarding control. | |

6.6.3.2.3.2.8 Type: ueRoleElem

**Table 6.6.3.2.3.2.8-1: Definition of type ueRole**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| roleName | string | M | 1 | Role name associated with UEs or WebRTC Endpoint Functions. | |
| numMax | integer | O | 0..1 | Maximum number of UEs that have the role represented by "roleName" and are connected to the RTC ID resource. If not set, maximum number is unlimited. | |
| accessControl | accessControl | O | 0..1 | Information for connection control. If not set, all connections are permitted unconditionally. | |
| anonimityControl | anonimityControl | O | 0..1 | Information for anonymity control of connecting UE. | |
| mc | array(mc) | O | 0..N | Information list of media channels related to the role. | |
| dc | array(ds) | O | 0..N | Information list of data streams related to the role. | |

6.6.3.2.3.2.9 Type: accessControl

**Table 6.6.3.2.3.2.9-1: Definition of type accessControl**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| allow | array(string) | O | 0..N | Originating RTC user ID list for UEs that are unconditionally permitted to connect. If set "#any", connection requests with/without any originating RTC user ID are permitted. If set "anonymous", connection requests without originating RTC user ID are permitted. | |
| deny | array(string) | O | 0..N | Originating RTC user ID list for UEs that are unconditionally prohibited to connect. If set "#any", connection requests with/without any originating RTC user ID are prohibited. If set "anonymous", connection requests without originating RTC user ID are prohibited. | |
| callback | array(string) | O | 0..N | Originating RTC user ID list for UEs that are dynamically permitted to connect based on CP's instruction with "call.in.requested" callback. "#any" and "anonymous" are treated as in the case of "allow". | |

6.6.3.2.3.2.10 Type: anonymityControl

**Table 6.6.3.2.3.2.10-1: Definition of type anonymityControl**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| oIdAllow | array(string) | O | 0..N | Originating RTC user ID list for UEs that unconditionally disclose their ID in signalling and API. If set "#any", connection requests with any originating RTC user ID is disclosed. | |
| oIdDeny | array(string) | O | 0..N | Originating RTC user ID list for UEs that unconditionally conceal their ID in signalling and API. If set "#any", connection requests with any originating RTC user ID is concealed. | |
| oIdCallback | array(string) | O | 0..N | Originating RTC user ID list for UEs that dynamically disclose their ID in signalling and API based on CP's instruction with "call.in.requested" callback. "#any" are treated as in the case of "oIdAllow". | |

6.6.3.2.3.2.11 Type: mc

**Table 6.6.3.2.3.2.11-1: Definition of type mc**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| mcGroupLabel | string | M | 1 | Label name specifying the media channel group to which the UE's media belongs. | |

6.6.3.2.3.2.12 Type: dc

**Table 6.6.3.2.3.2.12-1: Definition of type dc**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| dcGroupLabel | string | M | 1 | Label name specifying the data channel group to which the UE's data channel belongs. | |

6.6.3.2.3.2.13 Type: mcGroupLabelElem

**Table 6.6.3.2.3.2.13-1: Definition of type mcGroupLabelElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| mcGroupName | string | M | 1 | Name of media channel group. It is required to be unique in a specific RTC ID resource. | |
| numMax | uint32 | O | 0..1 | Maximum number of media channels that have the label indicated in mc. If not set, the number is unlimited. | |
| numMinPerUe | uint32 | O | 0..1 | Minimum number of media channels that have the label indicated in mc and belong to a specific UE. If not set, zero is applied. | |
| numMaxPerUe | uint32 | O | 0..1 | Maximum number of media channels that have the label indicated in mc and belong to a specific UE. If not set, the number is unlimited. | |
| mediaDesc | mediaDesc | M | 1 | Media description for the media channel group. | |
| setupControl | setupControl | M | 1 | Information for media setup control. | |

6.6.3.2.3.2.14 Type: mediaDesc

**Table 6.6.3.2.3.2.14-1: Definition of type mediaDesc**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| media | string | M | 1 | Media type. "audio" or "video" is set. | |
| codec | array(codecElem) | M | 1..N | Information for supported codec for this media channel. | |
| mediaAttribute | array(mediaAttributeElem) | O | 0..N | Information for media attribute. | |
| direction | string | M | 1 | Media direction. If "upstream" is set, the media is uni-directional from network to UE (recvonly in network's viewpoint). If "downstream" is set, the media is uni-directional from UE to network (sendonly in network's viewpoint). If "bidirectional" is set, the media is bi-directional (equal to sendrecv). If "inactive" is set, media forwarding is temporarily unavailable. | |

6.6.3.2.3.2.15 Type: codecElem

**Table 6.6.3.2.3.2.15-1: Definition of type codecElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| encodingName | string | M | 1 | String value following "encoding-name" specified in IETF RFC 8866 [53]. If any codec is allowed, only one codecElem is set and encodingName is required to be "#any". If specific codecs are set, network's initial offer include all listed codecs and. When UE send offer, codecs set in this value are allowed. | |
| clockRate | uint32 | O | 0..1 | String value following "clock-rate" specified in IETF RFC 8866 [53]. | |
| encodingParams | uint32 | O | 0..1 | String value following "encoding-params" specified in IETF RFC 8866 [53]. If specific value is set to this key, clockRate is required to be set specific value. An offer from network has this encoding-params and UE's offer with different encoding-params will be denied. If this key is not set, network's offer has no encoding-params and UE's offer with encoding-params will be denied. | |
| formatSpecificValue | string | O | 0..1 | String value following "format-specific-value" specified in IETF RFC 8866 [53]. An offer from network has a=fmtp line based on this value. If this key is not set, network's offer does not have a=fmtp line. Regardless of this key, a=fmtp line in offers from UE are not checked. | |

6.6.3.2.3.2.16 Type: mediaAttributeElem

**Table 6.6.3.2.3.2.16-1: Definition of type mediaAttributeElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| ptime | uint32 | O | 0..1 | Number value following "ptime" specified in IETF RFC 8866 [53]. This key is set only if media type is set "audio". An offer from UE that has different ptime value will be denied. An offer from network has a=ptime line based on this value. If this key is not set, a=ptime line in UE's offer will be not checked and derived a=ptime value will be used in network's offer. | |
| extmap | array(extmapElem) | O | 0..N | Information for RTP extension. | |

6.6.3.2.3.2.17 Type: extmapElem

**Table 6.6.3.2.3.2.17-1: Definition of type extmapElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| extensionName | string | M | 1 | String value following "extensionname" specified in IETF RFC 8285 [37]. If any RTP extension headers are allowed, only one extmapElem is set to "extmap" and extensionName is required to be set "#any". An offer from network has a=rtpmap line based on this value. An offer from UE that has a=rtpmap line that is not included in "extmap" will be denied. If this key is not set, an offer from network has no a=rtpmap line. | |

6.6.3.2.3.2.18 Type: setupControl

**Table 6.6.3.2.3.2.18-1: Definition of type setupControl**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| iniOffer | iniOffer | O | 0..1 | Information for negotiation control for initial offer. | |
| iniAnswer | iniAnswer | O | 0..1 | Information for negotiation control for initial answer. | |
| subOffer | subOffer | O | 0..1 | Information for negotiation control for subsequent offer. | |
| subAnswer | subAnswer | O | 0..1 | Information for negotiation control for subsequent answer. | |
| NOTE: If each attribute is not set, default rule is applied. | | | | | |

6.6.3.2.3.2.19 Type: iniOffer

**Table 6.6.3.2.3.2.19-1: Definition of type iniOffer**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| direction | string | O | 0..1 | If "network" is set, initial offer is always sent from network. If "ue" is set initial offer is always sent from UE. If "both" is set, initial offer can be sent from UE or network. If not set, "network" is applied. | |
| onMsetupReq | string | O | 0..1 | If "required" is set, initial offer is required to be set in msetup request. If "optional" is set, initial offer can be set in msetup request. If "none" is set, initial offer is not allowed to be set in msetup request. If not set, "optional" is applied. | |
| oaTimeout | uint32 | O | 0..1 | Limited time in seconds for receiving the initial answer in response to the initial offer until timeout. If zero is set, the time is unlimited. If a timeout occurs, the media session will be disconnected. If not set, 180 is applied. | |

6.6.3.2.3.2.20 Type: iniAnswer

**Table 6.6.3.2.3.2.20-1: Definition of type iniAnswer**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| onMsetupRes | string | O | 0..1 | If "required" is set, initial answer is required to be set in msetup response. If "optional" is set, initial answer can be set in msetup response. If "none" is set, initial answer is not allowed to be set in msetup response. If not set, "optional" is applied. | |
| oaTimeout | uint32 | O | 0..1 | Limited time in seconds for receiving the initial answer in response to the initial offer until timeout. If zero is set, the time is unlimited. If a timeout occurs, the media session will be disconnected. If not set, 180 is applied. | |

6.6.3.2.3.2.21 Type: subOffer

**Table 6.6.3.2.3.2.21-1: Definition of type subOffer**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| direction | string | O | 0..1 | If "network" is set, subsequent offer is only sent from network. If "ue" is set subsequent offer is only sent from UE. If "both" is set, subsequent offer can be sent from UE or network. If "none" is set, subsequent offer is not allowed. If not set, "network" is applied. | |
| oaTimeout | uint32 | O | 0..1 | Limited time in seconds for receiving the subsequent answer in response to the subsequent offer until timeout. If zero is set, the time is unlimited. If a timeout occurs, the media session will be disconnected. If not set, 180 is applied. | |

6.6.3.2.3.2.22 Type: subAnswer

**Table 6.6.3.2.3.2.22-1: Definition of type subAnswer**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| oaTimeout | uint32 | O | 0..1 | Limited time in seconds for receiving the subsequent answer in response to the subsequent offer until timeout. If zero is set, the time is unlimited. If a timeout occurs, the media session will be disconnected. If not set, 180 is applied. | |

6.6.3.2.3.2.23 Type: dcGroupLabelElem

**Table 6.6.3.2.3.2.23-1: Definition of type dcGroupLabelElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| dsGroupName | string | M | 1 | Name of data channel group. It is required to be unique in a specific RTC ID resource. | |
| numMax | uint32 | O | 0..1 | Maximum number of data channels that have the label indicated in ds. If not set, the number is unlimited. | |
| numMinPerUe | uint32 | O | 0..1 | Minimum number of data channels that have the label indicated in ds and belong to a specific UE. If not set, zero is applied. | |
| numMaxPerUe | uint32 | O | 0..1 | Maximum number of data channels that have the label indicated in ds and belong to a specific UE. If not set, the number is unlimited. | |
| subprotocol | string | O | 0..1 | String value following "subprotocol" parameter specified in IETF RFC 8864 [52]. | |
| ordered | boolean | O | 0..1 | String value following "ordered" parameter specified in IETF RFC 8864 [52]. If not set, "true" is applied. | |
| maxretr | uint32 | O | 0..1 | String value following "max-retr" parameter specified in IETF RFC 8864 [52]. If not set, reliable transmission is performed. | |
| maxtime | uint32 | O | 0..1 | String value following "max-time" parameter specified in IETF RFC 8864 [52]. If not set, reliable transmission is performed. | |
| priority | uint16 | O | 0..1 | String value following "priority" parameter specified in IETF RFC 8864 [52]. If not set, 256 is applied. | |

6.6.3.2.3.2.24 Type: routingRuleElem

**Table 6.6.3.2.3.2.24-1: Definition of type routingRuleElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| routingName | string | M | 1 | Name of the routing rule. | |
| routing | array(routingElem) | M | 2..N | Description of media/data routing pipeline. First element indicates input stage. Last element indicates output stage. Other middle element(s) indicate several media/data processes. | |

6.6.3.2.3.2.25 Type: routingElem

**Table 6.6.3.2.3.2.25-1: Definition of type routingElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| element | array(string) | M | 1..N | Elements of media/data pipeline. If the routingElem is first element, media/data group label, routingName of another routingRuleElem or processer name is set. If the routingElem is last element, media/data group label or routingName of another routingRuleElem is set. If the routingElem is not first or last one, only processor name can be set. | |
| NOTE: Processor names are listed in Table 6.6.3.2.3.2.25-2. | | | | | |

**Table 6.6.3.2.3.2.25-2: Definition of processor**

| Processor name | Description |
|---|---|
| std.mc.dup | Media process performs media duplication. Single input media (e.g., audio, video) is duplicated and sent to multiple output. Previous stage (media group or routingRule) is required to have only one output media. If the next stage is routingRule, only one media channel is output. If the next stage is group label, one media channel is output to each UE. |
| std.mc.cast | Media process performs media duplication in large scale (e.g., more than hundreds output). (NOTE 1) |
| std.mc.forward | Media process performs forwarding. Input media are duplicated and forwarded to outputs. Previous stage can be one or more group labels or routingRules with multiple media channels. If "loopback=true" parameter is set, input media is also sent to source UE. If parameter is not set, "loopback=false" is applied. |
| std.mc.audio.mix | Media process performs audio mixing. Multiple input audio are processed and output single mixed audio. |
| std.dc.dup | Data stream duplication. The rule for input and output follows "std.mc.dup". |
| std.dc.forward | Data stream forwarding. The rule for input and output follows "std.mc.forward". |
| std.dc.mix | Data stream mixing (e.g., data channel stream multiplexed with label). The rule for input and output follows "std.mc.audio.mix". |
| NOTE 1: In cases, large scale media duplication has different internal implementation from ordinary one (e.g., multi-stage duplication). Therefore, "std.mc.dup" and "std.md.cast" are listed as different processor.<br>NOTE 2: Processors are expected to have internal attribute (e.g., allowed input/output group label, expected number of media/data) | |

6.6.3.2.3.3 Simple data types and enumerations

6.6.3.2.3.3.1 Simple data types

The simple data types defined in table 6.6.3.2.3.3.1-1 shall be supported.

**Table 6.6.3.2.3.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

## 6.6.3.2.4 Error Handling

General error responses are defined in clause 6.6.3.1.5.

### 6.6.3.3 Notification of RTC ID resource deletion

#### 6.6.3.3.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's RTC ID resource enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

#### 6.6.3.3.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

#### 6.6.3.3.3 Notification operation

##### 6.6.3.3.3.1 Description

The Callback operation is used by the RTC ID resource enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that RTC ID resource deletion, requested by CP, has been completed.

##### 6.6.3.3.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.3.3.2-1.

**Table 6.6.3.3.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.3.3.2-2 and the response data structure and response codes described in table 6.6.3.3.3.2-3.

**Table 6.6.3.3.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of deleted RTC ID resource |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.3.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | Completion of RTC ID resource deletion is acknowledged. |

**Table 6.6.3.3.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

### 6.6.3.3.4 Data Model

#### 6.6.3.3.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.3.4.1-1 depicts the data types used for Notification of RTC ID resource deletion API. Those data types are shared among following callback operations. As described in the "Applicability", optional keys are not used in this API.

**Table 6.6.3.3.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Callback request body | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Callback response body | |
| mediaControlElem | Clause 6.6.3.3.4.2.3 | Information for media control | |

Table 6.6.3.3.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.3.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| mediaInfo | Clause 6.4.5.5.4.3.19 | Re-used in rtcCallbackReq and mediaControlElem object. | |

#### 6.6.3.3.4.2 Structured data types

##### 6.6.3.3.4.2.1 Type: rtcCallbackReq

**Table 6.6.3.3.4.2.1-1: Definition of type rtcCallbackReq**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| trigger | string | M | 1 | Callback trigger described in Table 6.6.3.2.3.2.4-2. | |
| rtcId | string | M | 1 | The RTC resource ID that triggered the event for the callback request. | |
| seqId | uint32 | M | 1 | The event sequence number initiated by the corresponding RTC resource ID, starting from zero. | |
| oId | string | O | 0..1 | Originating ID of a new media session setup attempt. RTC user ID or RTC resource ID is expected to be set. | "call.in.requested" "call.in.accepted" "call.in.connected" |
| mediaInfo | mediaInfo | O | 0..1 | MediaInfo object included in msetup request. | "call.in.requested" "call.in.accepted" "call.in.connected" |
| participantId | string | O | 0..1 | Participant ID assigned by WSF to UE. | "call.in.accepted" "call.in.connected" "call.disconnected" |
| mediaControl | array(mediaControlElem) | O | 0..N | Information for mediaControl. | "call.media.requested" |
| cleared | boolean | O | 0..1 | Set to true if UE connection status is cleared. | "resource.resumed" |

6.6.3.3.4.2.2 Type: rtcCallbackRes

**Table 6.6.3.3.4.2.2-1: Definition of type rtcCallbackRes**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| success | boolean | M | 1 | Set to true if corresponding request is successfully processed. | |
| ueRole | string | O | 0..1 | Indicate UE's role based on media routing properties described in clause 6.6.3.2.3.2.6. | "call.in.requested"<br>"call.in.accepted"<br>"call.in.connected" |
| mediaControl | array(mediaControlElem) | O | 0..N | Information for media control. | "call.in.accepted"<br>"call.in.connected"<br>"call.disconnected"<br>"call.media.requested" |

6.6.3.3.4.2.3 Type: mediaControlElem

**Table 6.6.3.3.4.2.3-1: Definition of type mediaControlElem**

| Attribute name | Data type | P | Cardinality | Description | Applicability |
|---|---|---|---|---|---|
| participantId | string | M | 1 | Participant ID assigned by WSF to UE. | |
| mediaInfo | mediaInfo | M | 1 | MediaInfo object included in msetup request of the UE. | |

6.6.3.3.4.3 Simple data types and enumerations

6.6.3.3.4.3.1 Simple data types

**Table 6.6.3.3.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

### 6.6.3.3.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

## 6.6.3.4 Notification of forced RTC ID resource deletion

### 6.6.3.4.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's RTC ID resource enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

### 6.6.3.4.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

### 6.6.3.4.3 Notification operation

#### 6.6.3.4.3.1 Description

The Callback operation is used by the RTC ID resource enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that RTC ID resource has been deleted independently of the CP's request.

#### 6.6.3.4.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.4.3.2-1.

**Table 6.6.3.4.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.4.3.2-2 and the response data structure and response codes described in table 6.6.3.4.3.2-3.

**Table 6.6.3.4.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of deleted RTC ID resource |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.4.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | RTC ID resource deletion is acknowledged. |

**Table 6.6.3.4.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

### 6.6.3.4.4 Data Model

#### 6.6.3.4.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.4.4.1-1 depicts the data types specifically used for Notification of forced RTC ID resource deletion API.

**Table 6.6.3.4.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.4.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.4.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. Optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. Optional keys are not used. | |

#### 6.6.3.4.4.2 Structured data types

None.

#### 6.6.3.4.4.3 Simple data types and enumerations

##### 6.6.3.4.4.3.1 Simple data types

**Table 6.6.3.4.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

### 6.6.3.4.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

## 6.6.3.5 Notification of RTC ID resource suspended

### 6.6.3.5.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's RTC ID resource enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

### 6.6.3.5.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

### 6.6.3.5.3 Notification operation

#### 6.6.3.5.3.1 Description

The Callback operation is used by the RTC ID resource enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that RTC ID resource has been temporarily unavailable.

#### 6.6.3.5.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.5.3.2-1.

**Table 6.6.3.5.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.5.3.2-2 and the response data structure and response codes described in table 6.6.3.5.3.2-3.

**Table 6.6.3.5.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of suspended RTC ID resource |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.5.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | The state transition of RTC ID resource to the suspended state is acknowledged. |

**Table 6.6.3.5.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

## 6.6.3.5.4 Data Model

### 6.6.3.5.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.5.4.1-1 depicts the data types specifically used for Notification of RTC ID resource suspended API.

**Table 6.6.3.5.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.5.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.5.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. Optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. Optional keys are not used. | |

### 6.6.3.5.4.2 Structured data types

None.

### 6.6.3.5.4.3 Simple data types and enumerations

#### 6.6.3.5.4.3.1 Simple data types

**Table 6.6.3.5.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

#### 6.6.3.5.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

### 6.6.3.6 Notification of RTC ID resource resumed

#### 6.6.3.6.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's RTC ID resource enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

#### 6.6.3.6.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

#### 6.6.3.6.3 Notification operation

##### 6.6.3.6.3.1 Description

The Callback operation is used by the RTC ID resource enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that RTC ID resource has been recovered from suspended state.

##### 6.6.3.6.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.6.3.2-1.

**Table 6.6.3.6.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.6.3.2-2 and the response data structure and response codes described in table 6.6.3.6.3.2-3.

**Table 6.6.3.6.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of resumed RTC ID resource |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.6.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | The state transition of RTC ID resource to the resumed is acknowledged. |

**Table 6.6.3.6.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

#### 6.6.3.6.4 Data Model

##### 6.6.3.6.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.6.4.1-1 depicts the data types specifically used for Notification of RTC ID resource resumed API.

**Table 6.6.3.6.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.6.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.6.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. "cleared" key can be set. Other optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. Optional keys are not used. | |

##### 6.6.3.6.4.2 Structured data types

None.

##### 6.6.3.6.4.3 Simple data types and enumerations

###### 6.6.3.6.4.3.1 Simple data types

**Table 6.6.3.6.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

#### 6.6.3.6.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

### 6.6.3.7 Notification of user call in requested

#### 6.6.3.7.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's connection control enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

#### 6.6.3.7.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

#### 6.6.3.7.3 Notification operation

##### 6.6.3.7.3.1 Description

The Callback operation is used by the connection control enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that a new attempt of media session setup has been requested by the UE.

##### 6.6.3.7.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.7.3.2-1.

**Table 6.6.3.7.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.7.3.2-2 and the response data structure and response codes described in table 6.6.3.7.3.2-3.

**Table 6.6.3.7.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of the new attempt of media session setup by UE. |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.7.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | The new attempt of media session setup by UE is acknowledged and CP provides the instruction including connection control and role assignment. |

**Table 6.6.3.7.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

#### 6.6.3.7.4 Data Model

##### 6.6.3.7.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.7.4.1-1 depicts the data types specifically used for Notification of user call in requested API.

**Table 6.6.3.7.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.7.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.7.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| mediaInfo | Clause 6.4.5.5.4.3.19 | Re-used in rtcCallbackReq. | |
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. Keys "oId" and "mediaInfo" are required to be set. Other optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. "ueRole" key can be set. Other optional keys are not used. | |

#### 6.6.3.7.4.2 Structured data types

None.

#### 6.6.3.7.4.3 Simple data types and enumerations

##### 6.6.3.7.4.3.1 Simple data types

**Table 6.6.3.7.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

### 6.6.3.7.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

## 6.6.3.8 Notification of user call in accepted

### 6.6.3.8.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's connection control enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

### 6.6.3.8.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

### 6.6.3.8.3 Notification operation

#### 6.6.3.8.3.1 Description

The Callback operation is used by the connection control enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that a new attempt of media session setup of UE has been accepted by the operator network.

#### 6.6.3.8.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.8.3.2-1.

**Table 6.6.3.8.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.8.3.2-2 and the response data structure and response codes described in table 6.6.3.8.3.2-3.

**Table 6.6.3.8.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of the accepted attempt of media session setup by UE |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.8.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | The accept of media session setup by UE is acknowledged and CP provides the instruction including connection control and role assignment. |

**Table 6.6.3.8.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

## 6.6.3.8.4 Data Model

### 6.6.3.8.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.8.4.1-1 depicts the data types specifically used for Notification of user call in accepted API.

**Table 6.6.3.8.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.8.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.8.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| mediaInfo | Clause 6.4.5.5.4.3.19 | Re-used in rtcCallbackReq. | |
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. Keys "oId", "mediaInfo" and "participantId" are required to be set. Other optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. "ueRole" and "mediaControl" key can be set. | |
| mediaControlElem | Clause 6.6.3.3.4.2.3 | Re-used in rtcCallbackRes. | |

#### 6.6.3.8.4.2 Structured data types

None.

#### 6.6.3.8.4.3 Simple data types and enumerations

##### 6.6.3.8.4.3.1 Simple data types

**Table 6.6.3.8.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

### 6.6.3.8.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

## 6.6.3.9 Notification of user call in connected

### 6.6.3.9.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's connection control enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

### 6.6.3.9.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

### 6.6.3.9.3 Notification operation

#### 6.6.3.9.3.1 Description

The Callback operation is used by the connection control enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that a new media session has been established and media have been connected.

#### 6.6.3.9.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.9.3.2-1.

**Table 6.6.3.9.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.9.3.2-2 and the response data structure and response codes described in table 6.6.3.9.3.2-3.

**Table 6.6.3.9.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of the established media session |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.9.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | The connected media session is acknowledged and CP provides the instruction including role assignment and media routing control. |

**Table 6.6.3.9.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

### 6.6.3.9.4 Data Model

#### 6.6.3.9.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.9.4.1-1 depicts the data types specifically used for Notification of user call in connected API.

**Table 6.6.3.9.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.9.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.9.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| mediaInfo | Clause 6.4.5.5.4.3.19 | Re-used in rtcCallbackReq. | |
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. Keys "oId", "mediaInfo" and "participantId" are required to be set. Other optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. "ueRole" and "mediaControl" key can be set. | |
| mediaControlElem | Clause 6.6.3.3.4.2.3 | Re-used in rtcCallbackRes. | |

#### 6.6.3.9.4.2 Structured data types

None.

#### 6.6.3.9.4.3 Simple data types and enumerations

##### 6.6.3.9.4.3.1 Simple data types

**Table 6.6.3.9.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

### 6.6.3.9.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

## 6.6.3.10 Notification of user call in connected

### 6.6.3.10.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's connection control enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

### 6.6.3.10.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

### 6.6.3.10.3 Notification operation

#### 6.6.3.10.3.1 Description

The Callback operation is used by the connection control enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that an established media session has been disconnected.

#### 6.6.3.10.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.10.3.2-1.

**Table 6.6.3.10.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.10.3.2-2 and the response data structure and response codes described in table 6.6.3.10.3.2-3.

**Table 6.6.3.10.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of the disconnected media session |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.10.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | The media session disconnection is acknowledged and CP provides the instruction for media routing control. |

**Table 6.6.3.10.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

### 6.6.3.10.4 Data Model

#### 6.6.3.10.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.10.4.1-1 depicts the data types specifically used for Notification of user call disconnected API.

**Table 6.6.3.10.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.10.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.10.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| mediaInfo | Clause 6.4.5.5.4.3.19 | Re-used in mediaControlElem. | |
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq. "participantId" is required to be set. Other optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes. "mediaControl" key can be set. | |
| mediaControlElem | Clause 6.6.3.3.4.2.3 | Re-used in rtcCallbackRes. | |

#### 6.6.3.10.4.2 Structured data types

None.

#### 6.6.3.10.4.3 Simple data types and enumerations

##### 6.6.3.10.4.3.1 Simple data types

**Table 6.6.3.10.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

### 6.6.3.10.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

### 6.6.3.11 Notification of media routing query

#### 6.6.3.11.1 API URI

The operation is a notification operation for subscribed events, and HTTP requests from the OP's MDFC enforcer are made to the Callback URI set in the RTC ID resource management operations in clause 6.6.3.2.

#### 6.6.3.11.2 Resources

The operation is a notification operation for subscribed events, and CP does not have structured resources.

#### 6.6.3.11.3 Notification operation

##### 6.6.3.11.3.1 Description

The Callback operation is used by the MDFC enforcer to notify the CP, which has subscribed to events related to registered RTC ID resources, about the occurrence of events. This operation is used to notify events that media are added or removed in the existing media session.

##### 6.6.3.11.3.2 Notification operation definition

The POST method is required to be used for event notification and the URI is to be the one set in the RTC ID resource registration.

Callback URI: {**callbackURI**}

This method is required to support the URI query parameters described in table 6.6.3.11.3.2-1.

**Table 6.6.3.11.3.2-1: URI query parameters supported by the POST method on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

This method is required to support the request data structures described in table 6.6.3.11.3.2-2 and the response data structure and response codes described in table 6.6.3.11.3.2-3.

**Table 6.6.3.11.3.2-2: Data structures supported by the POST Request Body on this resource**

| Data type | P | Cardinality | Description |
|---|---|---|---|
| rtcCallbackReq | M | 1..N | Callback request including the information of added or removed media |
| NOTE: Multiple callback operations can be sent simultaneously and bundled in a request. | | | |

**Table 6.6.3.11.3.2-3: Data structures supported by the POST Response Body on this resource**

| Data type | P | Cardinality | Response codes | Description |
|---|---|---|---|---|
| rtcCallbackRes | M | 1 | 200 OK | Addition or removal of media is acknowledged and CP provides the instruction of media routing control. |

**Table 6.6.3.11.3.2-4: Headers supported by the 200 Response Code on this resource**

| Name | Data type | P | Cardinality | Description |
|---|---|---|---|---|
| n/a | | | | |

#### 6.6.3.11.4 Data Model

##### 6.6.3.11.4.1 General

This clause describes the application data model supported by the API.

Table 6.6.3.11.4.1-1 depicts the data types specifically used for Notification of media routing query API.

**Table 6.6.3.11.4.1-1: API specific Data Types**

| Data type | Section defined | Description | Applicability |
|---|---|---|---|
| n/a | | | |

Table 6.6.3.11.4.1-2 describes data types re-used by the API service.

**Table 6.6.3.11.4.1-2: Re-used Data Types**

| Data type | Reference | Comments | Applicability |
|---|---|---|---|
| mediaInfo | Clause 6.4.5.5.4.3.19 | Re-used in mediaControlElem. | |
| rtcCallbackReq | Clause 6.6.3.3.4.2.1 | Re-used for rtcCallbackReq.<br>"mediaControl" key is required to be set.<br>Other optional keys are not used. | |
| rtcCallbackRes | Clause 6.6.3.3.4.2.2 | Re-used for rtcCallbackRes.<br>"mediaControl" key is required to be set. | |
| mediaControlElem | Clause 6.6.3.3.4.2.3 | Re-used in rtcCallback Req and rtcCallbackRes. | |

#### 6.6.3.11.4.2 Structured data types

None.

#### 6.6.3.11.4.3 Simple data types and enumerations

##### 6.6.3.11.4.3.1 Simple data types

**Table 6.6.3.11.4.3.1-1: Simple data types**

| Type Name | Type Definition | Description | Applicability |
|---|---|---|---|
| n/a | | | |

#### 6.6.3.11.5 Error Handling

General error responses are defined in clause 6.6.3.1.5.

### 6.6.4 Solution evaluation

The proposed service control procedures and APIs fulfil the purpose that CP's service logic managers instruct operator's service logic enforcers to achieve the co-operated services (with flexible resource management and elaborate media routing control) described in corresponding key issue. Then, it is proposed to implement the procedures and APIs in the stage 3 specification of RTC on the basis of the proposals above.

# 6.7 Solution #6: WSF Discovery mechanism

## 6.7.1 Solution description

### 6.7.1.1 General

This solution addresses Key Issue #6.

This clause identifies the mechanism which discovers a WSF in the connected operator network without user manual setting, regardless of the connected operator network.

There are following possible mechanisms to find a WSF without user settings.

a) Media Session Handler (via RTC-5 API) (3GPP TS 26.506 [12])

b) Edge application enabler (EAS discovery) (3GPP TS 23.558 [7])

c) PCO in NAS signalling during PDU session set up (3GPP TS 23.501 [4], 3GPP TS 23.548 [6])

d) DNS resolution

### 6.7.1.2 Analysis on possible mechanisms

As described in 3GPP TS 26.506 [12] and the clause 6.2 of this document, RTC services need to support both native WebRTC application (i.e., WebRTC non-browser type endpoint) and web application (i.e., WebRTC browser type endpoint).

a) and b) are the mechanisms using application enabler specified in 3GPP. However, in the current situation, most of the Oss (e.g., android, iOS) and the web browsers (e.g., chrome, firefox) do not support these enablers for JavaScript Application as JavaScript API. Then, a) and b) are not suitable for the time being.

c) is the mechanism to get a server information from Protocol Configuration Option (PCO) during PDU session establishment, however, there are same issue as a) and b) to apply this mechanism. Then, c) is also not suitable for the time being.

d) intends to use the local DNS server in the connected operator network to resolve the single specific FQDN into the actual IP address of the server in the connected operator network. This mechanism does not have the limitation mentioned above, then d) is the possible candidate of the WSF discovery mechanism.

Therefore, this solution studies a mechanism which apply d) using a specific URL (which is common among operators) to discovers a WSF in the connected operator network without user manual setting, regardless of the connected operator network.

## 6.7.2 Common URL based WSF discovery mechanism

### 6.7.2.1 General

This solution studies the mechanism which discovers the WSF in the connected operator network using a specific URL which is common among operators (names the URL as "common URL" in this document) and local DNS server in the connected operator network.

As the prerequisites of the study on WSF discovery mechanism, the following requirements specified in 3GPP TS 26.113 [10] need to be considered.

1) The mechanism can identify the signalling protocol used for the RTC session set up, since there are multiple signalling protocols for RTC services (i.e., SWAP, RESPECT).

2) Secure WebSocket (WSS) connection is applicable between the WebRTC endpoint and the WSF.

Above requirements are not fulfilled, if the common URL indicates only the WSF URL (e.g., the public TLS certificate cannot be prepared.). Then, the URLs for the WSF discovery mechanism are specified as follows.

Common URL:
A specific URL which is common among RTC operator networks and is used to get the WSF URL(s) from the WSF discovery function in the connected operator network. This URI indicates the signalling protocol in addition to WSF URI, which is derived from the above requirement 1).

WSF URL:
Secure WebSocket URI of WSF which is specified in 3GPP TS 26.113 [10]. The hostname of the WSF URL is specific hostname for RTC service and assigned by the operator.

Considering the above, this Solution studies the followings.

1) Common URL format

2) Common URL based WSF discovery procedure

### 6.7.2.2 Common URL format

This clause studies the format of common URL.

Common URL needs to indicate the signalling protocol which expected to be used by application, since multiple signalling protocols (i.e., SWAP and RESPECT) are applicable for RTC session set up as described in clause 6.7.2.1. Then, the common URL need to include "protocolName" which specified in 3GPP TS 26.113 [10]. Therefore, the following format is proposed as common URL.

**CommonURL:** {commonHostname}/<protocolName>

NOTE 1: WebSocket URI includes "protocol version". However, "protocol version" is not included in common URL, since the compatibility between versions and its version management depend on the signalling protocol.

For "commonHostname", it seems appropriate that RTC applies the domain name ".3gppservices.org" as 3gpp service, which is defined in 3GPP 5GMS (3GPP TS 26.512 [14]) as default AF's hostname. Then, "commonHostname" in common URL is proposed as following:

{commonHostname}: "rtc.3gppservices.org"

NOTE 2: As an alternative domain name for (commonHostname), there are IETF RFC 6762 based domain name (e.g., .internal). However, the IETF based solution is not studied since the 3GPP based approach is appropriate for RTC service.

In case of RESPECT protocol studied in this document, the Common URL will be following URL.

- rtc.3gppservices.org/3gpp-respect

### 6.7.2.3 Common URL based WSF discovery procedure

#### 6.7.2.3.1 General

This clause studies the procedure at the WebRTC endpoint to discover the WSF in the connected operator network by using common URL.

As described in clause 6.7.2.1, common URL is used to get the WSF URL(s) from WSF discovery function. Then, the UE procedure for discovering and connecting to the WSF is as follows.

i) Get WSF URL(s) from a WSF discovery function by using common URL

ii) Connect to a selected WSF from the obtained WSF URL(s) by WSS (secure WebSocket)

NOTE 1: The operator who provide this WSF discovery mechanism needs to provide DNS server to resolve the commonHostname (i.e., rtc.3gppservices.org) of common URL into the IP address of the WSF discovery function.

NOTE 2: The method to decide the connecting WSF from the obtained WSF list depends on the application. This solution addresses the procedure for step i). Step ii) is studied in Key Issue #3 and corresponding solution in this document.

#### 6.7.2.3.2 Protocol

There are two possible protocols for getting WSF URL(s) using common URL.

a) HTTP

b) WebSocket

For the following reasons, WebSocket connection is too much for getting WSF URL. Then this solution studies HTTP-based procedure for getting the WSF URL.

- Push notification from discovery function is not required.
- WSF discovery procedure is expected to be triggered when the RTC application is activated or in case of WSF connection error, then the frequency of execution is low.

NOTE: Cross-Origin Resource Sharing (CORS) needs to be considered for WSF discovery since the domain of Common URL and WSF URL are different, as described in the prerequisites in clause 6.7.2.1.

#### 6.7.2.3.3 Procedure

The following procedure is proposed as HTTP based WSF discovery procedure.

i) The RTC application sends a HTTP GET request to the Common URL (the request is sent to WSF discovery function)

ii) The WSF discovery function sends back an HTTP response as follows, depending on whether the indicated signalling protocol is supported or not in the connected operator network.

a) 200 (OK)
When the connected operator network supports the indicated signalling protocol, the WSF discovery function sends back an HTTP 200 (OK) response. The response body (Content-Type: application/json) includes WSF URLs (WebSocket URI specified in 3GPP TS 26.113 [10]) for all protocol versions which the operator network supports. The response body format is protocol-independent. Example of the response body for RESPECT is shown below:

<Response body for RESPECT>

```
{
  "v1": {
    "wsfUrl": ["wss://wsf-1.example.com/3gpp-respect/v1", "wss://wsf-2.example.com/3gpp-respect/v1"]
  }
}
```

b) 404 (Not Found):
When the connected operator network does not support the indicated singalling protocol, the WSF discovery function sends an HTTP 404 (Not Found) response.

#### 6.7.2.3.4 Definition of the HTTP response body for RESPECT

The definition of the HTTP response body for WSF discovery procedure using Common URL for RESPECT.

**Table 6.7.2.3.4-1: Information Element in the response body for RESPECT**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| v1 | v1Info | 1 | This information element is for WSF information for RESPET version 1. |

**Table 6.7.2.3.4-2: Data type definition of v1Info**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| wsfUrl | array(string) | 1 | This information element indicates the WSF URL(s). The format of the WSF URL is required to be the WebSocket URI specified in 3GPP TS 26.113 [10].<br>e.g., wss://wsf.example.com/3gpp-respect/v1 |

#### 6.7.2.3.5 Common URL based WSF discovery flow example

Following message flow is an example of WSF discovery procedure using Common URL for RESPECT.

1) HTTP GET Request (RTC application -> WSF discovery function)

   GET /3gpp-respect HTTP/1.1

   Host: rtc.3gppservices.org

   -- other HTTP headers are snipped --

2) HTTP 200 OK response (WSF discovery function -> RTC application)

   HTTP/1.1 200 OK

   Content-Type: application/json

-- other HTTP headers are snipped --

```
{
 "v1": {
   "wsfUrl": ["wss://wsf.example.com/3gpp-respect/v1"]
 }
}
```

## 6.7.3 Functional entity supporting WSF discovery function

This clause studies the functional deployment of WSF discovery function in generic RTC architecture specified in 3GPP TS 26.506 [12].

As described in the above clauses, WSF discovery function provides the WSF URL(s) in the connected operator network to RTC applications. The WSF discovery function is required to provide WSF URL(s) to RTC applications even if MSH (RTC-5) is not applicable. Then, it seems better to implement the function in RTC AS. In RTC AS functionalities, Application Supporting Web Function (ASWF) is appropriate function to implement WSF discovery function, since ASWF is the function to support RTC applications as a web server.

## 6.7.4 Solution evaluation

The proposed WSF discovery mechanism using common URL described in clause 6.7.2 fulfils the purpose of the mechanism (i.e., the WSF discovery mechanism without user manual setting and applicable regardless of the connected

operator network) described in the corresponding key issue and consistent with existing 3GPP specification. Then, it is proposed to implement the proposed WSF discovery mechanism as the functionality supported at the ASWF into the stage 2 specification of RTC (i.e., 3GPP TS 26.506 [12]) as an optional mechanism.

# 6.8 Solution #7: Interworking with IMS network

## 6.8.1 Solution description

This solution addresses Key Issue #7.

This solution identifies the followings to support interworking between RTC network and IMS network:

1) applicable interface between RTC network to IMS network,
2) supported interworking scenarios between RTC network and IMS network,
3) functional requirements for RTC-IMS interworking; and
4) RTC architecture enhancements for RTC-IMS interworking.

As a prerequisite, this solution is required to have no impact on existing IMS technical specifications and implementations.

## 6.8.2 Interface between RTC network and IMS network

### 6.8.2.1 General

This clause identifies the applicable interface for interworking between RTC network and IMS network.

A solution for WebRTC-based service has been specified in Annex U of 3GPP TS 23.228 [3], where the WebRTC endpoint can access to IMS network via user-network interface (UNI) by introducing eP-CSCF and eIMS-AGW as in Figure 6.8.2.1.

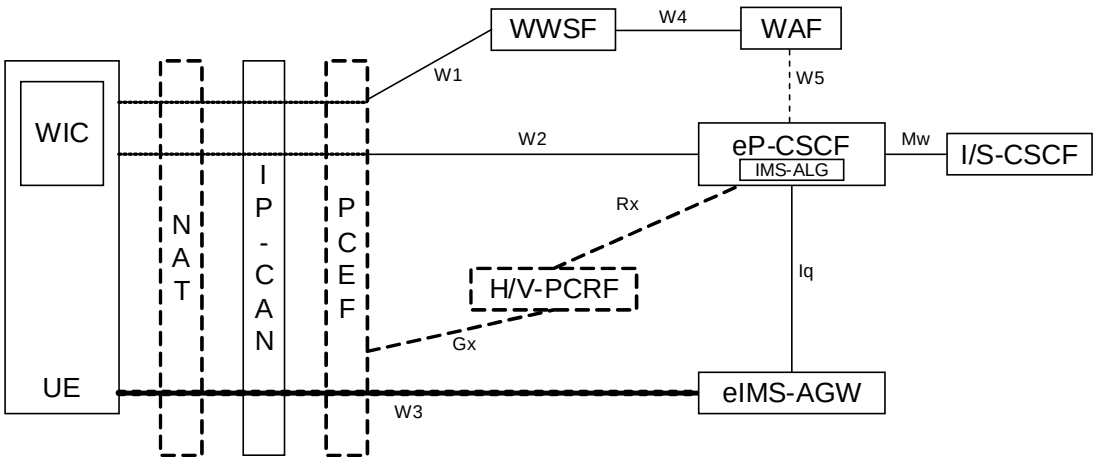


**Figure 6.8.2.1: WebRTC IMS architecture and reference model in 3GPP TS 23.228**

However, there is a possible demand that RTC network inter-connect to IMS network as another external IP multimedia network (so-called non-IMS).

Then, this solution considers RTC-IMS inter-connection using the network-to-network interface (NNI) between RTC network and IMS network.

Based on the prerequisite that this solution is required to have no impact on existing IMS technical specifications and implementations, this solution assumes the following conditions:

- RTC user (RTC endpoint) and IMS user (IMS UE) has its own MSISDN.

- In media session setup from RTC to IMS network, RTC endpoint (WebRTC endpoint) initiates media session by sending a media session setup request conforms to RTC signalling protocol (RESPECT) in this document, and both WSF and IWF forward the request towards IMS network based on the MSISDN of terminating IMS UE (tel URI) contained in the request.
- In media session setup from IMS to RTC network, IMS UE initiates media session by sending a media session setup request conforms to 3GPP TS 24.229 [8], and IMS functional entities (e.g., S-CSCF) forward the request towards RTC network based on MSISDN of terminating RTC endpoint available in the request – this is an existing functionality of IMS network.

Table 6.8.2.1-1 shows the identifier of terminating endpoint for each scenario.

**Table 6.8.2.1-1: Identifier of terminating endpoint for each scenario**

| Originating endpoint | Terminating endpoint | |
|---|---|---|
| | RTC endpoint | IMS UE |
| RTC endpoint | RTC user identity | MSISDN |
| IMS UE | MSISDN | Public user identity |

NOTE: The usage of identifier other than MSISDN for RTC-IMS interworking scenario is FFS.

### 6.8.2.2 Applicable interface between RTC network and IMS network

3GPP TS 29.162 [15] already defines the interface between IMS network and external IP multimedia network as shown in Figure 6.8.2.2.1. This interface is appropriate for RTC-IMS interworking scenario, since RTC network is considered as an external IP multimedia network. Therefore, this solution applies this interface for interworking between RTC network and IMS network.

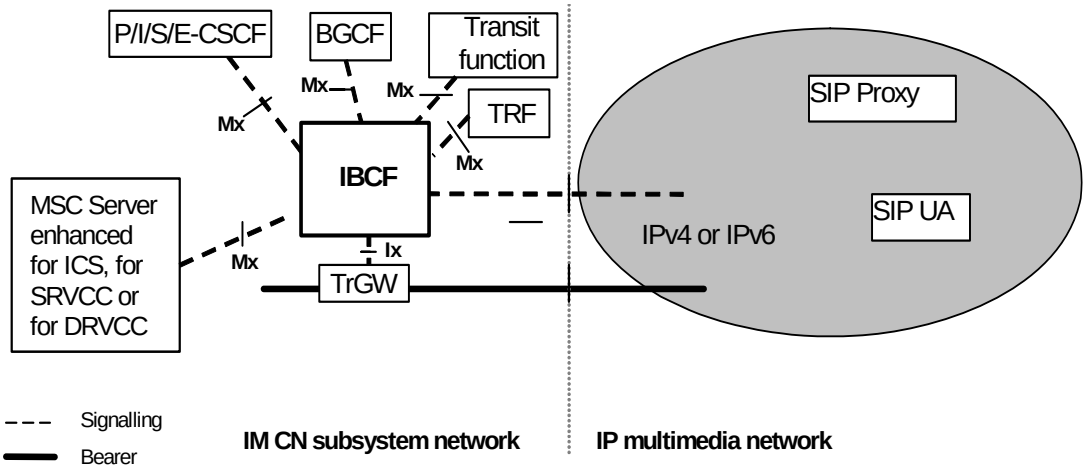


**Figure 6.8.2.2-1: Interworking model between IMS network and external IP Multimedia Network**

## 6.8.3 Interworking scenarios

### 6.8.3.1 General

This clause identifies the supported interworking scenarios that consists of connection patterns and media session between RTC network and IMS network.

### 6.8.3.2 Supported connection patterns

This clause identifies the supported connection patterns between RTC network and IMS network.

It is considered that there are following connection patterns.

1) Basic call between RTC endpoint and IMS UE

a) RTC endpoint initiates the media session setup to IMS UE

b) IMS UE initiates the media session setup to RTC endpoint

2) Conference call owned by media server

a) RTC endpoint connects to a conference room provided by IMS network

b) IMS UE connects to a conference room provided by RTC network

Since there is no different signalling requirement over the interface addressed in clause 6.8.2.2 between the connection pattern 1-a) and 2-a), this solution addresses 1-a), 1-b) and 2-b).

#### 6.8.3.3 Supported media session

This clause identifies the supported media session for interworking between RTC network and IMS network.

Media session provided by RTC network and/or IMS network could be categorized into the following two types:

- Basic & legacy audio call/conference
- Immersive media call/conference

As for basic audio call/conference, the interworking functionality needs to consider the several differences (e.g., signalling protocols, media capability, media transport protocols) between RTC media session and IMS media session.

As for immersive media call/conference, the interworking functionality does not need to consider the difference of media capability between RTC media session and IMS media session, compared to basic audio call/conference. This is because the two endpoints of the immersive media session are considered to have same media capabilities. That is, it is expected that immersive media call can be interconnected by using interworking specification for basic call.

Therefore, this solution focuses on the interworking of basic & legacy audio call/conference, which will cover the functional requirements for interworking of immersive media call/conference.

### 6.8.4 Functional requirements for RTC-IMS interworking

#### 6.8.4.1 General

This clause identifies the functional requirements for RTC-IMS interworking, based on the interface, interworking scenarios in clause 6.8.2 and 6.8.3. As described in clause 6.8.1, this solution is required to have no impact on existing IMS technical specifications and implementations as a basic requirement.

#### 6.8.4.2 Functional requirements for RTC network

This clause describes the functional requirements for RTC network.

1. RTC network is required to interwork the signalling message between RTC signalling protocol (RESPECT) in this document and SIP based IMS signalling protocol.

   a) The RTC signalling protocol message initiated by RTC endpoint is required to be interworked to SIP based IMS signalling message and forwarded to IMS network at the boundary of RTC network.

   b) The SIP based IMS signalling massage received from IMS network is required to be interworked to RTC signalling protocol message and forwarded to RTC endpoint at the boundary of the RTC network.

   c) The difference between RTC signalling protocol and SIP based IMS signalling protocol is required to be terminated at the boundary of RTC network.

2. RTC network is required to interwork the media session between RTC and IMS networks.

   a) Media transport protocol is required to be interworked between RTC media session and IMS media session at the boundary of RTC network. For example, DTLS/SRTP needs to be terminated and interworked to RTP, RTP and RTCP multiplexing needs to be terminated if not supported by connected IMS network.

b) When an SFU is applied for the media session on the RTC side, multiple media stream is required to be composed to a single media stream for IMS media session at the boundary of RTC network.

c) The differences of supported RTP header extension between RTC media session and IMS media session are required to be terminated at the boundary of RTC network.

3. RTC network is required to be possible to identify the call destined for IMS network or RTC endpoint by MSISDN available in signalling message and forward the call based on MSISDN.

#### 6.8.4.3 Functional requirements for IMS network

There are no functional requirements for IMS network.

### 6.8.5 RTC architecture enhancement for RTC-IMS interworking

This clause describes the enhancement on the RTC architecture, considering the functional requirements descried in clause 6.8.4 of this document.

As described in clause 4.2 of 3GPP TS 26.506 [12], IWF (Interworking Function) and TGF (Transport Gateway Function) are defined as the functions supporting border control functionality to intwer-connect with different network.

The reference point between RTC network and IMS network is not defined in the current 3GPP TS 26.506 [12]. Therefore, it is proposed that the interface defined in 3GPP TS 29.162 [15] is applied for interworking between RTC network and IMS network.

Figure 6.8.5-1 shows the logical connection architecture for inter-connection between RTC network and IMS network.

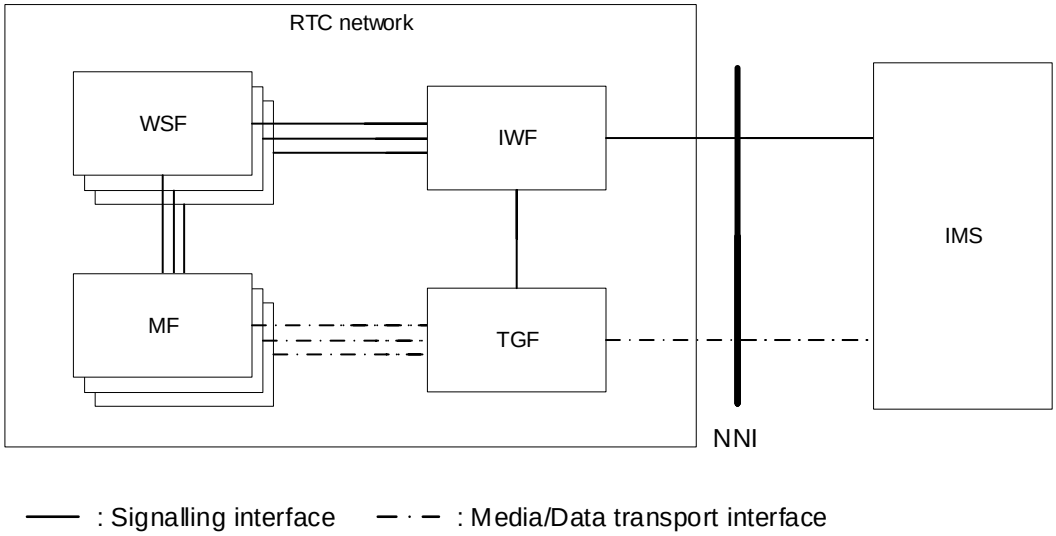


**Figure 6.8.5-1: Logical connection architecture for RTC-IMS inter-connection**

### 6.8.6 Solution evaluation

This solution proposes the enhancement on existing RTC generic architecture to support RTC-IMS interwork. The proposed architcture fulfills the requriements described in clause 6.8.4 with no impact on IMS specifications and implementations. Therefore, it is proposed that the interface defined in 3GPP TS 29.162 [15] is applied for interworking between RTC network and IMS network and reflected into stage2 specification of RTC (i.e., 3GPP TS 26.506 [12]).

Protocol-level interworking between RTC network and IMS network based on the functional requirements and architecture enhancements proposed in this solution is addressed in Key Issue #8 and Solution #8.

# 6.9 Solution #8: Protocol-level interworking between RTC network and IMS network

## 6.9.1 Solution description

This solution addresses Key Issue #8.

This solution identifies the protocol-level interworking between RTC network and IMS network for basic & legacy audio call/conference, based on the functional requirements and architecture described in Solution #4.

- C-Plane signalling interworking
- U-Plane media related interworking

This solution premises the followings:

1) RESPECT endpoint (UE) and IMS UE has its own MSISDN. Both RTC network and IMS network can identify the destination network based on MSISDN.
2) RTC network and IMS network are within the trust domain.
3) For C-plane signalling interworking, the IWF:
   - supports the interworking between RESPECT described in this document and IMS SIP/SDP defined in 3GPP TS 29.165 [16], as referred in 3GPP TS 29.162 [15],
   - terminates RTC or IMS specific capabilities/features and does not interwork to IMS or RTC network,
   - is able to manipulate SDP in order to compose multiple media streams from RTC network to a single media stream for IMS media session if an SFU is applied for the media session on the RTC network side and
   - does not interwork C-Plane signalling message for session keep-alive.

   NOTE 1: Interface specifications on the non-roaming II-NNI between home IMS networks defined in 3GPP TS 29.165 [16] are applied.
4) For U-plane media related interworking, the TGF:
   - interworks audio media between RTC network and IMS network,
   - adopts 3GPP TS 26.114 [11] as U-Plane media specification for the interface between RTC network and IMS network, as specified in TS 29.165 [16] and
   - is able to compose multiple media streams from RTC network to a single media stream for IMS media session based on the instruction from IWF.

   NOTE 2: Interworking of immersive media is FFS in this solution, as described in Solution #4.
5) The IMS network does not support optional capabilities/features. This aims to interwork with any IMS network.
6) The security aspects on the interface between RTC network and IMS network are compliant with 3GPP TS 29.162 [15].

## 6.9.2 C-Plane signalling interworking

### 6.9.2.1 General

This clause describes the followings as the RESPECT-IMS SIP/SDP interworking specification at the IWF:

- Protocol stack
- Interworking between RESPECT and IMS SIP/SDP signalling messages

### 6.9.2.2 Protocol stack

Figure 6.9.2.2-1 shows the protocol stack for interworking between RESPECT and IMS SIP/SDP.

The IWF performs the signalling protocol interworking (RESPECT - IMS SIP/SDP) and the underlying protocol interworking (Secure WebSocket - UDP/TCP) at the boundary between RTC and IMS networks.

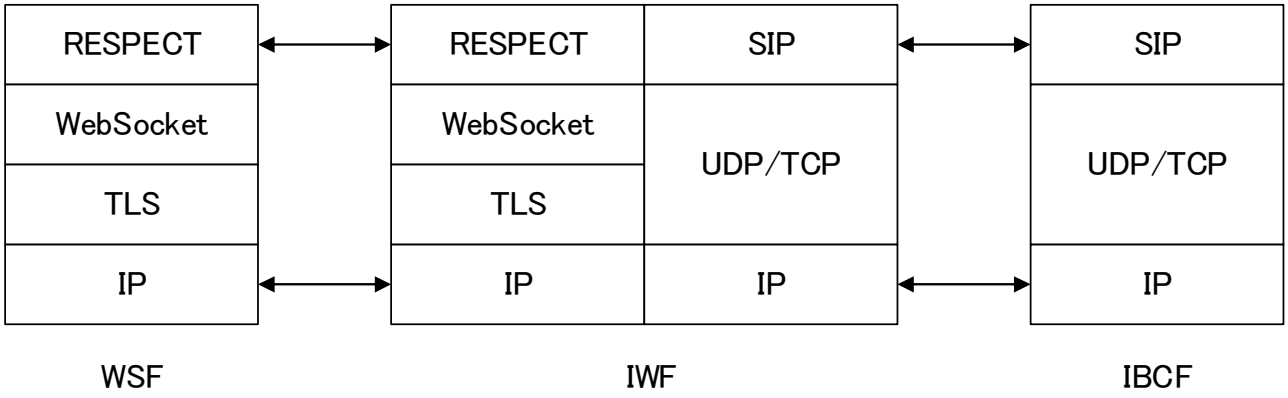


**Figure 6.9.2.2-1: Protocol interworking between RESPECT and IMS SIP**

### 6.9.2.3 Interworking procedures at the IWF

#### 6.9.2.3.1 General

This clause describes the interworking procedures at the IWF.

For the interface between RTC network and IMS network, the non-roaming II-NNI between home IMS networks defined in 3GPP TS 29.165 [16] are applied. Therefore, SIP REGISTER request/response will never be sent over the interface.

For the purposes of this document, the interworking procedures described here basically focus on the information element shared between RTC and IMS network (e.g., identifier of destination, originating user ID). Also, the procedures for abnormal cases are not considered.

The call flows and messages example are described in annex C of this document to grasp the whole picture of the interworking.

#### 6.9.2.3.2 Media session setup from RTC to IMS

##### 6.9.2.3.2.1 General

This clause describes the interworking procedures at the IWF for media session setup initiated from RTC network. The procedures in this clause are needed for media session establishment.

In this case, the IWF needs to manage both the media session state and participant state and inform the WSF of these status for the media session according to RESPECT protocol as a terminating RESPECT endpoint (AS), since the succeeding entity (i.e., IBCF) does not aware these RESPECT specific state.

##### 6.9.2.3.2.2 Receiving msetup request containing a preOffer

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.2.2-1.

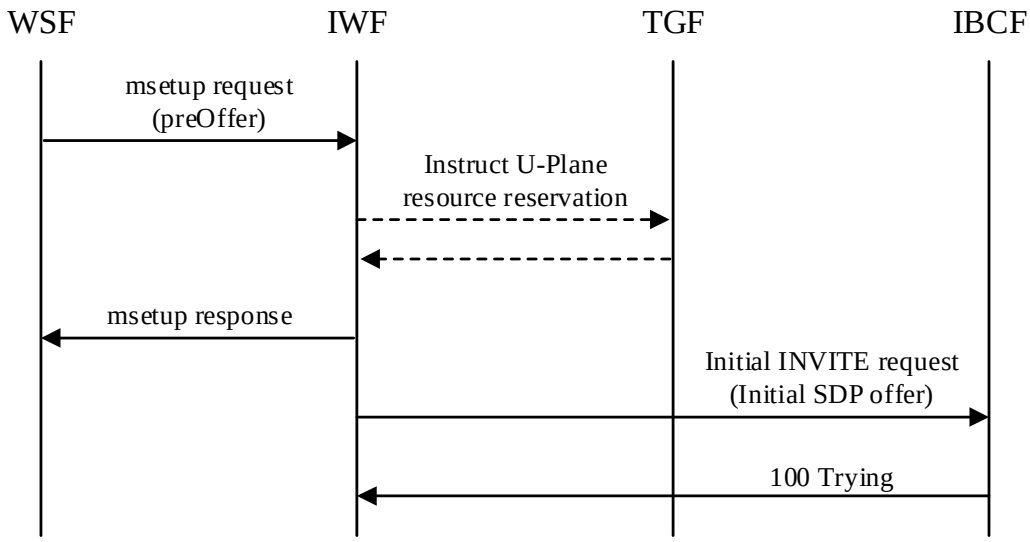


**Figure 6.9.2.3.2.2-1: Receiving msetup request containing a preOffer**

When receiving an msetup request containing "preOffer" type of the "mediaInfo" key from the WSF, then the IWF handles the "preOffer" as an initial SDP offer and then retrieves the IP address, port number for SIP, SIP domain name of the IBCF corresponding to an MSISDN set in "tn" field of the "dId" key, based on the pre-configured information in the IWF or ENUM transactions.

Upon successful retrieval, the IWF needs to:

1) store the received "preOffer" and construct an initial SDP offer being sent to the IBCF based on the local policy;
2) instruct the IGF to reserve U-Plane resource for the media session based on the received "preOffer" and the constructed initial SDP offer;
3) send an msetup response to the WSF containing:
   a) the "mediaInfo" key containing:
      - "type" field set to "info";
      - the "participantDesc" sub-key containing;
        * the object in which "actType" filed set to "add", "participantId" field set to anonymized RTC user ID of an originating RESPECT endpoint locally generated at the IWF and "userState" field set to "joiningIn";
   b) the "mediaSessionState" key set to "accepted" value;
   c) the other keys required for the msetup response as RESPECT endpoint;
4) send an SIP initial INVITE request to the IBCF containing:
   a) Request-URI set to the SIP URI containing global number digits in "tn" field of the "dId" key in the received msetup request;
   b) To header field containing the same SIP URI with Request-URI;
   c) From header field set to the SIP URI containing:
      - if "tn" field is present in the "user" sub-key of the "oId" key of the received msetup request, then the global number digits in this "tn" field of "user" sub-key;
      - if "tn" field is not present in the "user" sub-key of the "oId" key but in the "network" sub-key of the "oId" key, then the global number digits in this "tn" field of "network" sub-key;
      - otherwise, the anonymous URI;

d) P-Asserted-Identity header field set to the tel URI containing:

- the global number digits in "tn" field of the "network" sub-key of the "oId" key, if present in the received msetup request;
- otherwise, the unavailable URI;

e) Privacy header field set to:

- "id", if both "tn" field of the "network" sub-key and the "privacy" sub-key set to "id" of the "oId" key are present in the received msetup request;
- "none", if "tn" field of the "network" sub-key of the "oId" key is present but no "privacy" subkey is present in the received msetup request;

f) Identity header field constructed from the received "identity", "info", "alg" and "ppt" fields, if "passport" sub-key of the "oId" key is present;

g) Attestation-Info header field set to "A" according to RESPECT protocol;

h) Origination-Id header field set to the UUID locally generated at the IWF;

i) the other SIP header field required for the SIP initial INVITE request, which is locally generated as a SIP UA regardless of the received msetup request; and

j) SDP set to the initial SDP offer constructed at step 1).

When receiving SIP 100 (Trying) response to the SIP initial INVITE request, the IWF recognizes the SIP initial INVITE was successfully received at the IBCF.

#### 6.9.2.3.2.3 Receiving SIP 18x response containing an initial SDP answer

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.2.3-1.

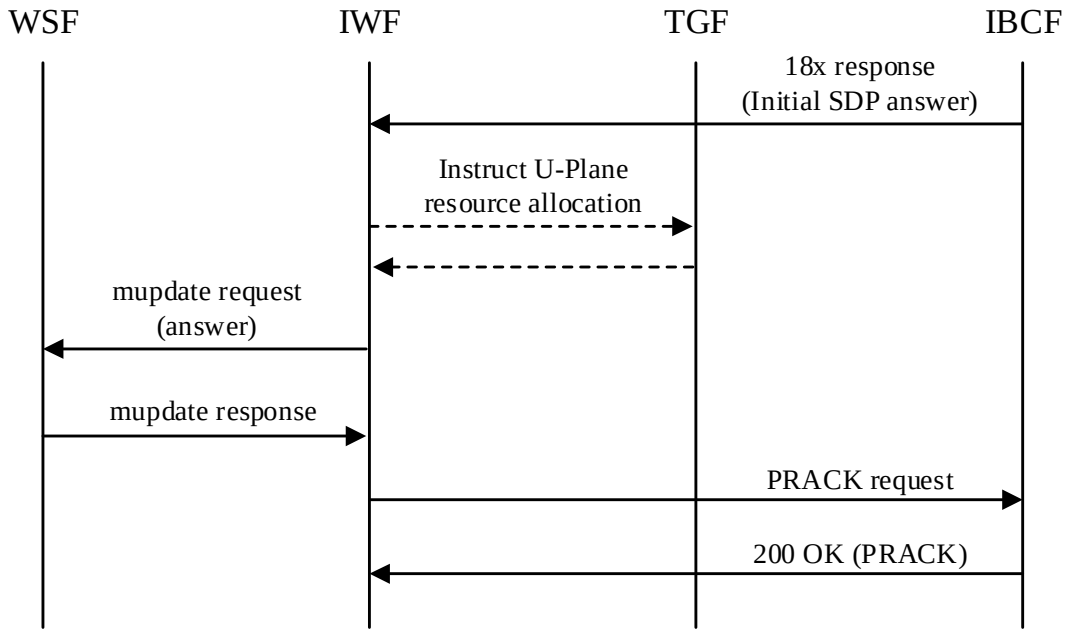


**Figure 6.9.2.3.2.3-1: Receiving SIP 18x response containing an initial SDP answer**

When receiving a SIP 180 (Ringing) response or a SIP 183 (Session Progress) response containing an initial SDP answer from the IBCF, then the IWF needs to:

1) construct "sdp" sub-key as an initial SDP answer being sent to the WSF based on the stored "preOffer";
2) instruct the IGF to allocate U-Plane resource for the media session based on the received initial SDP answer and the constructed initial SDP answer;

3) send an mupdate request to the WSF containing:

   a) the "mediaInfo" key containing:

      - "type" field set to "answer";
      - the "sdp" sub-key constructed at step 1);
      - if media description(s) of audio and/or video are contained in the constructed "sdp" sub-key, the object(s) per media description(s) in the "metadata" sub-key of the "mc" sub-key containing:
        * "index number" field set to the number corresponding to "preOffer";
        * "actType" field set to either "aly" or "dcl";
        * if "actType" field is set to "aly", the "state" sub-key including both "connected" and "routed" field with appropriate value depending on the answer from IGF at step 2);
      - if a media description of data channel is contained in the constructed "sdp" sub-key, the "dc" sub-key including "sdp index" field set to the number corresponding to "preOffer" and the object(s) per SCTP stream in the "metadata" sub-key containing:
        * "id" field set to the SCTP stream number managed in the TGF;
        * "actType" field set to either "aly" or "dcl";
        * if "actType" field is set to "aly", the "state" sub-key including both "connected" and "routed" field with appropriate value depending on the answer from IGF at step 2);
      - the "participantDesc" sub-key containing:
        * the object in which "actType" filed set to "add", "participantId" field set to anonymized RTC user ID of a terminating IMS UE locally generated at the IWF and "userState" field set to "alerting", if the object for the anonymized RTC user ID was not previously sent to the WSF;
        * the object in which "actType" filed set to "mod", "participantId" field set to anonymized RTC user ID of the originating RESPECT endpoint and "userState" field set to "joined";

   b) the "mediaSessionState" key set to:

      - if all the "state" sub-key in "mc" and "dc" sub-key is "true" for "routed" field, then "routed" value;
      - if all the "state" sub-key in "mc" and "dc" sub-key is "true" for "connected" field, then "connected" value;
      - otherwise, "connecting" value;

   c) the other keys required for the msetup response as RESPECT endpoint.

When receiving a successful mupdate response from the WSF, then the IWF needs to send a SIP PRACK request to the IBCF, if a Require header field set to "100rel" was present in the received 18x response.

When receiving a successful response to the SIP PRACK request, then the IWF recognizes the SIP PRACK request was successfully received at the terminating side.

#### 6.9.2.3.2.4 Receiving SIP 18x response not containing SDP

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.2.4-1.

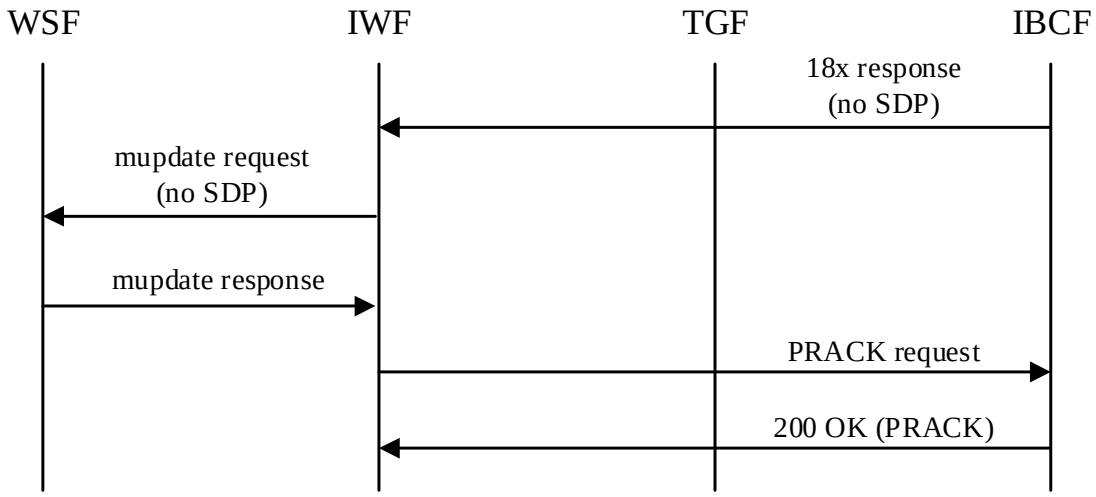


**Figure 6.9.2.3.2.4-1: Receiving SIP 18x response not containing SDP**

When receiving a SIP 180 (Ringing) response or a SIP 183 (Session Progress) response not containing SDP from the IBCF, then the IWF needs to:

1) send an mupdate request to the WSF containing:

   a) the "mediaInfo" key containing:

      - "type" field set to "info";

      - the "participantDesc" sub-key containing:

        * the object in which "actType" filed set to "add", "participantId" field set to anonymized RTC user ID of a terminating IMS UE locally generated at the IWF and "userState" field set to "alerting";

   b) the other keys required for the msetup response as RESPECT endpoint.

When receiving a successful mupdate response from the WSF, then the IWF needs to send a SIP PRACK request to the IBCF, if a Require header field set to "100rel" was present in the received 18x response.

When receiving a successful response to the SIP PRACK request, then the IWF recognizes the SIP PRACK request was successfully received at the terminating side.

#### 6.9.2.3.2.5 Receiving SIP 200 (OK) response containing an initial SDP answer to the initial INVITE request

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.2.5-1.

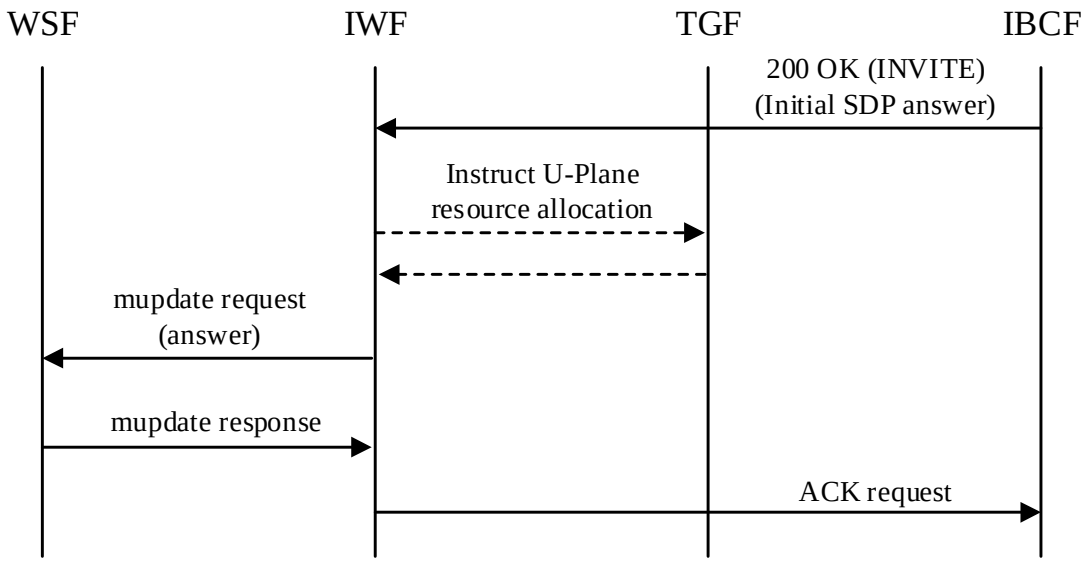


**Figure 6.9.2.3.2.5-1: Receiving SIP 200 (OK) response containing an initial SDP answer to the initial INVITE request**

When receiving a SIP 200 (OK) response containing an initial SDP answer from the IBCF, then the IWF needs to:

1) construct "sdp" sub-key as an initial SDP answer being sent to the WSF based on the stored "preOffer";
2) instruct the IGF to allocate U-Plane resource for the media session based on the received initial SDP answer and the constructed initial SDP answer;
3) send an mupdate request to the WSF containing:
   a) the "mediaInfo" key containing:
      - "type" field set to "answer";
      - the "sdp" sub-key constructed at step 1);
      - if media description(s) of audio and/or video are contained in the constructed "sdp" sub-key, the object(s) per media description(s) in the "metadata" sub-key of the "mc" sub-key containing:
        * "index number" field set to the number corresponding to "preOffer";
        * "actType" field set to either "aly" or "dcl";
        * if "actType" field is set to "aly", the "state" sub-key including both "connected" and "routed" field with appropriate value depending on the answer from IGF at step 2);
      - if a media description of data channel is contained in the constructed "sdp" sub-key, the "dc" sub-key including "sdp index" field set to the number corresponding to "preOffer" and the object(s) per SCTP stream in the "metadata" sub-key containing:
        * "id" field set to the SCTP stream number managed in the TGF;
        * "actType" field set to either "aly" or "dcl";
        * if "actType" field is set to "aly", the "state" sub-key including both "connected" and "routed" field with appropriate value depending on the answer from IGF at step 2);
      - the "participantDesc" sub-key containing:
        * the object in which "actType" filed set to "mod", "participantId" field set to anonymized RTC user ID of a terminating IMS UE and "userState" field set to "joined", if the object for the anonymized RTC user ID was previously sent to the WSF;

* the object in which "actType" filed set to "add", "participantId" field set to anonymized RTC user ID of a terminating IMS UE locally generated at the IWF and "userState" field set to “joined”, if the object for the anonymized RTC user ID was not previously sent to the WSF;

* the object in which "actType" filed set to "mod", "participantId" field set to anonymized RTC user ID of the originating RESPECT endpoint and "userState" field set to "joined";

b) the "mediaSessionState" key set to:

- if all the "state" sub-key in "mc" and "dc" sub-key is "true" for "routed" field, then "routed" value;

- if all the "state" sub-key in "mc" and "dc" sub-key is "true" for "connected" field, then "connected" value;

- otherwise, "connecting" value;

c) the other keys required for the msetup response as RESPECT endpoint.

When receiving a successful mupdate response from the WSF, then the IWF needs to send a SIP ACK request to the IBCF.

#### 6.9.2.3.2.6 Receiving SIP 200 (OK) response not containing SDP to the initial INVITE request

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.2.6-1.

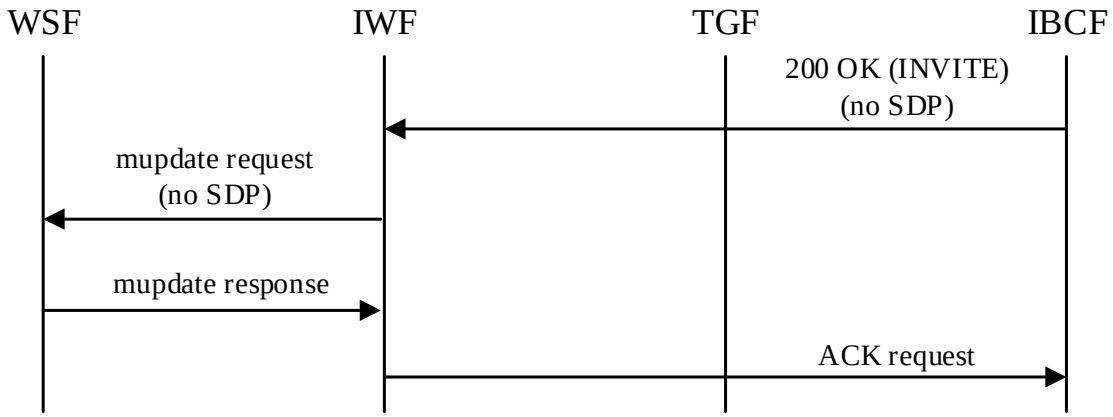


**Figure 6.9.2.3.2.6-1: Receiving SIP 200 (OK) response not containing SDP to the initial INVITE request**

When receiving a SIP 200 (OK) response not containing SDP from the IBCF, then if the IWF already received an initial SDP answer in SIP 18x response the IWF needs to:

1) send an mupdate request to the WSF containing:

a) the "mediaInfo" key containing:

- "type" field set to "info";

- the "participantDesc" sub-key containing:

* the object in which "actType" filed set to "mod", "participantId" field set to anonymized RTC user ID of a terminating IMS UE and "userState" field set to "joined";

b) the other keys required for the msetup response as RESPECT endpoint.

When receiving a successful mupdate response from the WSF, then the IWF needs to send a SIP ACK request to the IBCF.

#### 6.9.2.3.3 Media session setup from IMS to RTC

##### 6.9.2.3.3.1 General

This clause describes the interworking procedures at the IWF for media session setup initiated from IMS network. The procedures in this clause are needed for media session establishment.

In this case, the IWF does not need to manage both the media session state and participant state and will be informed these status from the WSF, since the succeeding entity (i.e., WSF) does aware these RESPECT specific state.

##### 6.9.2.3.3.2 Receiving SIP initial INVITE request containing an initial SDP offer

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.3.2-1.

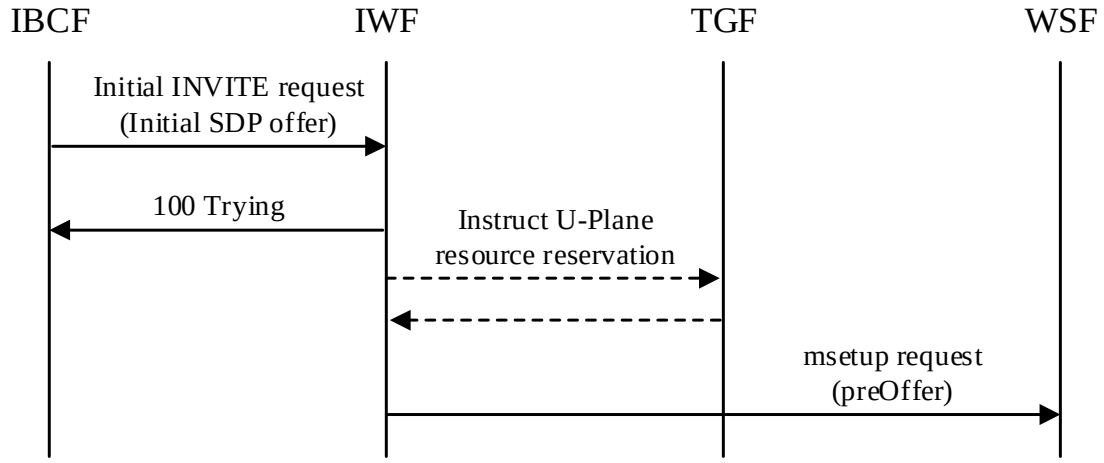


**Figure 6.9.2.3.3.2-1: Receiving SIP initial INVITE request containing an initial SDP offer**

When receiving a SIP initial INVITE request containing an initial SDP offer from the IBCF, then the IWF handles the initial SDP offer as a "preOffer" and then retrieves both the identifier of the destination WSF and RTC ID (RTC user ID or RTC resource ID) corresponding to an MSISDN set in the Request-URI by accessing the internal ASWF.

Upon successful retrieval, the IWF needs to:

1) creates control session for C-Plane signalling if it does not exist;
2) store the received initial SDP offer and construct "preOffer" based on the local policy;
3) instruct the IGF to reserve U-Plane resource for the media session based on the received initial SDP offer and the constructed "preOffer";
4) send a SIP 100 (Trying) response to the IBCF;
5) send an msetup request to the WSF containing:
    a) "uri" field set to either the RTC user ID or the RTC resource ID retrieved in the "dId" key;
    b) "tn" field set to an MSISDN in the "user" sub-key of the "oId" key, if global number digits are present in the received From header field;
    c) "displayName" field set to a display-name in the "user" sub-key of the "oId" key, if display-name is present in the received From header field;
    d) "tn" field set to an MSISDN in the "network" sub-key of the “oId” key, if global number digits are present in the received P-Asserted-Identity header field;
    e) "displayName" field set to a display-name in the "network" sub-key of the "oId" key, if display-name is present in the received P-Asserted-Identify header field;
    f) "privacy" sub-key set to "id" if the Privacy header field set to "id" is received;

g) "identity", "info", "alg" and "ppt" fields set to the corresponding value of the Identity header field, if received;

h) the "mediaInfo" key containing:

- "type" field set to "preOffer";
- the "sdp" sub-key constructed at step 2);
- if media description(s) of audio and/or video are contained in the constructed "sdp" sub-key, the object(s) per media description(s) in the "metadata" sub-key of the "mc" sub-key containing:
  * "index number" field set to the number corresponding to "preOffer";
  * "actType" field set to either "add";
- if a media description of data channel is contained in the constructed "sdp" sub-key, the "dc" sub-key including "sdp index" field set to the number corresponding to "preOffer" and the object(s) per SCTP stream in the "metadata" sub-key containing:
  * "id" field set to the SCTP stream number managed in the TGF;
  * "actType" field set to "add";

i) the other keys required for the msetup request as RESPECT endpoint.

#### 6.9.2.3.3.3 Receiving msetup response not containing SDP with media session state change

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.3.3-1.

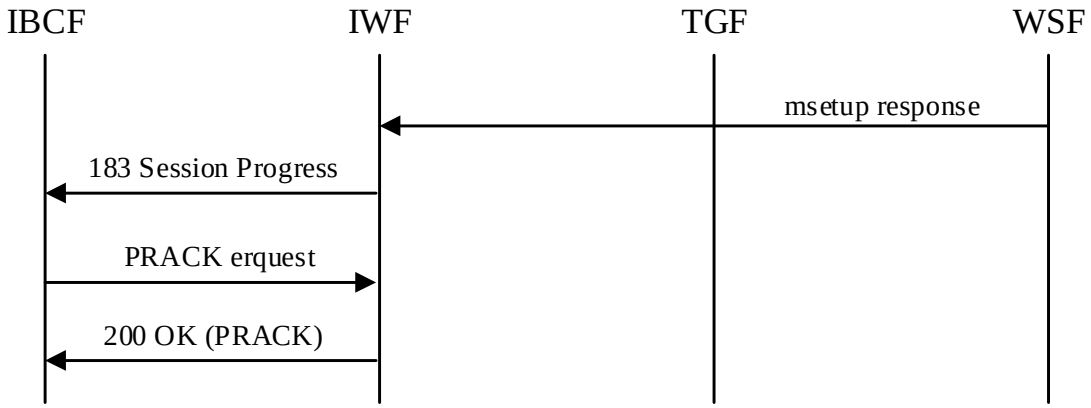


**Figure 6.9.2.3.3.3-1: Receiving msetup response not containing SDP with media session state change**

When receiving a msetup response containing the "mediaSessionState" key set to "accepted", then the IWF needs to send a SIP 183 (Session Progress) response not containing SDP to the initial INVITE request.

When receiving a SIP PRACK request from the IBCF, then the IWF sends a SIP response to the PRACK request.

#### 6.9.2.3.3.4 Receiving mupdate request not containing SDP without media session state change

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.3.4-1.

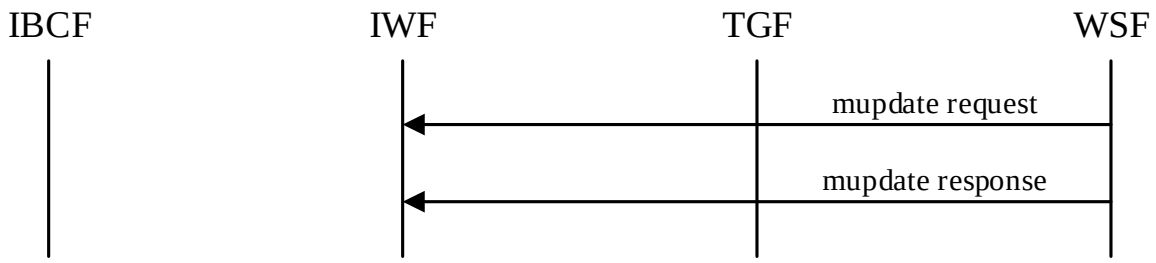


**Figure 6.9.2.3.3.4-1: Receiving mupdate request not containing SDP without media session state change**

When receiving a msetup response from the WSF, if the response;

- does not contain the "mediaSessionState" key; and
- does not contain the "mediaInfo" key or contain the "info" type of "mediaInfo" key;

then the IWF needs to send a mupdate response to the WSF as RESPECT endpoint.

### 6.9.2.3.3.5 Receiving mupdate request containing an "answer" with media session state change

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.3.5-1.

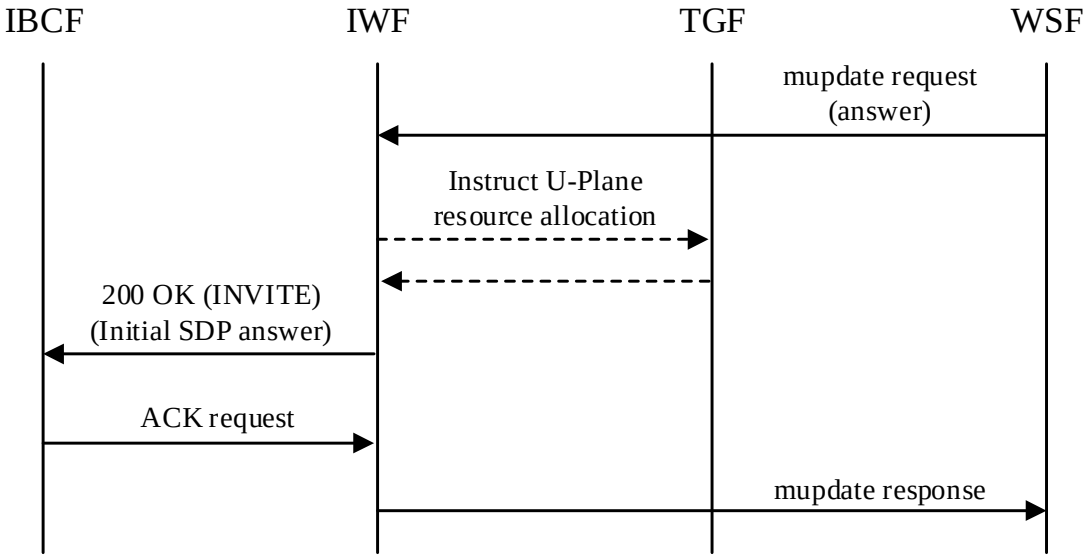


**Figure 6.9.2.3.3.5-1: Receiving mupdate request containing an "answer" with media session state change**

When receiving an mupdate request, if the request;

- contains the "mediaSessionState" key set to either "routed" or "connected"; and
- contains the "answer" type of "mediaInfo" key;

then the IWF needs to:

1) store the received "answer" type of "mediaInfo" key and construct an initial SDP answer based on the stored initial SDP offer;
2) instruct the IGF to allocate U-Plane resource for the media session based on the received "answer" type of "mediaInfo" key and the constructed initial SDP answer; and
3) send a SIP 200 (OK) response to the initial INVITE request containing the constructed initial SDP answer.

When receiving SIP ACK request from the IBCF, the IWF needs to send a msetup response to the WSF containing the keys required for the mupdate request as RESPECT endpoint.

#### 6.9.2.3.3.6 Receiving mupdate request containing an "offer" without media session state change

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.3.6-1.

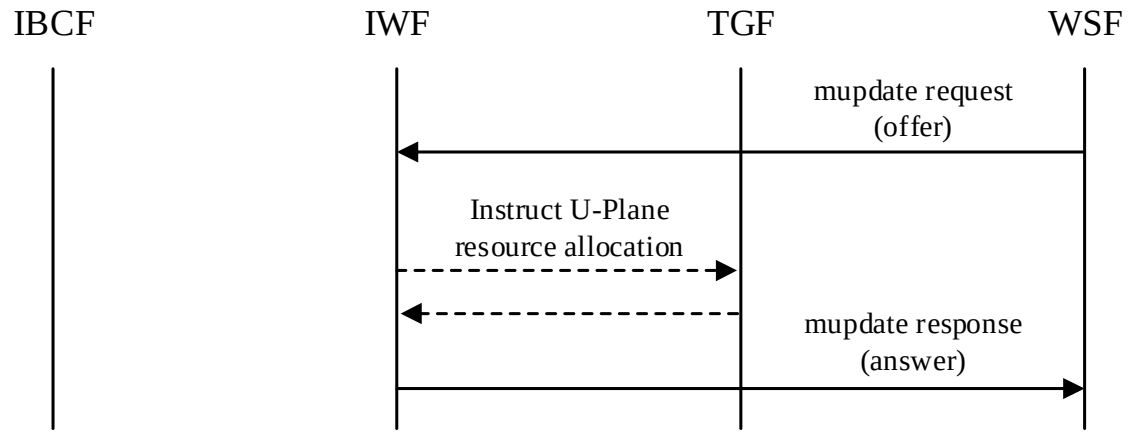


**Figure 6.9.2.3.3.6-1: Receiving mupdate request containing an "offer" without media session state change**

When receiving an mupdate request, if the request;

- does not contain the "mediaSessionState" key; and
- contains the "offer" type of "mediaInfo" key;

then the IWF needs to:

1) store the received "offer" type of "mediaInfo" key;
2) construct an "answer" type of the "sdp" sub-key of the "mediaInfo" key based on the received "mediaInfo" key;
3) construct an initial SDP answer based on the stored SDP offer and store the constructed SDP answer;
4) instruct the IGF to allocate U-Plane resource for the media session based on the received "offer" type of "mediaInfo" key, the constructed "answer" type of "mediaInfo" key and the constructed initial SDP answer; and
5) send an mupdate response containing:
   a) the "mediaInfo" key containing:
      - "type" field set to "answer";
      - the "sdp" sub-key constructed at step 2);
      - if media description(s) of audio and/or video are contained in the constructed "sdp" sub-key, the object(s) per media description(s) in the "metadata" sub-key of the "mc" sub-key containing:
        * "index number" field set to the number corresponding to "offer";
        * "actType" field set to either "aly" or "dcl";
      - if a media description of data channel is contained in the constructed "sdp" sub-key, the "dc" sub-key including "sdp index" field set to the number corresponding to "offer" and the object(s) per SCTP stream in the "metadata" sub-key containing:
        * "id" field set to the SCTP stream number managed in the TGF;
        * "actType" field set to either "aly" or "dcl";

b) the other keys required for the msetup response as RESPECT endpoint.

the constructed "answer" type of "mediaInfo" key.

#### 6.9.2.3.3.7 Receiving mupdate request not containing SDP with media session state change

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.3.7-1.

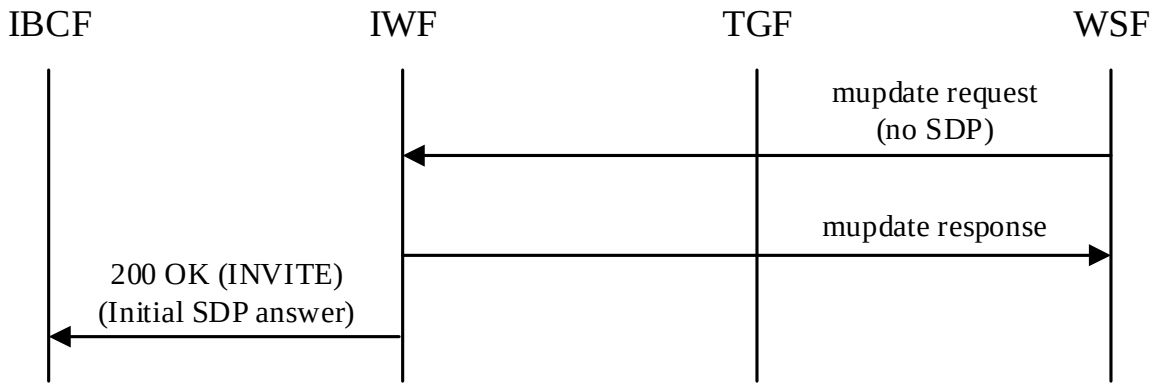


**Figure 6.9.2.3.3.7-1: Receiving mupdate request not containing SDP with media session state change**

When receiving an mupdate request, if the request;

- contains the "mediaSessionState" key; and
- does not contain the "mediaInfo" key or contain the "info" type of "mediaInfo" key;

then the IWF needs to:

1) send an mupdate response containing the keys required for the mupdate response as RESPECT endpoint; and
2) send a SIP 200 (OK) response to the initial INVITE request containing the stored initial SDP answer, if the IWF does not previously send the response to the IBCF.

When receiving SIP ACK request from the IBCF, the IWF recognizes the SIP 200 (OK) response to the initial INVITE was successfully processed at IMS side.

### 6.9.2.3.4 Media session set up callcellation from RTC to IMS

#### 6.9.2.3.4.1 General

This clause describes the interworking procedures at the IWF for media session setup cancellation from RTC network.

#### 6.9.2.3.4.2 Receiving mdisc request for establishing media session

The simple call flow addressed in this procedure clause is shown in Figure 6.9.2.3.4.2-1.

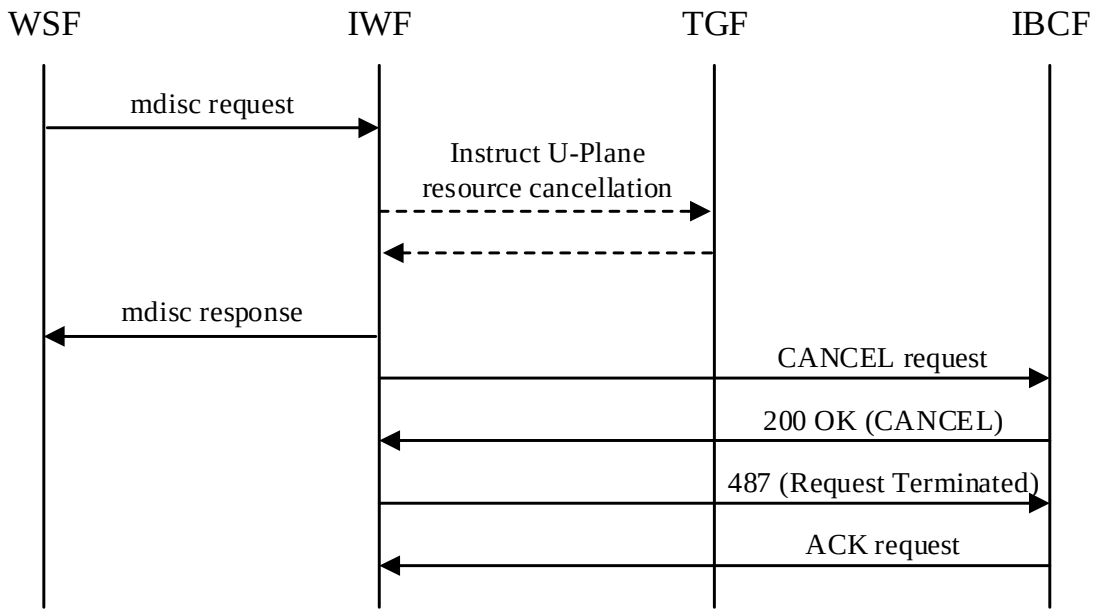


**Figure 6.9.2.3.4.2-1: Receiving mdisc request for establishing media session**

When receiving an mdisc request for a media session which has not been reached the media session status "connected" or "routed", then the IWF needs to:

1) instruct the IGF to cancel U-Plane resource for the media session;
2) send an mdisc response to the WSF; and
3) send a SIP CANCEL request for the INVITE transaction to the IBCF.

When receiving a SIP 200 (OK) response to the SIP CANCEL request, then IWF needs to send a SIP 487 (Request Terminated) response as a final SIP response to the IBCF. Then, the IWF will receive the ACK request from the IBCF.

### 6.9.2.3.5 Media session set up callcellation from IMS to RTC

#### 6.9.2.3.5.1 General

This clause describes the interworking procedures at the IWF for media session setup cancellation from IMS network.

#### 6.9.2.3.5.2 Receiving SIP CANCEL request for establishing media session

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.5.2-1.

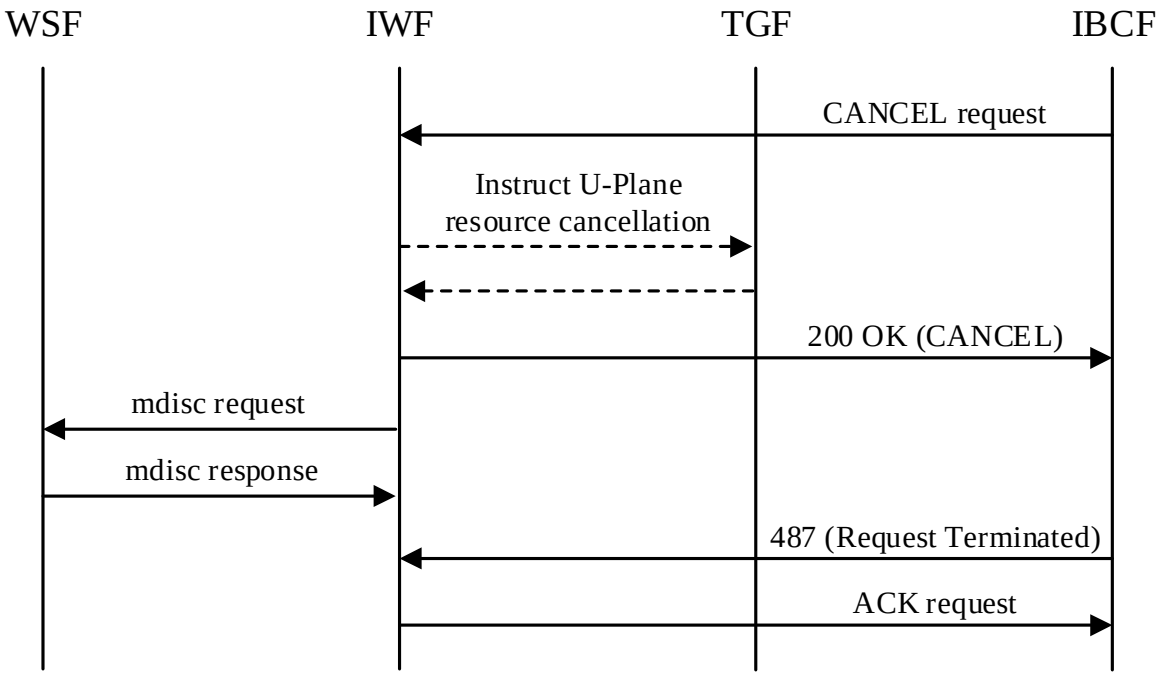


**Figure 6.9.2.3.5.2-1: Receiving SIP CANCEL request for establishing media session**

When receiving a SIP CANCEL request for a media session, then the IWF needs to:

1) instruct the IGF to cancel U-Plane resource for the media session;
2) send a SIP 200 (OK) response to the SIP CANCEL request to the IBCF; and
3) send an mdisc request to the IWF.

When receiving a SIP 487 (Request Terminated) response as a final SIP response from the IBCF, then IWF needs to send ACK request to the IBCF.

### 6.9.2.3.6 Media session update from RTC to IMS

The interworking procedures at the IWF essential for the media session update in the aspect of RTC-IMS interworking are not identified for the time being.

### 6.9.2.3.7 Media session update from IMS to IMS

The interworking procedures at the IWF essential for the media session update in the aspect of RTC-IMS interworking are not identified for the time being.

### 6.9.2.3.8 Media session release from RTC to IMS

#### 6.9.2.3.8.1 General

This clause describes the interworking procedures at the IWF for media session release of established media session from RTC network.

#### 6.9.2.3.8.2 Receiving mdisc request for established media session

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.8.2-1.

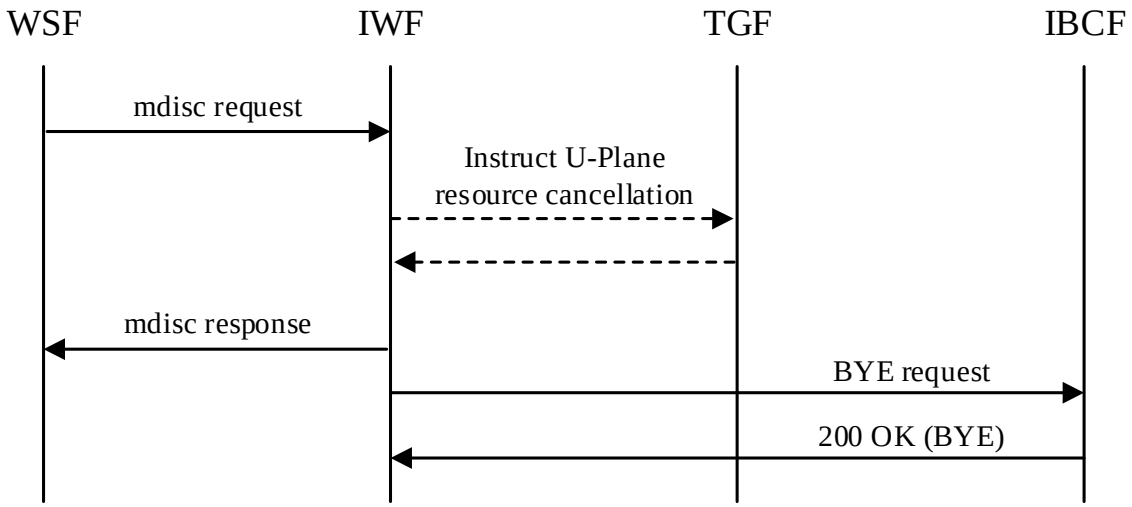


**Figure 6.9.2.3.8.2-1: Receiving mdisc request for established media session**

When receiving an mdisc request for a media session in either "connected" or "routed" media session state from the WSF, the IWF needs to:

1) instruct the IGF to deallocate U-Plane resource for the media session;
2) send an mdisc response to the WSF containing the keys required for the response as RESPECT endpoint; and
3) send a SIP BYE request to the IBCF.

When receiving a successful response to the SIP BYE request, then the IWF recognizes the SIP BYE request was successfully processed on the IMS side.

### 6.9.2.3.9 Media session release from IMS to RTC

#### 6.9.2.3.9.1 General

This clause describes the interworking procedures at the IWF for media session release of established media session from IMS network.

#### 6.9.2.3.9.2 Receiving SIP BYE request for established media session

The simple call flow addressed in this procedure is shown in Figure 6.9.2.3.9.2-1.

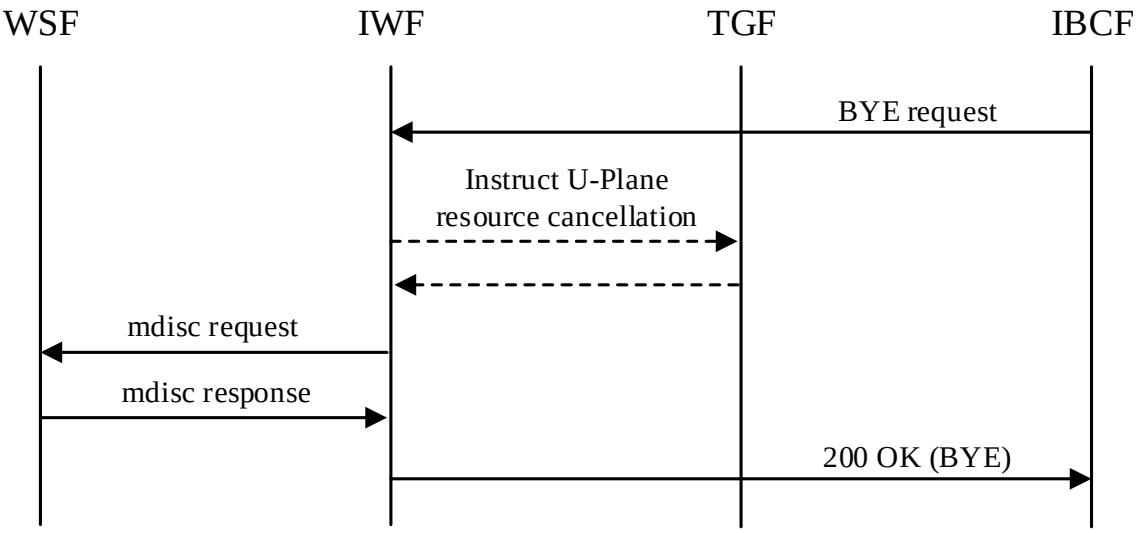


**Figure 6.9.2.3.9.2-1: Receiving SIP BYE request for established media session**

When receiving a SIP BYE request for a media session in either "connected" or "routed" media session state from the IBCF, the IWF needs to:

1) instruct the IGF to deallocate U-Plane resource for the media session; and
2) send an mdisc request to the WSF.

When receiving a successful response to the mdisc request, then the IWF sends a SIP 200 (OK) response to the BYE request towards the IBCF.

## 6.9.3 U-Plane media related interworking

### 6.9.3.1 General

This clause describes the followings as U-Plane media related interworking specification at the TGF:

- Protocol stack
- RTC media mixising for IMS

### 6.9.3.2 Protocol stack

Figure 6.9.3.2-1 shows the protocol stack for U-Plane media related interworking between RTC and IMS networks.

The IGF provides the following functionalities for U-Plane:

- U-Plane Protocol over the UDP interworking
- Coded transcoding
- NAPT (Network Address Port Translation), IPv4 - IPv6 conversion
- Termination of capabilities which is not supported by the other side. (e.g., multiplexing RTP data and control packets on a single port as specified in IETF RFC 5761 [24C] and IETF RFC 8035 [32B] may not be supported in the IMS side.)

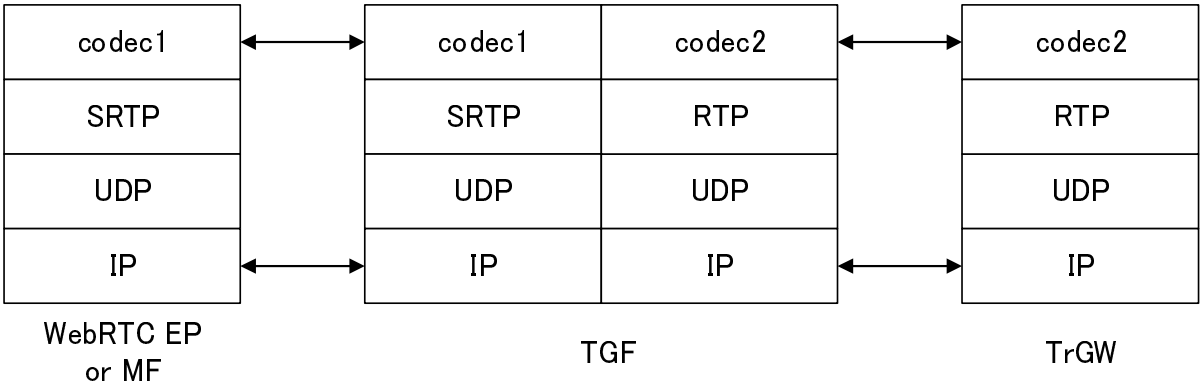


**Figure 6.9.3.2-1: Protocol interworking between RTC media and IMS media**

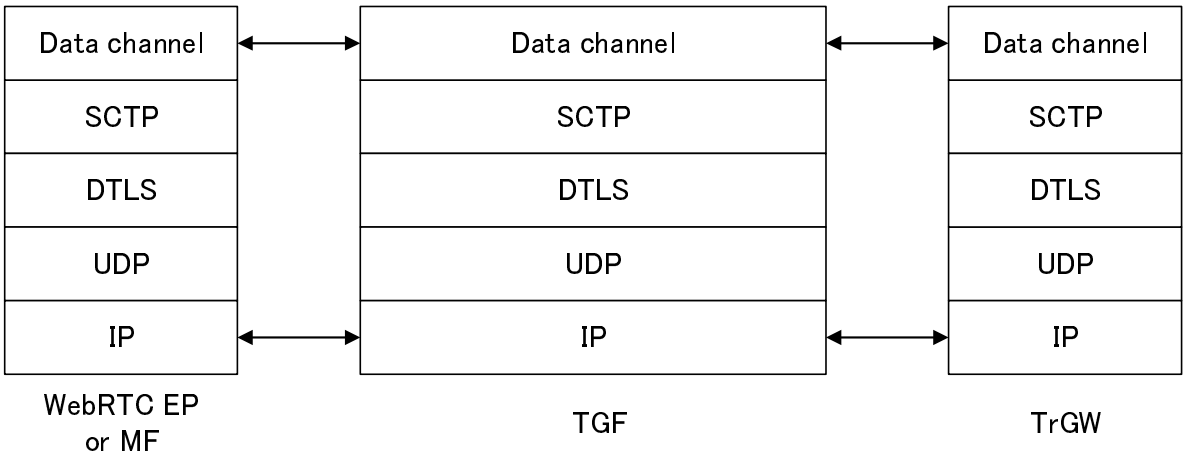


**Figure 6.9.3.2-2: Protocol interworking between RTC data channel and IMS data channel**

TGF is not expected to transcode WebRTC data channel related protocols, since DCMTSI client in IMS network supports IMS data channel which complies to WebRTC data channel as described in 3GPP TS 26.114 [11]. If the called MTSI client is not DCMTSI client, then the data channel related SDP offer is ignored at the MTSI client.

NOTE: Data channel related SDP offer might be removed or transcoded at the IWF or IBCF, based on the operator policy and the inter-operator agreements.

### 6.9.3.3 RTC media mixising for IMS

When SFU is used for the conference media session in the RTC network, a media stream will be generated from each RTC endpoint (UE) and sent all the media streams to each RTC endpoint (UE). Therefore, the RTC endpoint (UE) needs to receive media streams from all the participants in the conference and perform SDP offer/answer every time a new participant joins the conference. However, an IMS or IMS UE not expecting frequent media changes or huge media streams may not be able to handle the above SDP offer/answer correctly.

Therefore, the IGF needs to perform compose multiple media streams from RTC network to a single media stream for IMS media session based on the instruction from the IWF as described in clause 6.9.2.

## 6.9.5 Solution evaluation

This solution proposes the protocol-level interworking between RTC network and IMS network, based on the proposed functional requirements and architecture in Solution #4. The proposed solution realizes the interconnecttion of media session between RTC network and IMS network using existing IMS specification, and it is confirmed that the solution is feasible for nomal cases. Therefore, it is proposed to use this specification as a basis of stage 3 normative work.

# 6.10 Solution #9: Tethered cases

## 6.10.1 Solution description

This solution addresses Key Issue #9.

For sub key issue #9-2 (WebRTC Endpoint should be whether on the tethered device or on the tethering device), there are three design options:

- Solution #9-2-1:
  The WebRTC Endpoint resides on the tethering device (e.g., on the phone). In this case, the tethered device (e.g., AR glasses) serves as a display (for video and audio).

- Solution #9-2-2:
  The WebRTC Endpoint resides on the tethered device (e.g., on the AR glasses). In this case, the tethering device (e.g., the phone) serves as a relay.

- Solution #9-2-3:
  The WebRTC Endpoint is split into two parts: one is the application (which is called WebRTC Endpoint App) residing on the tethered device, and the other one (which is called WebRTC Endpoint Support Function) is the signalling functions that communicate with the support functions in the eiRTCW architecture, residing on the tethering device.
  The WebRTC Endpoint App maps to the Native WebRTC App or the Web App in the RTC general architecture, and WebRTC Endpoint Support Function maps to RTC endpoint in the general architecture shown in Figure 6.10.1-1.
  A new interface Rt-u is created that allows the communication between the two parts. The Rt-u interface performs functions similar to those performed by the RTC-6, the RTC-7 and the WebRTC API interfaces in the iRTCW architecture. However, since the Rt-u interface is not within the same device (i.e., the UE), it involves the setup of a communication channel that may be defined by a protocol number and two port numbers.

Solution #9-2-3 is preferred because it allows the 5G system to have more control over the session (compared to Solution #9-2-2 while providing better QoS (compared to Solution #9-2-1).

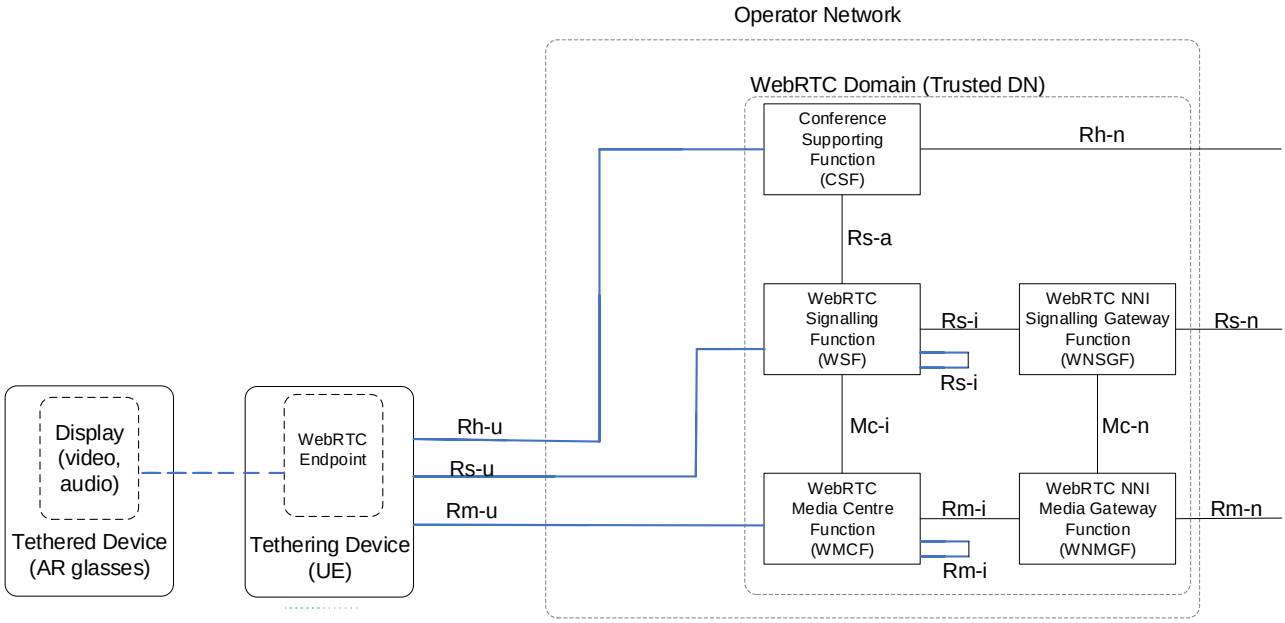


**Figure 6.10.1-1: Solution #9-2-1: WebRTC Endpoint resides on the tethering device**

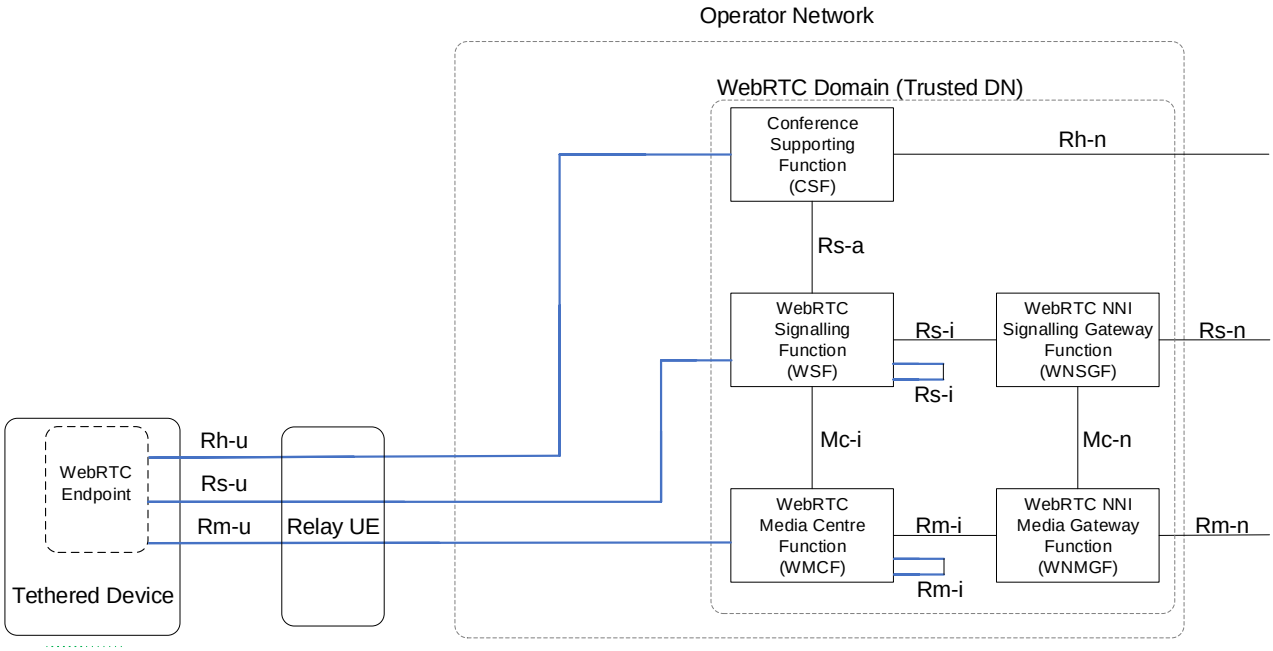


**Figure 6.10.1-2: Solution #9-2-2: WebRTC Endpoint resides on the tethered device**

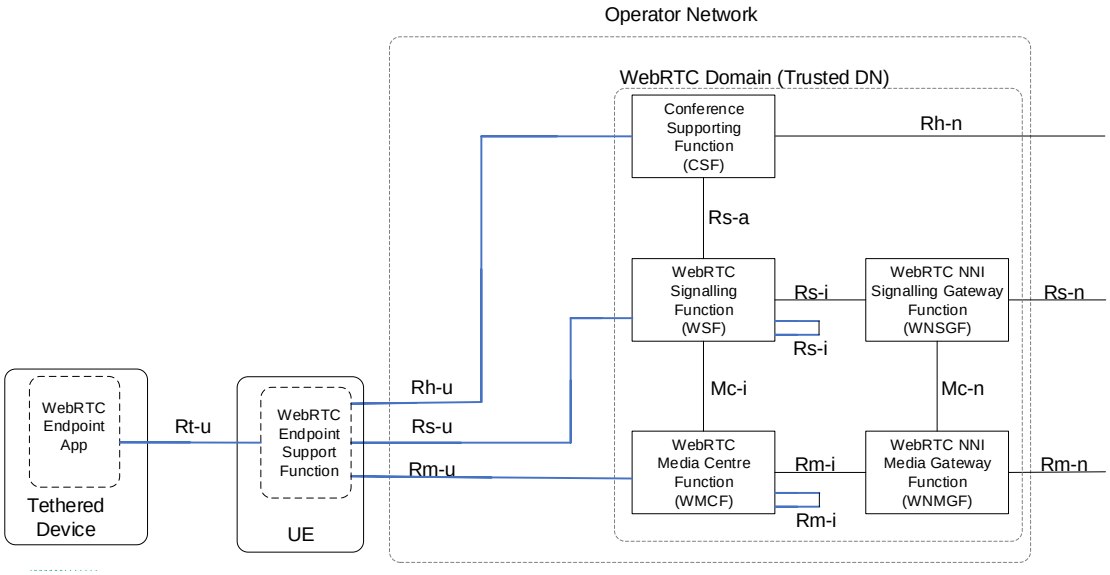


**Figure 6.10.1-3: Solution #9-2-3: WebRTC Endpoint is split, and the first part WebRTC Endpoint App resides on the tethered device and the second part WebRTC Endpoint Support Function resides on the tethering device.**

NOTE 1: When the WebRTC Endpoint App corresponds to the Web App, the Rt-u interface maps to the WebRTC API interface in the general RTC architecture. When the WebRTC Endpoint App corresponds to the Native WebRTC App, the Rt-u interface maps to the RTC-6 and the RTC-7 interfaces in the general RTC architecture. Rs-u and Rm-u will go through the UE and Rt-u to the WebRTC Endpoint App for collaboration scenario 3 and collaboration scenario 4.

NOTE 2: The Rh-u, Rs-u and the Rm-u interfaces are the same as in the architecture described in clause 6.2 in this document.

NOTE 3: Rt-u in the RTC architecture is FFS.

### 6.10.2 Solution evaluation

This solution addresses sub Key Issue #9-2 and proposes possible tethered architecture for tethering/tethered devices. However, there are remaining sub Key Issues that could be further study.

## 6.11 Solution #10: Security considerations

### 6.11.1 Solution Description

This solution addresses Key Issue #10.

As a solution to deliver the trusted user identity in the RTC networks, it is reasonable for the RTC user identity to adopt the concept of "trust domain" defined in IETF RFC 3324 [22] which is constructed by human, since this concept has been widely used in the IMS network to use/provide the trusted user identity for telephony services.

In some case, the solution may need more reliability/strength for the interconnection scenario (i.e., collaboration scenario 4), since the "trust domain" is constructed by human being (e.g., thorough bi-lateral agreement). To address this case, this solution addresses the adaptation of "Calling number verification using signature verification and attestation information" used for the IMS interconnection defined in 3GPP TS 29.165 [16].

In the subsequent clauses, the followings are described as a solution:

- Adaptation of the trust domain (clause 6.11.2)
- Network-asserted identity within the trust domain (clause 6.11.3)
- Adaptation of calling number verification using signature verification and attestation information (clause 6.11.4)

### 6.11.2 Adaptation of the trust domain

In the RTC network conforms to this document, a member of a trust domain consists of the C-Plane functional entities (i.e., WSF and IWF). C-Plane functional entities that belongs to the other RTC network can be a member of the trust domain only if there is an inter-connection agreements.

All the functional entity in the trust domain needs to comply with a clearly defined specification set and a functional entity in the trust domain needs to know that all the functional entities in the trust domain will behave as defined in the specification set through configuration information. This specification set is termed "Spec(T)" in IETF RFC 3324 [22].

In this document, "Spec(T)" is given per feature or information element(s) in C-Plane signalling message. Also, the concept of the trust domain is intended for the identity related problem, but this document may extend the use of the trust domain for the protection of sensitive information within the operator network.

### 6.11.3 Network-asserted identity within the trust domain

As a network-asserted identity of an originating RTC user, this document supports the following three types of identifies and these are contained in the "network" object of "oId" object as defined in this document:

- RTC user ID (URI format),
- Telephone number (global number digits excluding "+"), and
- Display name.

These are identities initially derived by a functional entity in an RTC network (i.e., WSF) as a result of an authentication process over C-Plane signalling, and these are not RTC user-provided identities.

This document also provides the privacy mechanism just like IETF RFC 3325 [23], by which an originating UE indicates the privacy to prevent presentation of its identities to a WebRTC endpoint of final destination. Based on this privacy information, a functional entity on the terminating side within a trust domain will handle (forward, delete or anonymize) the originating RTC user's identities.

This feature is a target of trust domain and a "Spec(T)" for this feature is given as follows:

1. Protocol requirements

   - The protocol specification specific to this feature (handling of "network" and "privacy" object in "oId" object) described in this document needs to be supported.

2. Authentication requirements

   - RTC users need to be authenticated through authentication procedures using the signalling protocol described in this document. As an authentication mechanism, the signalling protocol in this document applies bearer authentication, basic authentication, and digest authentication.

3. Security requirements

   - Members within the trust domain needs to be in trusted DN and use the Secure WebSocket for transport of signalling messages.

4. Scope of trust domain for this feature

   - A trust domain for the network-asserted identity consists of C-Plane functional entities (i.e., WSF and IWF) in an RTC network. In addition to this, C-Plane functional entities that belongs to the other RTC network can be a member of the trust domain only if there is an inter-connection agreements on this feature.

### 6.11.4 Adaptation of calling number verification using signature verification and attestation information

For the interconnection scenario (i.e., collaboration scenario 4), this document adapts a framework "calling number verification using signature verification and attestation information" used for the IMS interconnection defined in 3GPP TS 29.165 [16] with an extension. In this framework, an originating RTC network sets a signature for an originating RTC user identity into a setup request of C-Plane signalling and then a terminating RTC network within a trust domain validates the signature. This signature is generated at an originating RTC network by constructing "PASSporT (Personal Assertion Token)" JSON object and signing a hash of this JSON object with private key associated with the appropriate credential for the identity. At the terminating RTC network, the received signature is validated by using public key.

While target of verification is a telephone number of an originating user in IMS interconnection, this document extends its use to validating both RTC user ID (URI format) and telephone number.

NOTE: The mechanism defined in IETF RFC 8224 [34] are used in IMS network, and this mechanism works not only for telephone-number but also URI.

As an extension in this framework, this document also applies the feature to provide the levels of attestation to a terminating RTC network as with IMS network.

This feature is a target of trust domain and a "Spec(T)" for this feature is given as follows:

1. Protocol requirements

   - The protocol specification specific to this feature (handling of "passport" object in "oId" object) described in this document needs to be supported.

2. Authentication requirements

   - RTC users need to be authenticated through authentication procedures using the signalling protocol described in this document. As an authentication mechanism, the signalling protocol in this document applies bearer authentication, basic authentication, and digest authentication.

3. Security requirements

   - Members within the trust domain needs to be in trusted DN and use the Secure WebSocket for transport of signalling messages.

4. Scope of trust domain for this feature

- A trust domain for the network-asserted identity consists of C-Plane functional entities (i.e., WSF and IWF) in an RTC network. In addition to this, C-Plane functional entities that belongs to the other RTC network can be a member of the trust domain only if there is an inter-connection agreements on this feature.

## 6.12 Solution #11: Related groups considerations

### 6.12.1 Solution description

This solution addresses Key Issue #11.

As for the stage 2 specification:

- Any necessary considerations for other 3GPP WG's stage 2 specifications are not identified since SA4 is responsible for the network architecture for RTC media services, thus the other 3GPP WG groups do not specify that network architecture.

- Also, any necessary considerations for other organizations are not identified since the stage 2 descriptions of this document are 3GPP-specific and have no impact on any specifications developed in the other organizations (i.e., IETF and W3C).

As for the stage 3 (e.g., C-Plane signalling protocol, API):

- The proposals in this document do not conflict with the IETF RFCs (i.e., referred IETF specifications in this document such as IETF RFC 8825 [44]) and W3C WebRTC 1.0 [65], therefore any considerations for other organizations are not needed.

### 6.12.2 Solution evaluation

There is no impact on the specifications of other WGs in 3GPP and organizations including IETF and W3C.

# 7 Key findings

## 7.1 General

In this clause, the solution evaluations in technical aspects for enabling collaboration scenario 4 and collaboration scenario 3 are summarized as conclusions of this Study Item (FS_eiRTCW).

## 7.2 Stage-2 aspect

The architecture diagram that best represents the architectural solution is shown in Figure 7.2-1. This architectural diagram consists of functional entities already defined in 3GPP TS 26.506 [12]. While Network Support function (NS-AF) is snipped in the figure, this document expects RTC AF to be integrated with WebRTC Signalling Function (WSF) to interact with 5GC via N5 interface, as indicated in 3GPP TS 26.506 [12] clause 4.2.5.

Also, as an extension of collaboration scenario 4, this document addresses the interworking between RTC network and IMS network over NNI which conforms to 3GPP TS 29.162 [15].

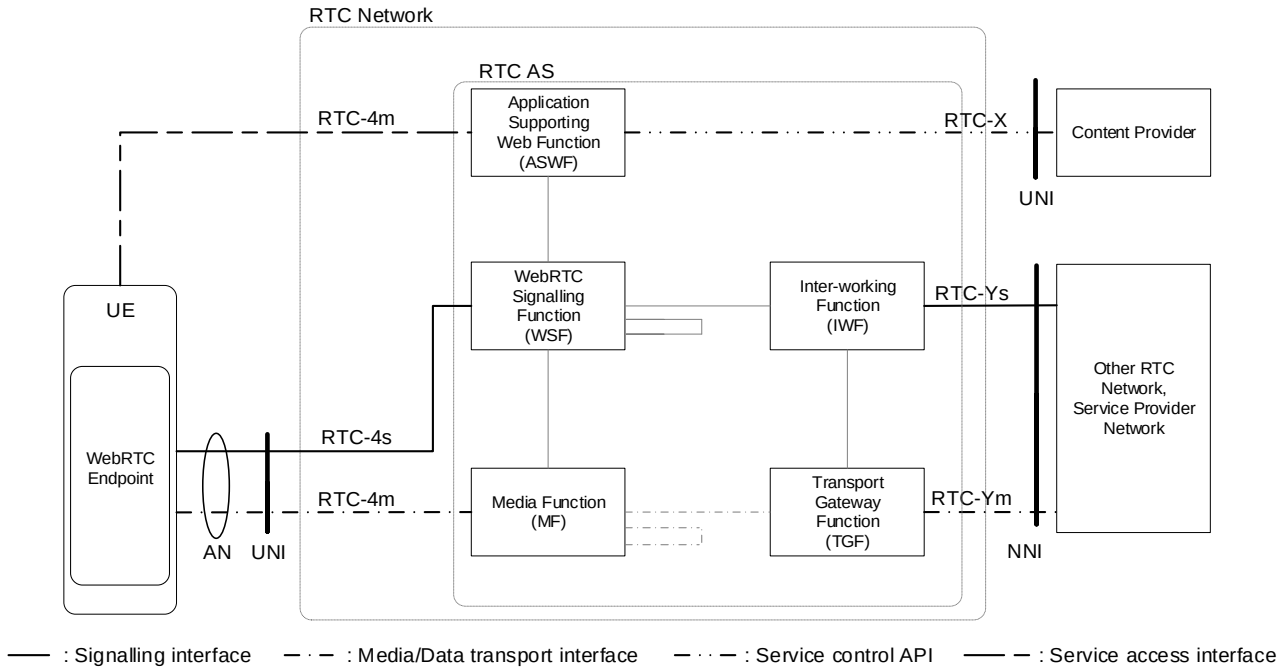


**Figure 7.2-1: Derivative RTC architecture diagram**

In order to enable both collaboration scenario 4 and collaboration scenario 3 defined in 3GPP TS 26.506 [12], this document identifies that the following enhancements are needed:

A) New reference points

- This document identifies that the following reference points are required to be newly introduced, in addition to the existing reference point defined in 3GPP TS 26.506 [12] clause 4.3. These reference points need to be assigned/defined considering the common architecture for 5GMS and RTC.

  1) RTC-X

     * To enable the immersive RTC service in collaboration with content providers, the interface between a ASWF and a content provider network for service control is needed. Over this reference point, ASWF is expected to provide the service control APIs.

  2) RTC-Y

     * To enable the collaboration scenario 4 supporting interoperability between multiple RTC networks, the NNI specification is needed.

     * RTC-Y is divided into two sub-interfaces, RTC-Ys (for C-Plane signalling) and RTC-Ym (U-Plane media/data transport).

B) Functional enhancements on the existing entities

- This document identifies that the functional enhancements on the existing entities shown in Table 7.2-2 are required against the functional definitions in 3GPP TS 26.506 [12] clause 4.2. This table does not show all the enhancements, but major enhancements identified in this document.

- This document does not identify the new functional entity needed.

**Table 7.2-2: Major expected functional enhancements**

| Entity | Expected functional enhancement | Interface | Solution |
|---|---|---|---|
| UE | Support of WSF discovery mechanism at UE. | RTC4m | Sol#6 |
| WSF | Interaction with MF for media session control. | - | - |
| | Interaction with ASWF for collaboration with web applications/services. | - | - |
| | Interaction with 5GC, using network Support function (NS-AF). | N5 | Sol#1 |
| | Authentication/authorization of the UE. | RTC-4s | Sol#2 |

| | | | |
|---|---|---|---|
| | Support of functionalities needed for service control.<br>- Connection control enforcer<br>- RTC ID resource handling enforcer | RTC-X | Sol#5 |
| | Signing and verification of network-asserted UE's ID. | RTC-Ys | Sol#10 |
| MF | Support of functionalities needed for service control.<br>- Media data forwarding control enforcer<br>- RTC exchange resource handling enforcer | RTC-X | Sol#5 |
| ASWF | Exposing the service control APIs. | RTC-X | Sol#5 |
| | Storage of user subscription data specific to MNO's WebRTC services. | - | - |
| | Providing supplementary files via best-effort transport different from the channels for real-time media. | RTC-4m | Sol#1 |
| | Providing WSF discovery functionality. | RTC-4m | Sol#6 |
| IWF | C-plane signalling protocol interworking between RTC network and IMS network. | - | Sol#7 |
| | Signing and verification of network-asserted UE's ID. | RTC-Ys | Sol#10 |
| TGF | U-Plane protocol interworking between RTC network and IMS network. | - | Sol#7 |

## 7.3 Stage-3 aspect

Based on the architectural and functional enhancements as summarized in clause 7.1, the following stage-3 level solutions are studied in this document.

1) A new C-Plane signalling protocol - RESPECT (Solution #3)
2) A new service control APIs (Solution #5)
3) A new mechanism for WSF discovery (Solution #6)
4) Interworking specifications at IGF/TGF for RTC-IMS inter-connection (Solution #8)

It is confirmed that all the solutions are technically feasible.

# Annex A (informative): Use cases

## A.1 General

In this annex, the following typical use cases targeted in this document are described.

- VR streaming in first person view (see clause A.2)
- VR streaming in third person view (see clause A.3)
- Voice communication and media analytics (see clause A.4)
- VR streaming over NNI (see clause A.5)
- Peer-to-peer communication (see clause A.6)

## A.2 VR streaming in first person view

### A.2.1 General

This use case is an example where a UE connects to an RTC ID resource (i.e., VR space) and receives audio/video media rendered by a content provider based on the user's operations (e.g., scroll, zoom out). In this use case, the user enjoys media streamed by the content provider in first person view.

### A.2.2 Service flows

1. A user visits a website provided by a content provider and sign-in to an account through authentication. After successful sign-in, the necessary JavaScript files are downloaded to the web browser in the UE and the procedure for establishing a media session with data network (RTC ID resource) is started when the user takes a certain action on the web browser. In this procedure, the content provider assigns the RTC ID resource (URI) which is the destination of media session connection, requests the operator to configure the URI, and informs the UE of the URI.
2. The content provider starts to set up a media session with data network using the URI after receiving a notification indicating the attempt of session establishment by the UE from the operator. After establishing the media session between the UE and the data network and between the content provider and the data network, the content provider sends audio and video media to the data network and exchanges the data via datachannel (Figure A.2.2-1). Accordingly, the user enters the personal room in the VR space.
3. The UE sends an application-specific data to content provider via the datachannel. Based on this data, the content provider renders media (e.g., displayed video media, BGM).

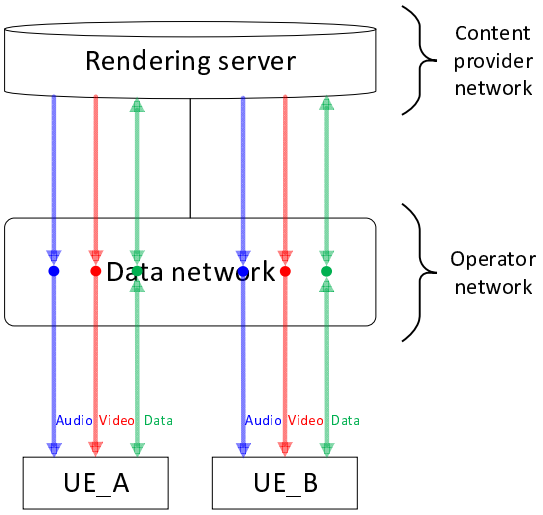


**Figure A.2.2-1: Media/data flows for VR streaming in first person view**

# A.3 VR streaming in third person view

## A.3.1 General

This use case is an example where multiple UEs connect to an RTC ID resource (i.e., VR room) and receive audio/video media rendered by a content provider based on the user's operations (e.g., scroll, zoom out). In this use case, the users enjoy same media streamed by the content provider in third person view.

## A.3.2 Service flows

1. The content provider establishes the media session with data network for streaming media for the third person view.

2. UE_A and UE_B establish the media sessions with the data network according to the flows described in clause A.2.2. After establishing the media sessions, the content provider sends audio and video media to the data network and exchanges the data via datachannel (Figure A.3.2-1). Accordingly, the users enter the VR space.

3. Each UEs sends the application-specific data to content provider via the datachannel. Based on this data, the content provider renders media (e.g., displayed video media, BGM). User_A and User_B see the same view (e.g., own avatar's moving and other user's avatar moving) in third person view if the users are in the same square in the VR space.

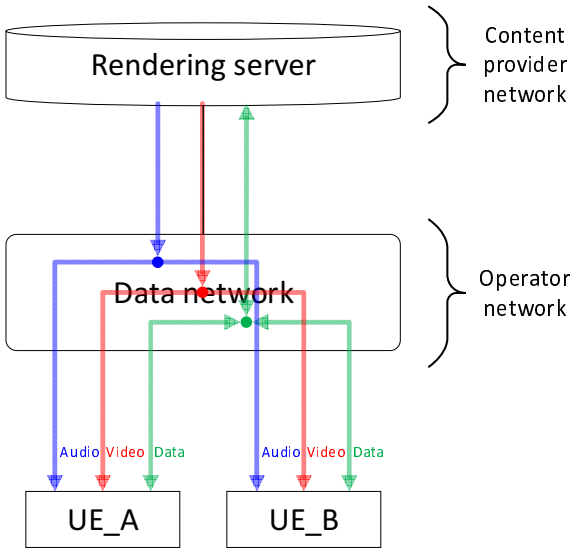


**Figure A.3.2-1: Media/data flows for VR streaming in third person view**

# A.4 Voice communication and media analytics

## A.4.1 General

This use case is an example where multiple users talk each other by connecting RTC ID resource_2 (i.e., conference room) after connecting to RTC ID resource_1 (i.e., VR space) according to the service flows described in clause A.3.2.

In this example, an analytics server exists in the content provider network in order to analyze audio and video media sent from the UEs and reflect the analysis result to media provided by the rendering server. The analytics server sends the analysis result to a rendering server in proprietary manner and based on the analysis result the content provider renders media (e.g., displayed video media, BGM). Audio/video media and data are not exchanged between the analytics server and the rendering server.

## A.4.2 Service flows

1. The analytics server in the content provider establishes the media session with the data network for analyzing upstreaming media (e.g., user's video and user's voice) from UE(s).

2. UE_A and UE_B establish the media sessions with the data network according to the flows described in clause A.3.2. In this procedure, the UE informs the data network not to send video media to anywhere if the user unwanted. After establishing the media sessions, the UEs in the same VR space can talk each other (Figure A.4.2-1).

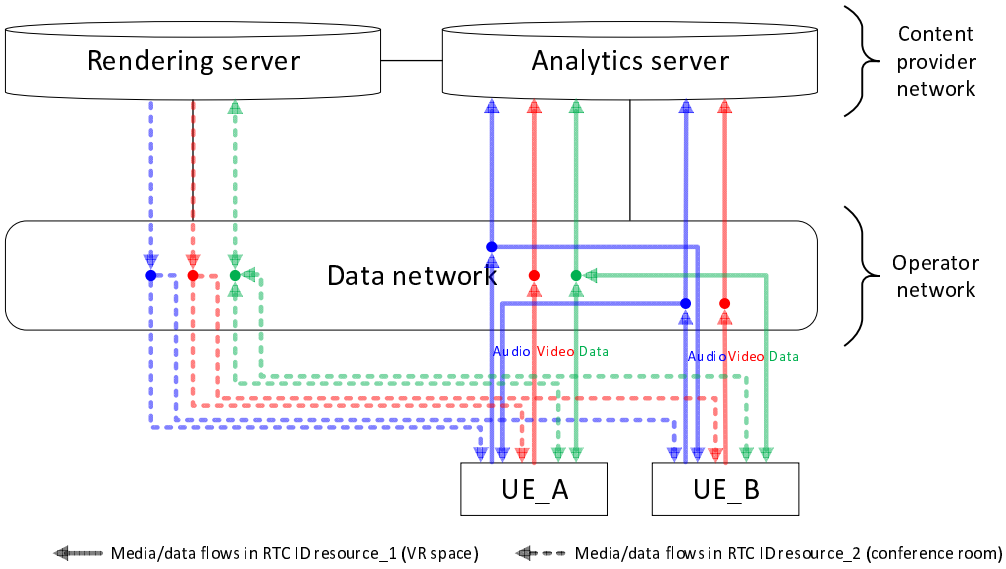


**Figure A.4.2-1: Media/data flows for voice communication and media analytics**

# A.5 VR streaming over NNI

## A.5.1 General

This use case is an example where a UE connects to an RTC ID resource (i.e., VR space) and receives audio/video media rendered by a content provider accommodated by different operator than that UE attached to, based on the user's operations (e.g., scroll, zoom out). In this use case, the user enjoys media streamed by the content provider in first person view.

## A.5.2 Service flows

1. There is no difference with service flows in clause A.2.1 except that the operator accommodating the content provider differs from that the UE attached to (Figure A.5.2-1).

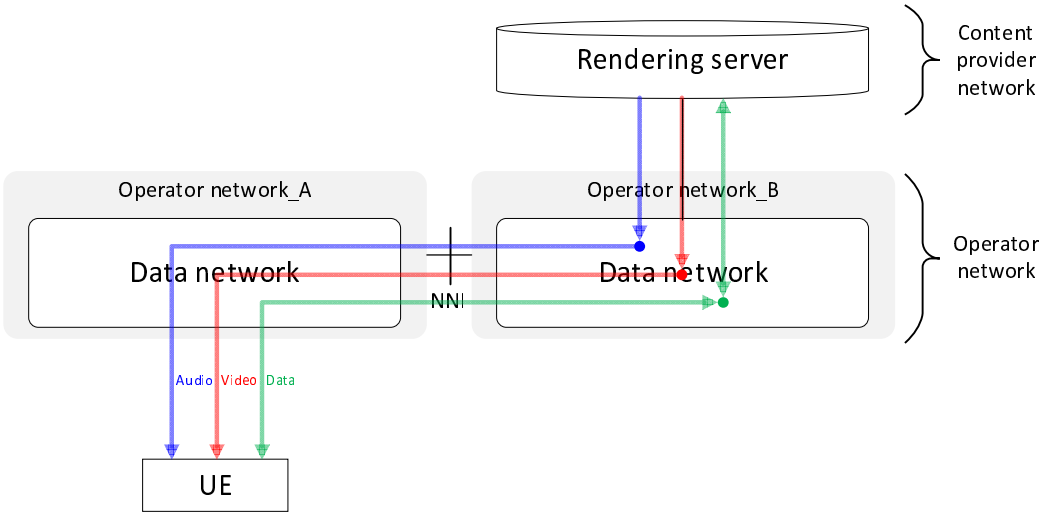


**Figure A.5.2-1: Media/data flows for VR streaming over NNI**

# A.6 Peer-to-peer communication

## A.6.1 General

This use case is an example where multiple users talk each other via the data network. UE_A and UE_B connect to an RTC ID resource (i.e., VR space) according to the service flows described in clause A.2.2.

In this example, UE_A makes a call to UE_B identified by the application specific identity (i.e., RTC user ID), or UE_A makes a call to the IMS-UE identified by the telephone number.

## A.6.2 Service flows

1. UE_A and UE_B establish the media sessions with the data network according to the flows described in clause A.2.2 for entering a VR space.

2. In this VR space, UE_A calls UE_B identified by application specific identity (i.e., RTC user ID) or calls IMS-UE identified by the telephone number and enjoy the voice communication in addition to the VR space (Figure A.6.2-1).

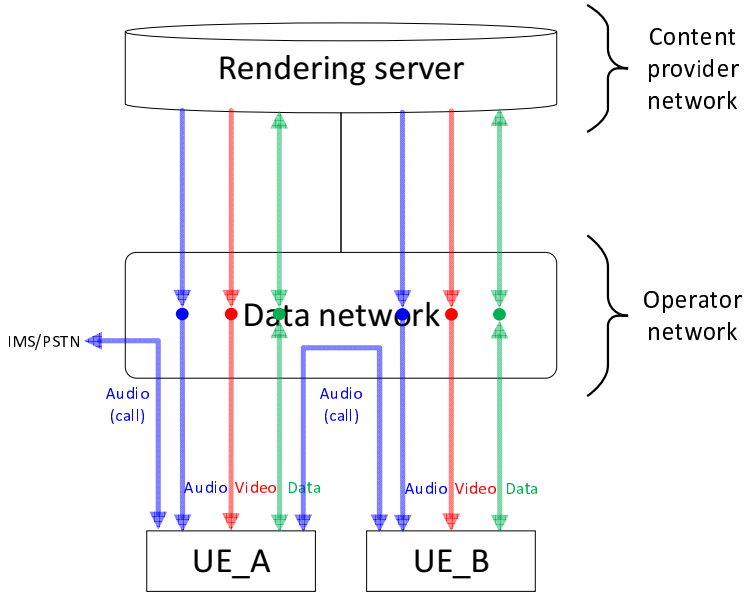


**Figure A.6.2-1: Media/data flows for Peer-to-peer communication**

# Annex B (informative): Message Examples for RESPECT call flow

## B.1 General

This annex provides the message examples for call flow described in clause 6.4.5.6.

- Authentication (clause B.2)
- Media session setup and disconnection for the operator self-contained RTC resource (clause B.3)
- Media session setup and disconnection for the RTC resource provided by other operator (clause B.4)
- Media session setup and disconnection between UEs within a single operator network (clause B.5)
- Media session setup and disconnection between UEs over inter-operator networks (clause B.6)

Parameters used in the example call flows is summarized in table B.1-1.

**Table B.1-1: Parameters used in message examples**

| Parameters | RTC network A | RTC network B |
|---|---|---|
| Domain name | rtc.example.com | rtc.another.com |
| RTC user ID | 3gpp-respect://user1@rtc.example.com (UE1)<br>3gpp-respect://user2@rtc.example.com (UE2) | 3gpp-respect://user2@rtc.another.com (UE2) |
| RTC resource ID | 3gpp-respect://resource1@rtc.example.com | 3gpp-respect://resource2@rtc.another.com |
| U-Plane address | 192.0.2.222 (UE1)<br>192.0.2.234 (UE2)<br>192.0.2.100 (MF)<br>192.0.2.200 (MF2)<br>192.0.2.111 (TGF1 for UE side)<br>192.0.3.111 (TGF1 for external side) | 192.0.100.222 (UE2)<br>192.0.100.200 (MF2)<br>192.0.100.111 (TGF2 for UE side)<br>192.0.3.222 (TGF2 for external side) |

RTC network A: The RTC network where the originating RESPECT endpoint (UE) is connected to.

RTC network B: The RTC network where the RTC network A is connected with.

The RTC network in the call flow does not support the features of "network-asserted identity" and "calling number verification using signature verification and attestation information" yet. However, the WSFs and IWFs in the call flow are within the trust domain for the feature of network asserted identity.

This clause provides message examples for call flow described in clause 6.4.5.6.

## B.2 Authentication

### B.2.1 General

This clause provides message examples for the message flow described in clause 6.4.5.6.2.

### B.2.1 Message examples for authentication

This clause provides message examples for the message flow for authentication described in clause 6.4.5.6.2.

The overall message flow is shown on Figure 6.4.5.6.2-1.

F1. "auth" request (UE1 to WSF)

```
{
  "msgType": "request",
  "method": "auth",
  "transactionId": 0,
  "rtcUserId": "3gpp-respect://user1@rtc.example.com",
  "authType": "Bearer",
  "authorization": "Bearer
   eyAiYWxnIjogIkhTMjU2IiwgInR5cCI6ICJKV1QiIH0K.eyAiaXNzIjogInRlc3QucmQubnR0IiwgInN1YiI6ICJtZXNzYW
   dlIGV4YW1wbGUiLCAiZXhwIjogImRteSIsICJlbWFpbCI6ICJkbXkiLCAiZ3JvdXBzIjogImRteS5kbXkiIH0K.OTEyMWEz
   M2RkN2MxOGZjZjI2NjcxNjQ2MTFiZmFjYjE4YTNhZTY5MmY2YWJkYmZiZGU1ZDQ4YTU5ZjljZGEyZQo="
}
```


F2. "auth" success response (WSF to UE1)


```
{
  "msgType": "response",
  "method": "auth",
  "transactionId": 0,
  "success": true,
  "expires": 3600
}
```


## B.2.2 Message examples for re-authentication

This clause provides message examples for the message flow for authentication described in clause 6.4.5.6.2.

The overall message flow is shown on Figure 6.4.5.6.2-2.

F1. "auth" request (UE1 to WSF)


```
{
  "msgType": "request",
  "method": "auth",
  "transactionId": 100,
  "rtcUserId": "3gpp-respect://user1@rtc.example.com",
  "authType": "Bearer",
  "authorization": "Bearer
   eyAiYWxnIjogIkhTMjU2IiwgInR5cCI6ICJKV1QiIH0K.eyAiaXNzIjogInRlc3QucmQubnR0IiwgInN1YiI6ICJtZXNzYW
   dlIGV4YW1wbGUiLCAiZXhwIjogImRteSIsICJlbWFpbCI6ICJkbXkiLCAiZ3JvdXBzIjogImRteS5kbXkiIH0K.OTEyMWEz
   M2RkN2MxOGZjZjI2NjcxNjQ2MTFiZmFjYjE4YTNhZTY5MmY2YWJkYmZiZGU1ZDQ4YTU5ZjljZGEyZQo="
}
```


F2. "auth" success response (WSF to UE1)


```
{
  "msgType": "response",
  "method": "auth",
  "transactionId": 100,
  "success": true,
  "expires": 3600
}
```


# B.3 Media session setup and disconnection for the operator self-contained RTC resource

This clause provides message examples for the call flow described in clause 6.4.5.6.3. The overall call flow is shown on Figure 6.4.5.6.3-1.

F1. "msetup" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 30,
  "mediaSessionId": "UE1-WSF1-001",
  "dId": {
    "uri": "3gpp-respect://resource1@rtc.example.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387905 3885262146 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706543 192.0.2.222 23456 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
```

```
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F2. "msetup" success response (WSF to UE1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 30,
  "success": true,
  "mediaSessionId": "UE-WSF-001",
  "mediaSessionState": "accepted"
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F3. "mupdate" request (WSF to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 131,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "UE1-WSF1-001",
  "mediaInfo": {
    "type": "offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 5223372036854775808 3885262150 IN IP4 198.0.2.101",
            "s=-",
            "c=IN IP4 198.0.2.100",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Ahk3Zah8",
            "a=ice-pwd:phiegh0M",
            "a=setup:actpass",
          ]
        },
        {
          "index": 1,
          "lines": [
```

```
            "m=audio 23456 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 23456 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 23456 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F4. "mupdate" success response (UE1 to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 131,
  "success": true,
  "updatedKeys": ["mediaInfo"],
  "mediaSessionId": "UE1-WSF1-001",
  "mediaInfo": {
    "type": "answer",
     "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387906 3885262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ... ",
            "a=ice-ufrag:ief0uCCC",
            "a=ice-pwd:ohFeeCCC",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706544 192.0.2.222 23456 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        },
        {
          "index": 2,
```

```
          "actType": "aly"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly"
        }
      ]
    }
  }
}
```

F5. "mupdate" request (WSF to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 133,
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionId": "UE1-WSF1-001",
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "info",
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "actType": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      }
    ]
  }
}
```

F6. "mupdate" success response (UE1 to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 133,
  "success": true,
  "mediaSessionId": "UE1-WSF1-001",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F7. "mdisc" request (UE1 to WSF)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 32,
  "mediaSessionId": "UE-WSF-001",
}
```

F8. "mdisc" success response (WSF to UE1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 32,
  "success": true,
  "mediaSessionId": "UE-WSF-001",
}
```

## B.4 Media session setup and disconnection for the RTC resource provided by other operator

This clause provides message examples for the call flow described in clause 6.4.5.6.4. The overall call flow is shown on Figure 6.4.5.6.4-1.

F1. "msetup" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 40,
  "mediaSessionId": "UE1-WSF1-004",
  "dId": {
    "uri": "3gpp-respect://resource2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387905 3885262146 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706543 192.0.2.222 23456 typ host generation 0"
```

```
          ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F2. "msetup" request (WSF1 to IWF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 440,
  "mediaSessionId": "WSF1-IWF1-004",
  "dId": {
    "uri": "3gpp-respect://resource2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    *** same as F1 ***
  }
}
```

F3. "msetup" request (IWF1 to IWF2)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 4440,
  "mediaSessionId": "IWF1-IWF2-004",
  "dId": {
    "uri": "3gpp-respect://resource2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 1123372036854775808 1185262150 IN IP4 198.0.3.112",
            "s=-",
            "c=IN IP4 198.0.3.111",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Bck3Zah8",
            "a=ice-pwd:hriegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
```

```
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F4. "msetup" request (IWF2 to WSF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 5540,
  "mediaSessionId": "IWF2-WSF2-004",
  "dId": {
    "uri": "3gpp-respect://resource2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 2223372036854775808 2285262150 IN IP4 198.0.100.112",
```

```
            "s=-",
            "c=IN IP4 198.0.100.111",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Dek3Zah8",
            "a=ice-pwd:Jeiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F5. "msetup" success response (WSF2 to IWF2)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 5540,
  "success": true,
  "mediaSessionId": "IWF2-WSF2-004",
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F6. "msetup" success response (IWF2 to IWF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 4440,
  "success": true,
  "mediaSessionId": "IWF1-IWF2-004",
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F7. "msetup" success response (IWF1 to WSF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 440,
  "success": true,
  "mediaSessionId": "WSF1-IWF1-004",
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F8. "msetup" success response (IWF1 to WSF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 40,
  "success": true,
  "mediaSessionId": "UE1-WSF1-004",
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F9. "mupdate" request (WSF2 to IWF2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 15541,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "IWF2-WSF2-004",
  "mediaInfo": {
    "type": "offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 9923372036854775808 9985262150 IN IP4 192.0.100.201",
            "s=-",
            "c=IN IP4 192.0.100.200",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Zyk3Zah8",
            "a=ice-pwd:Yxiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 23456 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 23456 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
```

```
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 23456 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F10. "mupdate" request (IWF2 to IWF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1541,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "IWF1-IWF2-004",
  "mediaInfo": {
    "type": "offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 8823372036854775808 8885262150 IN IP4 192.0.3.223",
            "s=-",
            "c=IN IP4 192.0.3.222",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Yxk3Zah8",
            "a=ice-pwd:Xwiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 23456 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
```

```
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 23456 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 23456 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F11. "mupdate" request (IWF1 to WSF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 141,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "WSF1-IWF1-004",
  "mediaInfo": {
    "type": "offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 7723372036854775808 7785262150 IN IP4 192.0.2.112",
            "s=-",
            "c=IN IP4 192.0.2.111",
```

```
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Yxk3Zah8",
            "a=ice-pwd:Xwiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 23456 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 23456 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 23456 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F12. "mupdate" request (WSF1 to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 41,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "UE1-WSF1-004",
  "mediaInfo": {
    *** same as F11 ***
  }
}
```

F13. "mupdate" success response (UE1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 41,
  "success": true,
  "updatedKeys": ["mediaInfo"],
  "mediaSessionId": "UE1-WSF1-004",
  "mediaInfo": {
    "type": "answer",
     "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4111686018427387906 3185262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ... ",
            "a=ice-ufrag:aef0uCCC",
            "a=ice-pwd:bhFeeCCC",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706544 192.0.2.222 23456 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
```

```
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        },
        {
          "index": 2,
          "actType": "aly"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly"
        }
      ]
    }
  }
}
```

F14. "mupdate" success response (WSF1 to IWF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 141,
  "success": true,
  "updatedKeys": ["mediaInfo"],
  "mediaSessionId": "WSF1-IWF1-004",
  "mediaInfo": {
    *** same as F11 ***
  }
}
```

F15. "mupdate" success response (IWF1 to IWF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1541,
  "success": true,
  "updatedKeys": ["mediaInfo"],
  "mediaSessionId": "IWF1-IWF2-004",
  "mediaInfo": {
    "type": "answer",
     "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 1123372036854775809 1185262151 IN IP4 198.0.3.112",
            "s=-",
            "c=IN IP4 198.0.3.111",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Bck3Zah8",
            "a=ice-pwd:hriegh0M",
            "a=setup:actpass"
           ]
        },
```

```
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        },
        {
          "index": 2,
          "actType": "aly"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly"
        }
      ]
    }
  }
}
```


F16. "mupdate" success response (IWF2 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 15541,
  "success": true,
  "updatedKeys": ["mediaInfo"],
  "mediaSessionId": "IWF2-WSF2-004",
  "mediaInfo": {
    "type": "answer",
     "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387906 3885262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ... ",
            "a=ice-ufrag:ief0uCCC",
            "a=ice-pwd:ohFeeCCC",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        },
        {
          "index": 2,
          "actType": "aly"
```

```
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly"
        }
      ]
    }
  }
}
```

F17. "mupdate" request (WSF2 to IWF2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 15543,
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionId": "WSF2-IWF2-004",
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "info",
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "actType": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      }
    ]
  }
}
```

F18. "mupdate" request (IWF2 to IWF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1543,
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionId": "IWF1-IWF2-004",
  "mediaSessionState": "routed",
  "mediaInfo": {
    *** same as F17 ***
  }
}
```

F19. "mupdate" request (IWF1 to WSF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 143,
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionId": "WSF1-IWF1-004",
  "mediaSessionState": "routed",
  "mediaInfo": {
    *** same as F18 ***
  }
}
```

F20. "mupdate" request (WSF1 to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 43,
  "mediaSessionId": "UE1-WSF1-004",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    *** same as F19 ***
  }
}
```

F21. "mupdate" success response (UE1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 43,
  "success": true,
  "mediaSessionId": "UE1-WSF1-004",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F22. "mupdate" success response (WSF1 to IWF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 143,
  "success": true,
  "mediaSessionId": "WSF1-IWF1-004",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F23. "mupdate" success response (IWF1 to IWF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1543,
  "success": true,
  "mediaSessionId": "IWF1-IWF2-004",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F24. "mupdate" success response (IWF2 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 15543,
  "success": true,
  "mediaSessionId": "IWF2-WSF2-004",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F25. "mdisc" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 42,
  "mediaSessionId": "UE1-WSF1-004"
}
```

F26. "mdisc" request (WSF1 to IWF1)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 442,
  "mediaSessionId": "WSF1-IWF1-004"
}
```

F27. "mdisc" request (IWF1 to IWF2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 4442,
  "mediaSessionId": "IWF1-IWF2-004"
}
```

F28. "mdisc" request (IWF2 to WSF2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 5542,
  "mediaSessionId": "IWF2-WSF2-004"
}
```

F29. "mdisc" success response (WSF2 to IWF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 5542,
  "success": true,
  "mediaSessionId": "WSF2-IWF2-004"
}
```

F30. "mdisc" success response (IWF2 to IWF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 4442,
  "success": true,
  "mediaSessionId": "IWF2-IWF1-004"
}
```

F31. "mdisc" success response (IWF1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 442,
  "success": true,
  "mediaSessionId": "IWF1-WSF1-004"
}
```

F32. "mdisc" success response (WSF1 to UE1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 442,
  "success": true,
  "mediaSessionId": "WSF1-UE1-004"
}
```

## B.5 Media session setup and disconnection between UEs within a single operator network

This clause provides message examples for the call flow described in clause 6.4.5.6.5. The overall call flow is shown on Figure 6.4.5.6.5-1.

F1. "msetup" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 50,
  "mediaSessionId": "UE1-WSF1-005",
  "dId": {
    "uri": 3gpp-respect://user2@rtc.example.com
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387905 3885262146 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
```

```
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706543 192.0.2.222 23456 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F2. "msetup" request (WSF1 to WSF2)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 550,
  "mediaSessionId": "WSF1-WSF2-005",
  "dId": {
    "uri": "3gpp-respect://user2@rtc.example.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    *** same as F1 ***
  }
}
```

F3. "msetup" success response (WSF2 to WSF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 550,
  "success": true,
  "mediaSessionId": WSF1-WSF2-005",
  "mediaSessionState": "accepted"
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
    ]
  }
}
```

F4. "msetup" success response (WSF to UE1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 50,
  "success": true,
  "mediaSessionId": "UE1-WSF1-005,
  "mediaSessionState": "accepted"
  "mediaInfo": {
    *** same as F3 ***
  }
}
```

F5. "msetup" request (WSF2 to UE2)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 1551,
  "mediaSessionId": "WSF2-UE2-005",
  "mediaSessionState": "accepted",
  "dId": {
    "uri": "3gpp-respect://user2@rtc.example.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "Offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 5553372036854775808 5555262150 IN IP4 192.0.2.201",
            "s=-",
            "c=IN IP4 192.0.2.200",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:M253Zah8",
            "a=ice-pwd:M25egh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
```

```
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      },
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F6. "msetup" success response (UE2 to WSF2)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 1551,
  "success": true,
  "mediaSessionId": "WSF2-UE2-005",
}
```

F7. "mupdate" request (WSF2 to WSF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 551,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "WSF1-WSF2-005",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F8. "mupdate" request (WSF1 to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 51,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "UE1-WSF1-005",
  "mediaInfo": {
    *** same as F7 ***
  }
}
```

F9. "mupdate" success response (UE1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 51,
  "success": true,
  "mediaSessionId": "UE1-WSF1-005",
  "updatedKeys": ["mediaInfo"]
}
```

F10. "mupdate" success response (WSF1 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 551,
  "success": true,
  "mediaSessionId": "WSF1-WSF2-005",
  "updatedKeys": ["mediaInfo"]
}
```

F11. "mupdate" request (UE2 to WSF2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1550,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "WSF2-UE2-005",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 5591686018427387906 5595262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2590706544 192.0.2.234 34567 typ host generation 0"
          ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
```

```
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        },
        {
          "index": 2,
          "actType": "aly"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly"
        }
      ]
    }
  }
}
```

F12. "mupdate" success response (WSF2 to UE2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1550,
  "mediaSessionId": "WSF2-UE2-005",
  "updatedKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "connected",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joined"
      }
    ]
  }
}
```

F13. "mupdate" request (WSF2 to WSF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 553,
  "mediaSessionId": "WSF1-WSF2-005",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 9523372036854775808 9585262150 IN IP4 192.0.2.201",
            "s=-",
            "c=IN IP4 192.0.2.200",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:FFk3Zah8",
            "a=ice-pwd:FFiegh0M",
            "a=setup:actpass",
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "actType": "aly",
          "state": {
```

```
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      },
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joined"
      }
    ]
  }
}
```

F14. "mupdate" request (WSF1 to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 53,
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaSessionId": "UE1-WSF1-005",
  "mediaInfo": {
    *** same as F13 ***
  }
}
```

F15. "mupdate" success response (UE1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 53,
  "success": true,
  "mediaSessionId": "UE1-WSF1-005",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F16. "mupdate" success response (WSF1 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 553,
  "success": true,
  "mediaSessionId": "WSF1-WSF2-005",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F17. "mupdate" request (WSF2 to UE2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1553,
  "mediaSessionId": "WSF2-UE2-005",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "info",
    "mc": {
      "metadata": [
        {
          "index": 1,
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      }
    ]
  }
}
```

F18. "mupdate" success response (UE2 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1553,
  "success": true,
  "mediaSessionId": "WSF2-UE2-005",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F19. "mdisc" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 52,
  "mediaSessionId": "UE1-WSF1-005"
}
```

F20. "mdisc" request (WSF1 to WSF2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 552,
  "mediaSessionId": "WSF1-WSF2-005"
}
```

F21. "mdisc" success response (WSF2 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 552,
  "success": true,
  "mediaSessionId": "WSF2-WSF1-005"
}
```

F22. "mdisc" request (WSF2 to UE2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 1555,
  "mediaSessionId": "WSF2-UE2-005"
}
```

F23. "mdisc" success response (WSF1 to UE1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 52,
  "success": true,
  "mediaSessionId": "WSF1-UE1-005"
}
```

F24. "mdisc" success response (UE2 to WSF21)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1555,
  "success": true,
  "mediaSessionId": "WSF2-UE2-005"
}
```

## B.6 Media session setup and disconnection between UEs over inter-operator networks

This clause provides message examples for the call flow described in clause 6.4.5.6.6. The overall call flow is shown on Figure 6.4.5.6.6-1.

F1. "msetup" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 60,
  "mediaSessionId": "UE1-WSF1-006",
  "dId": {
    "uri": 3gpp-respect://user2@rtc.another.com
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387905 3885262146 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706543 192.0.2.222 23456 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
```

```
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F2. "msetup" request (WSF1 to IWF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 660,
  "mediaSessionId": "WSF1-IWF1-006",
  "dId": {
    "uri": "3gpp-respect://user2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    *** same as F1 ***
  }
}
```

F3. "msetup" request (IWF1 to IWF2)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 6660,
  "mediaSessionId": "IWF1-IWF2-006",
  "dId": {
    "uri": "3gpp-respect://user2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 1123372036854775808 1185262150 IN IP4 198.0.3.112",
            "s=-",
            "c=IN IP4 198.0.3.111",
```

```
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Bck3Zah8",
            "a=ice-pwd:hriegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F4. "msetup" request (IWF2 to WSF1)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 16660,
  "mediaSessionId": "IWF2-WSF2-006",
  "dId": {
    "uri": "3gpp-respect://user2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 2223372036854775808 2285262150 IN IP4 198.0.100.112",
            "s=-",
            "c=IN IP4 198.0.100.111",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Dek3Zah8",
            "a=ice-pwd:Jeiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
```

```
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F5. "msetup" success response (WSF2 to IWF2)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 16660,
  "success": true,
  "mediaSessionId": "IWF2-WSF2-006",
  "mediaSessionState": "accepted"
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F6. "msetup" success response (IWF2 to IWF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 6660,
  "success": true,
  "mediaSessionId": "IWF1-IWF2-006",
  "mediaSessionState": "accepted"
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F7. "msetup" success response (IWF1 to WSF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 660,
  "success": true,
  "mediaSessionId": "WSF1-IWF1-006",
  "mediaSessionState": "accepted"
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F8. "msetup" success response (IWF1 to WSF1)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 60,
  "success": true,
  "mediaSessionId": "UE1-WSF1-006",
  "mediaSessionState": "accepted"
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F9. "msetup" request (WSF2 to UE2)

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 1661,
  "mediaSessionId": "WSF2-UE2-006",
  "mediaSessionState": "accepted",
  "dId": {
    "uri": "3gpp-respect://user2@rtc.another.com"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    },
    "network": {
      "uri: "3gpp-respect://user1@rtc.example.com",
    }
  },
  "mediaInfo": {
    "type": "Offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 5553372036854775808 5555262150 IN IP4 198.0.100.201",
            "s=-",
            "c=IN IP4 198.0.100.200",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
```

```
          "a=ice-ufrag:M253Zah8",
          "a=ice-pwd:M25egh0M",
          "a=setup:actpass"
         ]
      },
      {
        "index": 1,
        "lines": [
          "m=audio 9 UDP/TLS/RTP/SAVPF 96",
          "a=mid:0",
          "a=rtcp-mux-only",
          "a=rtcp-mux",
          "a=bundle-only",
          "a=rtcp-rsize",
          "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
          "a=rtpmap:96 opus/48000/2",
          "a=sendonly"
          ]
      },
      {
        "index": 2,
        "lines": [
          "m=video 9 UDP/TLS/RTP/SAVPF 97",
          "a=mid:1",
          "a=rtcp-mux-only",
          "a=rtcp-mux",
          "a=bundle-only",
          "a=rtcp-rsize",
          "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
          "a=rtpmap:97 H264/90000",
          "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
          "a=sendonly"
          ]
      },
      {
        "index": 3,
        "lines": [
          "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
          "a=mid:2",
          "a=bundle-only",
          "a=sctp-port:5000",
          "a=max-message-size:65536",
          "a=dcmap:0"
          ]
      }
    ]
  },
  "mc": {
    "metadata": [
      {
        "index": 1,
        "actType": "add"
      },
      {
        "index": 2,
        "actType": "add"
      }
    ]
  },
  "dc": {
    "sdpIndex": 3,
    "metadata": [
      {
        "id": 0,
        "actType": "add"
      }
    ]
  }
  "participantDesc": [
    {
      "acttype": "add",
      "participantId": "anonymized-RTCuserID-1",
      "userState": "joiningIn"
    },
    {
      "acttype": "add",
      "participantId": "anonymized-RTCuserID-2",
      "userState": "joiningIn"
```

```
      }
    ]
  }
}
```

F10. "msetup" success response (UE2 to WSF2)

```
{
  "msgType": "response",
  "method": "msetup",
  "transactionId": 1661,
  "success": true,
  "mediaSessionId": "WSF2-UE2-006",
}
```

F11. "mupdate" request (WSF2 to IWF2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 26661,
  "mediaSessionId": "WSF1-WSF2-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "add",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joiningIn"
      }
    ]
  }
}
```

F12. "mupdate" request (IWF2 to IWF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 6661,
  "mediaSessionId": "IWF1-IWF2-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "accepted",
  "mediaInfo": {
    *** same as F11 ***
  }
}
```

F13. "mupdate" request (IWF1 to WSF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 661,
  "mediaSessionId": "WSF1-IWF1-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "accepted",
  "mediaInfo": {
    *** same as F14 ***
  }
}
```

F14. "mupdate" request (WSF1 to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 61,
  "mediaSessionId": "UE1-WSF1-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "accepted",
  "mediaInfo": {
    *** same as F15 ***
  }
}
```

F15. "mupdate" success response (UE1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 61,
  "success": true,
  "mediaSessionId": "UE1-WSF1-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F16. "mupdate" success response (WSF1 to IWF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 661,
  "success": true,
  "mediaSessionId": "WSF1-IWF1-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F17. "mupdate" success response (IWF1 to IWF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 6661,
  "success": true,
  "mediaSessionId": "IWF1-IWF2-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F18. "mupdate" success response (IWF2 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 16661,
  "success": true,
  "mediaSessionId": "IWF2-WSF2-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F19. "mupdate" request (UE2 to WSF2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1660,
  "updatingKeys": ["mediaInfo"],
  "mediaSessionId": "WSF2-UE2-006",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
```

```
          "index": 0,
          "lines": [
            "v=0",
            "o=- 5591686018427387906 5595262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2590706544 192.0.2.234 34567 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        },
        {
          "index": 2,
          "actType": "aly"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly"
        }
      ]
    }
  }
}
```

F20. "mupdate" success response (WSF2 to UE2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1660,
  "mediaSessionId": "WSF2-UE2-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "connected",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joined"
      }
    ]
  }
}
```

F21. "mupdate" request (WSF2 to IWF2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 26663,
  "mediaSessionId": "WSF1-WSF2-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 9523372036854775808 9585262150 IN IP4 192.0.100.201",
            "s=-",
            "c=IN IP4 192.0.100.200",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:FFk3Zah8",
            "a=ice-pwd:FFiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
```

```
          "index": 3,
          "lines": [
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      },
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joined"
      }
    ]
  }
}
```


F22. "mupdate" request (IWF2 to IWF1)


```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 6663,
  "mediaSessionId": "IWF1-IWF2-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
```

```
          "o=- 8823372036854775808 8885262150 IN IP4 192.0.3.223",
          "s=-",
          "c=IN IP4 192.0.3.222",
          "t=0 0",
          "a=group:BUNDLE 0 1 2",
          "a=ice-lite",
          "a=fingerprint a=fingerprint sha-256 ...",
          "a=ice-ufrag:Yxk3Zah8",
          "a=ice-pwd:Xwiegh0M",
          "a=setup:actpass"
         ]
      },
      {
        "index": 1,
        "lines": [
          "m=audio 9 UDP/TLS/RTP/SAVPF 96",
          "a=mid:0",
          "a=rtcp-mux",
          "a=rtcp-rsize",
          "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
          "a=rtpmap:96 opus/48000/2",
          "a=sendonly"
          ]
      },
      {
        "index": 2,
        "lines": [
          "m=video 9 UDP/TLS/RTP/SAVPF 97",
          "a=mid:1",
          "a=rtcp-mux",
          "a=rtcp-rsize",
          "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
          "a=rtpmap:97 H264/90000",
          "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
          "a=sendonly"
          ]
      },
      {
        "index": 3,
        "lines": [
          "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
          "a=mid:2",
          "a=sctp-port:5000",
          "a=max-message-size:65536",
          "a=dcmap:0"
          ]
      }
    ]
  },
  "mc": {
    "metadata": [
      {
        "index": 1,
        "actType": "aly",
        "state": {
          "connected": "true",
          "routed": "true"
        }
      },
      {
        "index": 2,
        "actType": "aly",
        "state": {
          "connected": "true",
          "routed": "true"
        }
      }
    ]
  },
  "dc": {
    "sdpIndex": 3,
    "metadata": [
      {
        "id": 0,
        "actType": "aly",
        "state": {
          "connected": "true",
          "routed": "true"
```

```
          }
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      },
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joined"
      }
    ]
  }
}
```

F23. "mupdate" request (IWF1 to WSF1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 663,
  "mediaSessionId": "IWF1-WSF1-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 7723372036854775808 7785262150 IN IP4 192.0.2.112",
            "s=-",
            "c=IN IP4 192.0.2.111",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint a=fingerprint sha-256 ...",
            "a=ice-ufrag:Yxk3Zah8",
            "a=ice-pwd:Xwiegh0M",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendonly"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
```

```
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      },
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-2",
        "userState": "joined"
      }
    ]
  }
}
```

F24. "mupdate" request (WSF1 to UE1)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 63,
  "mediaSessionId": "UE1-WSF1-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    *** same as F23 ***
  }
}
```

F25. "mupdate" success response (UE1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 63,
  "success": true,
  "mediaSessionId": "UE1-WSF1-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F26. "mupdate" success response (WSF1 to IWF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 663,
  "success": true,
  "mediaSessionId": "WSF1-IWF1-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F27. "mupdate" success response (IWF1 to IWF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 6663,
  "success": true,
  "mediaSessionId": "IWF1-IWF2-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F28. "mupdate" success response (IWF2 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 26663,
  "success": true,
  "mediaSessionId": "IWF2-WSF2-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F29. "mupdate" request (WSF2 to UE2)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1663,
  "mediaSessionId": "WSF2-UE2-006",
  "updatingKeys": ["mediaSessionState","mediaInfo"],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "info",
    "mc": {
      "metadata": [
        {
          "index": 1,
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    },
```

```
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "acttype": "mod",
        "participantId": "anonymized-RTCuserID-1",
        "userState": "joined"
      }
    ]
  }
}
```

F30. "mupdate" success response (UE2 to WSF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1663,
  "success": true,
  "mediaSessionId": "WSF2-UE2-006",
  "updatedKeys": ["mediaSessionState","mediaInfo"]
}
```

F31. "mdisc" request (UE1 to WSF1)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 62,
  "mediaSessionId": "UE1-WSF1-006"
}
```

F32. "mdisc" request (WSF1 to IWF1)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 662,
  "mediaSessionId": "WSF1-IWF1-006"
}
```

F33. "mdisc" request (IWF1 to IWF2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 6662,
  "mediaSessionId": "IWF1-IWF2-006"
}
```

F34. "mdisc" request (IWF2 to WSF2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 16662,
  "mediaSessionId": "IWF2-WSF2-006"
}
```

F35. "mdisc" success response (WSF2 to IWF2)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 16662,
  "success": true,
  "mediaSessionId": "WSF2-IWF2-006"
}
```

F36. "mdisc" request (WSF2 to UE2)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 1665,
  "mediaSessionId": "WSF2-UE2-006"
}
```

F37. "mdisc" success response (IWF2 to IWF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 6662,
  "success": true,
  "mediaSessionId": "IWF2-IWF1-006"
}
```

F38. "mdisc" success response (IWF1 to WSF1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 662,
  "success": true,
  "mediaSessionId": "IWF1-WSF1-006"
}
```

F39. "mdisc" success response (WSF1 to UE1)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 62,
  "success": true,
  "mediaSessionId": "UE1-WSF1-006"
}
```

F40. "mdisc" success response (UE2 to WSF21)

```
{
  "msgType": "response",
  "method": "mupdate",
  "transactionId": 1665,
  "success": true,
  "mediaSessionId": "WSF2-UE2-006"
}
```

# Annex C (informative): Call flow examples for RTC-IMS interworking

## C.1 General

In this annex, the call flow examples for RTC-IMS interworking based on the specifications in clause 6.4.5 (for RESPECT endpoint) and clause 6.9.2.3 (for the TGF performing IMS interworking).

- Media session setup from RTC to IMS (clause C.2)
- Media session setup from IMS to RTC (clause C.3)

Parameters used in the example call flows is summarized in table C.1-1.

**Table C.1-1: Parameters used in call flow examples**

| Parameters | | RTC | IMS |
|---|---|---|---|
| SIP domain name | | rtc.example.com | ims.3gpp.org |
| Identity of UE | tel | 815033334444 | 818011112222 |
| | uri | 3gpp-respect://bob@rtc.example.com | n/a |
| Used audio codecs | | OPUS | EVS, AMR-WB, AMR |
| C-Plane address | | 192.0.2.123 (IWF for IMS side) | 192.0.2.234 (IBCF) |
| U-Plane address | | 192.0.2.111 (TGF for IMS side)<br>192.0.3.111 (TGF for RTC side)<br>192.0.3.222 (UE) | 192.0.2.222 (TrGW) |

The RTC network in the call flow does not support the features of "network-asserted identity" and "calling number verification using signature verification and attestation information" yet. However, the IBCF and the IWF in the call flow are within the trust domain for the feature of network asserted identity.

## C.2 Media session setup from RTC to IMS

Figure C.2-1 shows the example call flow of media session setup from RTC network to IMS network.

In this call flow, a UE acting as RESPECT endpoint (UE) has already been connected to a VR space, and this UE initiates a new media session towards the IMS-UE. Through the media session setup, audio media path is established between UE-TGF, TGF-TrGW. The media session is released by the IMS network side.

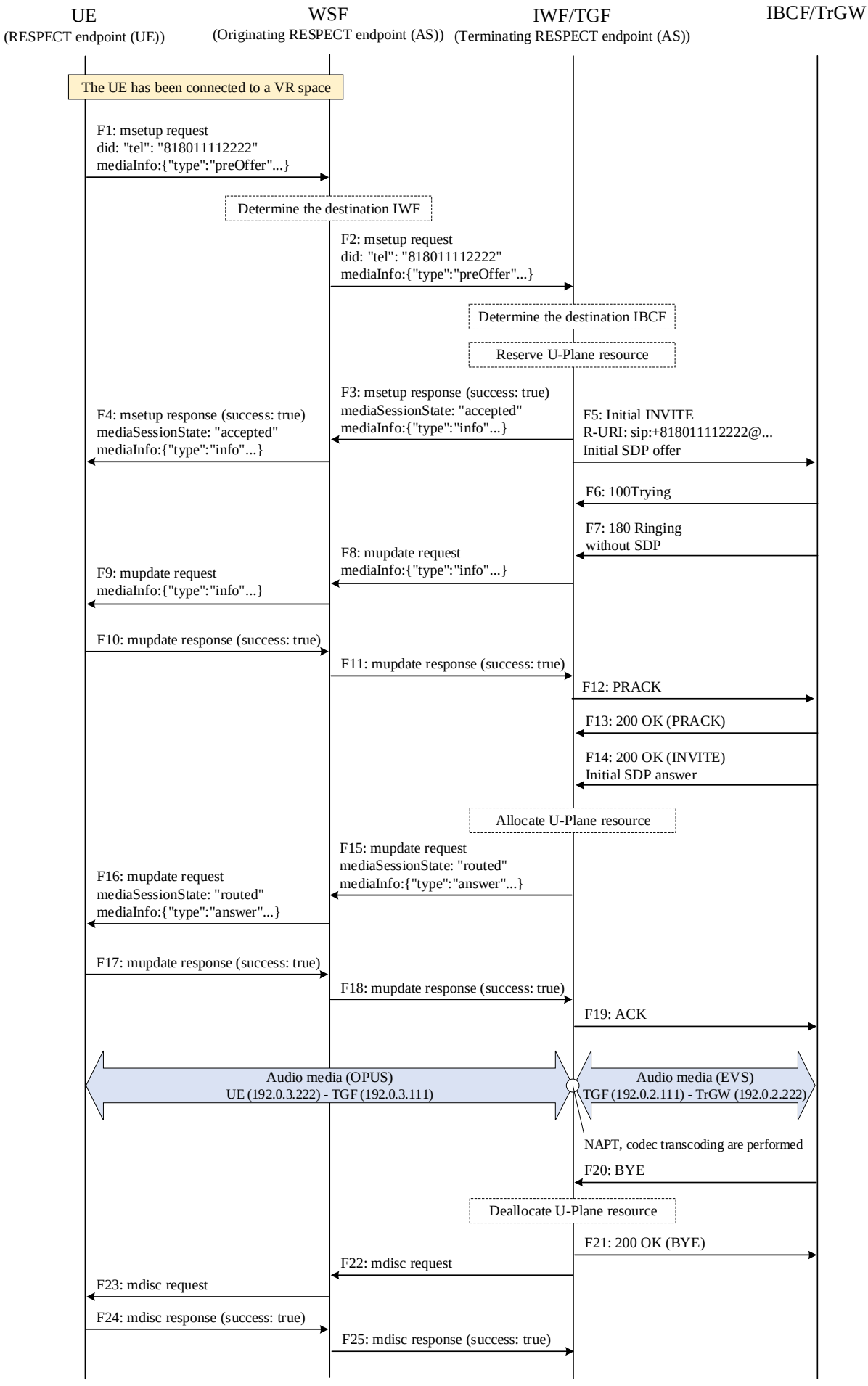
UE
(RESPECT endpoint (UE))
WSF
(Originating RESPECT endpoint (AS))
IWF/TGF
(Terminating RESPECT endpoint (AS))
IBCF/TrGW
The UE has been connected to a VR space
F1: msetup request
did: "tel": "818011112222"
mediaInfo:{"type":"preOffer"...}
Determine the destination IWF
F2: msetup request
did: "tel": "818011112222"
mediaInfo:{"type":"preOffer"...}
Determine the destination IBCF
Reserve U-Plane resource
F3: msetup response (success: true)
mediaSessionState: "accepted"
mediaInfo:{"type":"info"...}
F4: msetup response (success: true)
mediaSessionState: "accepted"
mediaInfo:{"type":"info"...}
F5: Initial INVITE
R-URI: sip:+818011112222@...
Initial SDP offer
F6: 100Trying
F7: 180 Ringing
without SDP
F8: mupdate request
mediaInfo:{"type":"info"...}
F9: mupdate request
mediaInfo:{"type":"info"...}
F10: mupdate response (success: true)
F11: mupdate response (success: true)
F12: PRACK
F13: 200 OK (PRACK)
F14: 200 OK (INVITE)
Initial SDP answer
Allocate U-Plane resource
F15: mupdate request
mediaSessionState: "routed"
mediaInfo:{"type":"answer"...}
F16: mupdate request
mediaSessionState: "routed"
mediaInfo:{"type":"answer"...}
F17: mupdate response (success: true)
F18: mupdate response (success: true)
F19: ACK
Audio media (OPUS)
UE (192.0.3.222) - TGF (192.0.3.111)
Audio media (EVS)
TGF (192.0.2.111) - TrGW (192.0.2.222)
NAPT, codec transcoding are performed
F20: BYE
Deallocate U-Plane resource
F21: 200 OK (BYE)
F22: mdisc request
F23: mdisc request
F24: mdisc response (success: true)
F25: mdisc response (success: true)

**Figure C.2-1: Media session setup from RTC to IMS**

F1: msetup request (UE to WSF)

- A UE acting as RESPECT endpoint (UE) sends an msetup request to the WSF over the existing control session.
- The UE provides its originating identities, and the UE offers bi-directional audio communication, therefore the direction attribute of audio media is set to "sendrecv".

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 30,
  "mediaSessionId": "UE1-WSF1-001",
  "dId": {
    "tn": "818011112222"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
      "tn": "815055556666"
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387905 3885262146 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706543 192.0.3.222 23456 typ host generation 0"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendrecv"
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 9 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:97 H264/90000",
            "a=fmtp:97 profile-level-id=...;sprop-parameter-sets=...",
            "a=sendonly"
            ]
        },
        {
          "index": 3,
          "lines": [
```

```
            "m=application 9 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            "a=bundle-only",
            "a=sctp-port:5000",
            "a=max-message-size:65536",
            "a=dcmap:0"
            ]
        }
      ],
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        },
        {
          "index": 2,
          "actType": "add"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "add"
        }
      ]
    }
  }
}
```

F2: msetup request (WSF to IWF)

- The WSF generates an msetup request based on the received msetup request and sends it to the IWF as a RESPECT client over the existing control session.
- In this case, the WSF does not manage both the media session state and participant state as an originating RESPECT endpoint (AS), since the succeeding entity is RESPECT endpoint (AS).
- Based on the local configuration, the WSF has a knowledge of the fact that the TGF controlled by the destination IWF will provide the U-Plane transport for the UE. Therefore, the WSF does not instruct the MF to reserve the U-Plane resource.
- The media session ID is unique per interface between two RESPECT client - server, then the ID is different from that user in F1 (between UE - WSF).
- The network-asserted identities of the originating UE are included in the signalling message. These identities are retrieved from the AWSF.

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 1000,
  "mediaSessionId": "WSF1-IWF1-10000",
  "dId": {
    "tn": "818011112222"
  },
  "oId": {
    "user": {
      "uri: "3gpp-respect://user1@rtc.example.com",
      "tn": "815055556666"
    },
    "network: {
      "uri: "3gpp-respect://user1@rtc.example.com",
      "tn": "815033334444"
    }
  },
  "mediaInfo": {
    *** same as F1 ***
  }
```

```
}
```

F3: msetup response (IWF to WSF)

- In this case, the IWF manages both the media session state and participant state as a terminating RESPECT endpoint (AS), since the succeeding entity is not RESPECT endpoint (AS).
- The media session state transits to "accepted" in the IWF since the msetup request is received at the terminating RESPECT endpoint (AS) (i.e., IWF). Then this media session state change is notified to the proceeding WSF.
- Also, the IWF updates the participant status in the IWF due to the session setup attempt from the originating RTC user and notifies the status change to the proceeding WSF.

```
{
  "msgType": "response",
  "method": "msetup",
  "success": "true",
  "transactionId": 1000,
  "mediaSessionId": "WSF1-IWF1-10000",
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "actType": "add",
        "participantId": "anonymized-RTCuserID",
        "userState": "JoiningIn"
      }
    ]
  }
}
```

F4: msetup response (WSF to UE)

```
{
  "msgType": "response",
  "method": "msetup",
  "success": "true",
  "transactionId": 30,
  "mediaSessionId": "UE1-WSF1-001",
  "mediaSessionState": "accepted",
  "mediaInfo": {
    *** same as F1 ***
  }
}
```

F5: initial INVITE request (IWF to IBCF)

- The IWF generates an initial INVITE request based on the received msetup request and local configurations.
- The TGF has U-Plane two addresses. One is for RTC network side, and another is for IMS network side. Here, the address for IMS side is set to c= line of the initial SDP offer. Only audio media containing the EVS/AMR/AMR-WB as audio codecs is offered based on local configurations, while the audio media containing OPUS, the video media and datachannel are offered on the RTC side. Therefore, TGF needs to perform NAPT and transcoding for audio media based on the instruction from IWF.
- The originating identities are interworked into the signalling message according to clause 6.9.2.3.

```
INVITE sip:+818011112222@ims.3gpp.org;user=phone SIP/2.0
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK12345678abcdefgh
Max-Forwards: 70
To: <sip:+818011112222@ims.3gpp.org;user=phone>
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 1 INVITE
Contact: <sip:192.0.2.123>
Privacy: none
P-Asserted-Identity: <tel:+815033334444>
P-Charging-Vector: icid-value=1234bc9876e;orig-ioi=rtc.example.com
```

```
Allow: INVITE,ACK,BYE,CANCEL,PRACK,UPDATE
Supported: 100rel,timer
Session-Expires: 300;refresher=uac
Min-SE: 300
Content-Type: application/sdp
Content-Length: 207

v=0
o=- 82664419472 82664419472 IN IP4 192.0.2.111
s=-
c=IN IP4 192.0.2.111
t=0 0
m=audio 10000 RTP/AVP 96 97 98
a=rtpmap:96 EVS/16000/1
a=fmtp:96 br=64; bw=swb; max-red=220
a=rtpmap:97 AMR-WB/16000/1
a=fmtp:97 mode-change-capability=2; max-red=220
a=rtpmap:98 AMR/8000/1
a=fmtp:98 mode-change-capability=2; max-red=220
```

F6: 100 (Trying) response (IBCF to IWF)

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK23456789bcdefghi
To: <sip:+818011112222@ims.3gpp.org;user=phone>
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 1 INVITE
Content-Length: 0
```

F7: 180 (Ringing) response (IBCF to IWF)

```
SIP/2.0 180 Ringing
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK12345678abcdefgh
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 1 INVITE
Contact: <sip:192.0.2.234>
P-Charging-Vector: icid-value=1234bc9876e;orig-ioi=rtc.example.com;term-ioi=ims.3gpp.org
Allow: INVITE,ACK,BYE,CANCEL,PRACK,UPDATE
Require: 100rel
RSeq: 1
Content-Length: 0
```

F8: mupdate request (IWF to WSF)

- The IWF updates the participant status in the IWF due to the reception of 180 (Ringing) response from the IMS network side and notifies the status change to the proceeding WSF.

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1001,
  "mediaSessionId": "WSF1-IWF1-10000",
  "updatingKeys": [ "mediaInfo" ],
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "actType": "add",
        "participantId": "anonymized-IMSueID",
        "userState": "alerting"
      }
    ]
  }
}
```

F9: mupdate request (WSF to UE)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 31,
  "mediaSessionId": "UE1-WSF1-001",
  "mediaSessionState": "accepted",
  "updatingKeys": [ "mediaInfo" ],
  "mediaInfo": {
    *** same as F8 ***
  }
}
```

F10: mupdate response (UE to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 31,
  "mediaSessionId": "UE1-WSF1-001",
  "updatedKeys": [ "mediaInfo" ]
}
```

F11: mupdate response (WSF to IWF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 1001,
  "mediaSessionId": "WSF1-IWF1-10000",
  "updatedKeys": [ "mediaInfo" ]
}
```

F12: PRACK request (IWF to IBCF)

```
PRACK sip:192.0.2.234 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK23456789bcdefghi
Max-Forwards: 70
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 2 PRACK
RAck: 1 1 INVITE
Content-Length: 0
```

F13: 200 (OK) response to PRACK request (IBCF to IWF)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK23456789bcdefghi
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 2 PRACK
Content-Length: 0
```

F14: 200 (OK) response to initial INVITE request (IBCF to IWF)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK12345678abcdefgh
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 1 INVITE
Contact: <sip:192.0.2.234>
```

```
P-Charging-Vector: icid-value=1234bc9876e;orig-ioi=rtc.example.com;term-ioi=ims.3gpp.org
Allow: INVITE,ACK,BYE,CANCEL,PRACK,UPDATE
Require: timer
Session-Expires: 300;refresher=uac
Content-Type: application/sdp
Content-Length: 207

v=0
o=- 82917391739 82917391739 IN IP4 192.0.2.222
s=-
c=IN IP4 192.0.2.222
t=0 0
m=audio 10000 RTP/AVP 96
a=rtpmap:96 EVS/16000/1
a=fmtp:96 br=64; bw=swb; max-red=220
```

F15: mupdate request (IWF to WSF)

- The IWF generates an mupdate request based on the received SIP 200 (OK) response to the initial INVITE request and local configurations.
- The IWF sets the TGF's address for RTC side into c= line of the answer. The video media and datachannel are declined by setting zero port number for corresponding m= line of the answer.
- The media session state transits to "routed" in the IWF, since the U-Plane resource at the TGF for both RTC and IMS sides were allocated ("connected") and needed media routing was setup ("routed"). Then the IWF notifies the proceeding WSF of this media session state change.
- Also, the IWF updates the participant status in the IWF due to the final response indicating the IMS-UE has joined in the media session and notifies the status change to the proceeding WSF. At this timing, the originating RTC user does not join in the media session, but it is expected that the originating RTC user will be joined after the reception of an mupdate request corresponding to this mupdate request, therefore, the IWF notifies the proceeding WSF of "joined" status for the originating RTC user.

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 1003,
  "mediaSessionId": "WSF1-IWF1-10000",
  "updatingKeys": [ "mediaSessionState", "mediaInfo" ],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 8223372036854775808 3885262147 IN IP4 192.0.3.112",
            "s=-",
            "c=IN IP4 192,0.3.111",
            "t=0 0",
            "a=group:BUNDLE 0 1 2",
            "a=ice-lite",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:Anoo6wiu",
            "a=ice-pwd:Wienu1Io",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 12345 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendrecv"
```

```
            ]
        },
        {
          "index": 2,
          "lines": [
            "m=video 0 UDP/TLS/RTP/SAVPF 97",
            "a=mid:1"
            ]
        },
        {
          "index": 3,
          "lines": [
            "m=application 0 UDP/DTLS/SCTP webrtc-datachannel",
            "a=mid:2",
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        },
        {
          "index": 2,
          "actType": "dcl"
        }
      ]
    },
    "dc": {
      "sdpIndex": 3,
      "metadata": [
        {
          "id": 0,
          "actType": "dcl"
        }
      ]
    }
    "participantDesc": [
      {
        "actType": "mod",
        "participantId": "anonymized-IMSueID",
        "userState": "joined"
      }
      {
        "actType": "mod",
        "participantId": "anonymized-RTCuserID",
        "userState": "joined"
      }
    ]
  }
}
```

F16: mupdate request (WSF to UE)

- Upon receiving the mupdate request containing the answer with the "routed" media session state, the UE can start the audio communication with IMS-UE.

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 33,
  "mediaSessionId": "UE1-WSF1-001",
  "updatingKeys": [ "mediaSessionState", "mediaInfo" ],
  "mediaSessionState": "routed",
  "mediaInfo": {
    *** same as F15 ***
  }
}
```

F17: mupdate response (UE to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 33,
  "mediaSessionId": "UE1-WSF1-001",
  "updatedKeys": [ "mediaSessionState", "mediaInfo" ]
}
```

F18: mupdate response (WSF to IWF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 1003,
  "mediaSessionId": "WSF1-IWF1-10000",
  "updatedKeys": [ "mediaSessionState", "mediaInfo" ]
}
```

F19: ACK request (IWF to IBCF)

```
ACK sip:192.0.2.234 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK34567890cdefghij
Max-Forwards: 70
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
From: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 1 ACK
Content-Length: 0
```

F20: BYE request (IBCF to IWF)

```
BYE sip:192.0.2.134 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK56789012efghijkl
Max-Forwards: 70
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
To: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 3 BYE
Content-Length: 0
```

F21: 200 (OK) response to BYE request (IWF to IBCF)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK56789012efghijkl
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=9876zyxw
To: <sip:+815055556666@rtc.example.com;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 3 BYE
Content-Length: 0
```

F22: mdisc request (IWF to WSF)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 1005,
  "mediaSessionId": "WSF1-IWF1-10000"
}
```

F23: mdisc request (WSF to UE)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 35,
  "mediaSessionId": "UE1-WSF1-001"
}
```

F24: mdisc response (UE to WSF)

```
{
  "msgType": "response",
  "method": "mdisc",
  "success": "true",
  "transactionId": 35,
  "mediaSessionId": "UE1-WSF1-001"
}
```

F25: mdisc response (WSF to IWF)

```
{
  "msgType": "response",
  "method": "mdisc",
  "success": "true",
  "transactionId": 1005,
  "mediaSessionId": "WSF1-IWF1-10000"
}
```

## C.3 Media session setup from IMS to RTC

Figure C.3-1 shows the example call flow of media session setup from IMS network to RTC network.

In this call flow, a UE acting as RESPECT endpoint (UE) has already been connected to a VR space, and an IMS-UE initiates a new media session towards this UE acting as RESPECT endpoint (UE). Through the media session setup, audio media path is established between UE-TGF, TGF-TrGW. The media session is released by the RTC network side.

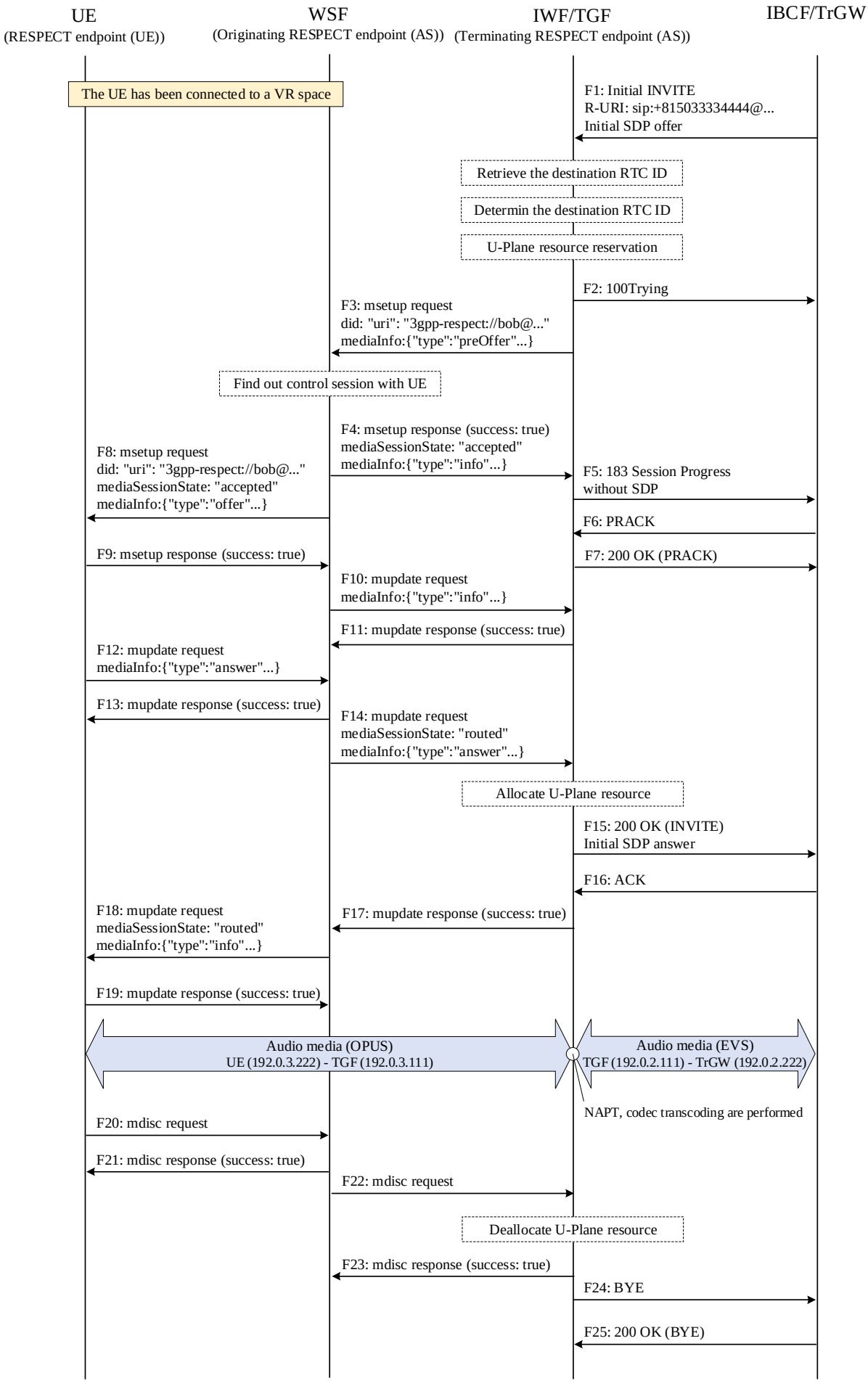
UE
(RESPECT endpoint (UE))
WSF
(Originating RESPECT endpoint (AS))
IWF/TGF
(Terminating RESPECT endpoint (AS))
IBCF/TrGW
The UE has been connected to a VR space
F1: Initial INVITE
R-URI: sip:+815033334444@...
Initial SDP offer
Retrieve the destination RTC ID
Determin the destination RTC ID
U-Plane resource reservation
F2: 100Trying
F3: msetup request
did: "uri": "3gpp-respect://bob@..."
mediaInfo:{"type":"preOffer"...}
Find out control session with UE
F4: msetup response (success: true)
mediaSessionState: "accepted"
mediaInfo:{"type":"info"...}
F8: msetup request
did: "uri": "3gpp-respect://bob@..."
mediaSessionState: "accepted"
mediaInfo:{"type":"offer"...}
F5: 183 Session Progress
without SDP
F6: PRACK
F9: msetup response (success: true)
F7: 200 OK (PRACK)
F10: mupdate request
mediaInfo:{"type":"info"...}
F11: mupdate response (success: true)
F12: mupdate request
mediaInfo:{"type":"answer"...}
F13: mupdate response (success: true)
F14: mupdate request
mediaSessionState: "routed"
mediaInfo:{"type":"answer"...}
Allocate U-Plane resource
F15: 200 OK (INVITE)
Initial SDP answer
F16: ACK
F18: mupdate request
mediaSessionState: "routed"
mediaInfo:{"type":"info"...}
F17: mupdate response (success: true)
F19: mupdate response (success: true)
Audio media (OPUS)
UE (192.0.3.222) - TGF (192.0.3.111)
Audio media (EVS)
TGF (192.0.2.111) - TrGW (192.0.2.222)
NAPT, codec transcoding are performed
F20: mdisc request
F21: mdisc response (success: true)
F22: mdisc request
Deallocate U-Plane resource
F23: mdisc response (success: true)
F24: BYE
F25: 200 OK (BYE)

**Figure C.3-1: Media session setup from IMS to RTC**

F1: initial INVITE request (IBCF to IWF)

- An IMS-UE in the IMS network initiates a call toward the RTC user by specifying its telephone-number into the Request-URI of the initial INVITE request.

```
INVITE sip:+815033334444@rtc.example.com;user=phone SIP/2.0
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgh
Max-Forwards: 70
To: <sip:+815033334444@rtc.example.com;user=phone>
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 1 INVITE
Contact: <sip:192.0.2.234>
Privacy: none
P-Asserted-Identity: <tel:+818011112222>
P-Asserted-Identity: <sip:+818011112222@ims.3gpp.org;user=phone>
P-Charging-Vector: icid-value=1234bc9876e;orig-ioi=ims.3gpp.org
Allow: INVITE,ACK,BYE,CANCEL,PRACK,UPDATE
Supported: 100rel,timer
Session-Expires: 300;refresher=uac
Min-SE: 300
Content-Type: application/sdp
Content-Length: 207

v=0
o=- 82917391739 82917391739 IN IP4 192.0.2.222
s=-
c=IN IP4 192.0.2.222
t=0 0
m=audio 20000 RTP/AVP 96 97 98
a=rtpmap:96 EVS/16000/1
a=fmtp:96 br=64; bw=swb; max-red=220
a=rtpmap:97 AMR-WB/16000/1
a=fmtp:97 mode-change-capability=2; max-red=220
a=rtpmap:98 AMR/8000/1
a=fmtp:98 mode-change-capability=2; max-red=220
```

F2: 100 (Trying) response (IWF to IBCF)

```
SIP/2.0 100 Trying
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgh
To: <sip:+815033334444@rtc.example.com;user=phone>
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 1 INVITE
Content-Length: 0
```

F3: msetup request (IWF to WSF)

- The IWF generates an msetup request based on the received initial INVITE request and local configurations. Then, the IWF sends it to the WSF as a RESPECT client over the existing control session.

- The IWF specify the RTC user ID retrieved from the ASWF by querying the telephone-number in the Request-URI of the received initial INVITE request.

- In this case, the IWF does not manage both the media session state and participant state as an originating RESPECT endpoint (AS), since the succeeding entity is RESPECT endpoint (AS).

- The network-asserted identities of the originating IMS-UE are interworked into the signalling message according to clause 6.9.2.3.

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 2000,
  "mediaSessionId": "IWF1-WSF1-20000",
  "dId": {
    "uri": "3gpp-respect://bob@rtc.example.com"
```

```
  },
  "oId": {
    "user": {
      "tn": "818011112222"
    },
    "network: {
      "tn": "818011112222"
    }
  },
  "mediaInfo": {
    "type": "preOffer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 8223372036854775809 3885262148 IN IP4 192.0.3.112",
            "s=-",
            "c=IN IP4 192,0.3.111",
            "t=0 0",
            "a=group:BUNDLE 0",
            "a=ice-lite",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:Anoo6wiu",
            "a=ice-pwd:Wienu1Io",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 56789 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendrecv"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        }
      ]
    }
  }
}
```

F4: msetup response (WSF to IWF)

- In this case, the WSF manages both the media session state and participant state as a terminating RESPECT endpoint (AS), since the succeeding entity is not RESPECT endpoint (AS). Also, the WSF will respond to the request before sending the request for both sides.

- The media session state transits to "accepted" in the WSF since the msetup request is received at the terminating RESPECT endpoint (AS) (i.e., WSF). Then this media session state change is notified to the proceeding IWF.

- Also, the WSF updates the participant status in the WSF due to the session setup attempt from the IMS network side and notifies the status change to the proceeding IWF.

```
{
  "msgType": "response",
  "method": "msetup",
  "success": "true",
  "transactionId": 2000,
  "mediaSessionId": "IWF1-WSF1-20000",
```

```
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "actType": "add",
        "participantId": "anonymized-IMSueID",
        "userState": "JoiningIn"
      }
    ]
  }
}
```

F5: 183 (Session Progress) response (IWF to IBCF)

```
SIP/2.0 183 Session Progress
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgh
To: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 1 INVITE
Contact: <sip:192.0.2.123>
P-Charging-Vector: icid-value=1234bc9876e;orig-ioi=ims.3gpp.org;term-ioi=rtc.example.com
Allow: INVITE,ACK,BYE,CANCEL,PRACK,UPDATE
Require: 100rel
RSeq: 1
Content-Length: 0
```

F6: PRACK request (IBCF to IWF)

```
PRACK sip:192.0.2.234 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgi
Max-Forwards: 70
To: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 2 PRACK
RAck: 1 1 INVITE
Content-Length: 0
```

F7: 200 (OK) response to PRACK request (IWF to IBCF)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgi
To: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.123
CSeq: 2 PRACK
Content-Length: 0
```

F8: msetup request (WSF to UE)

- The WSF handles the received "preOffer" as "offer" because the proceeding entity is the RESPECT endpoint (UE), and includes it into the msetup request.
- The WSF updates the participant status of the terminating RTC user since it is expected that the terminating RTC user will be alerted after the reception of this msetup request, therefore, the WSF notifies the UE of "alerting" status for the terminating RTC user in addition to the participant status of the originating IMS-UE.

```
{
  "msgType": "request",
  "method": "msetup",
  "transactionId": 131,
  "mediaSessionId": "WSF1-UE1-001",
  "dId": {
    "uri": "3gpp-respect://bob@rtc.example.com"
```

```
  },
  "oId": {
    "user": {
      "tn": "818011112222"
    },
    "network: {
      "tn": "818011112222"
    }
  },
  "mediaSessionState": "accepted",
  "mediaInfo": {
    "type": "offer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 8223372036854775809 3885262148 IN IP4 192.0.3.112",
            "s=-",
            "c=IN IP4 192,0.3.111",
            "t=0 0",
            "a=group:BUNDLE 0",
            "a=ice-lite",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:Anoo6wiu",
            "a=ice-pwd:Wienu1Io",
            "a=setup:actpass"
           ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 56789 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendrecv"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "add"
        }
      ]
    }
    "participantDesc": [
      {
        "actType": "add",
        "participantId": "anonymized-IMSueID",
        "userState": "JoiningIn"
      }
      {
        "actType": "add",
        "participantId": "anonymized-RTCuserID",
        "userState": "alerting"
      }
    ]
  }
}
```

F9: msetup response (UE to WSF)

```
{
  "msgType": "response",
  "method": "msetup",
  "success": "true",
  "transactionId": 131,
```

```
  "mediaSessionId": "WSF1-UE1-001"
}
```

F10: mupdate request (WSF to IWF)

- The WSF notifies the succeeding IWF of "alerting" status for the terminating RTC user as with F11.

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 2001,
  "mediaSessionId": "IWF1-WSF1-20000",
  "updatingKeys": [ "mediaInfo" ],
  "mediaInfo": {
    "type": "info",
    "participantDesc": [
      {
        "actType": "add",
        "participantId": "anonymized-RTCuserID",
        "userState": "alerting"
      }
    ]
  }
}
```

F11: mupdate response (IWF to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 2001,
  "mediaSessionId": "IWF1-WSF1-20000",
  "updatiedKeys": [ "mediaInfo" ]
}
```

F12: mupdate request (UE to WSF)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 130,
  "mediaSessionId": "WSF1-UE1-001",
  "updatingKeys": [ "mediaInfo" ],
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387906 3885262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706544 192.0.3.222 34567 typ host generation 0"
          ]
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
```

```
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendrecv"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly"
        }
      ]
    }
  }
}
```

F13: mupdate response (WSF to UE)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 130,
  "mediaSessionId": "WSF1-UE1-001",
  "updatedKeys": [ "mediaInfo" ]
}
```

F14: mupdate request (WSF to IWF)

- The WSF generates an mupdate request based on the received mupdate request.
- The media session state transits to "routed" in the WSF, since it is expected that U-Plane resource at the TGF for both RTC and IMS sides are allocated ("connected") and needed media routing is setup ("routed") after the reception of this mupdate request. Then the WSF notifies the proceeding IWF of this media session state change.
- Also, the WSF updates the participant status in the WSF due to the mupdate request containing the answer and notifies the status change to the proceeding IWF. At this timing, the originating IMS-UE does not join in the media session, but it is expected that the originating IMS-UE will be joined after the reception of a corresponding to this mupdate request, therefore, the IWF notifies the proceeding WSF of "joined" status for the originating IMS-UE.

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 2003,
  "mediaSessionId": "IWF1-WSF1-20000",
  "updatingKeys": [ "mediaSessionState", "mediaInfo" ],
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "answer",
    "sdp": {
      "part": [
        {
          "index": 0,
          "lines": [
            "v=0",
            "o=- 4611686018427387906 3885262147 IN IP4 0 0 0 0",
            "s=-",
            "c=IN IP4 0.0.0.0",
            "t=0 0",
            "a=group:BUNDLE 0",
            "a=ice-options:trickle",
            "a=fingerprint sha-256 ...",
            "a=ice-ufrag:ief0uBai",
            "a=ice-pwd:ohFee4ne",
            "a=setup:actpass",
            "a=candidate 1 1 UDP 2130706544 192.0.3.222 34567 typ host generation 0"
          ]
```

```
        },
        {
          "index": 1,
          "lines": [
            "m=audio 9 UDP/TLS/RTP/SAVPF 96",
            "a=mid:0",
            "a=rtcp-mux-only",
            "a=rtcp-mux",
            "a=bundle-only",
            "a=rtcp-rsize",
            "a=extmap:1 urn:ietf:params:rtp-hdrext:sdes:mid",
            "a=rtpmap:96 opus/48000/2",
            "a=sendrecv"
            ]
        }
      ]
    },
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "aly",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
    "participantDesc": [
      {
        "actType": "mod",
        "participantId": "anonymized-IMSueID",
        "userState": "joined"
      }
      {
        "actType": "mod",
        "participantId": "anonymized-RTCuserID",
        "userState": "joined"
      }
    ]
  }
}
```

F15: 200 (OK) response to initial INVITE request (IWF to IBCF)

- The IWF generates a SIP 200 (OK) response to the initial INVITE request based on the received mupdate request and local configurations.
- The IWF sets the TGF's address for IMS side into c= line of the answer and selects EVS as audio codec.
- Upon receiving the SIP 200 (OK) response to the initial INVITE request, the IMS-UE can start the audio communication with UE in the RTC network.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgh
To: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 1 INVITE
Contact: <sip:192.0.2.123>
P-Charging-Vector: icid-value=1234bc9876e;orig-ioi=ims.3gpp.org;term-ioi=rtc.example.com
Allow: INVITE,ACK,BYE,CANCEL,PRACK,UPDATE
Require: timer
Session-Expires: 300;refresher=uac
Content-Type: application/sdp
Content-Length: 207

v=0
o=- 82917391739 82917391739 IN IP4 192.0.2. 111
s=-
c=IN IP4 192.0.2.111
t=0 0
m=audio 20000 RTP/AVP 96
a=rtpmap:96 EVS/16000/1
```

```
a=fmtp:96 br=64; bw=swb; max-red=220
```

F16: ACK request (IBCF to IWF)

```
ACK sip:192.0.2.123 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.234:5060;branch=z9hG4bK12345678abcdefgh
Max-Forwards: 70
To: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
From: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 1 ACK
Content-Length: 0
```

F17: mupdate response (IWF to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 2003,
  "mediaSessionId": "IWF1-WSF1-20000",
  "updatedKeys": [ "mediaSessionState", "mediaInfo" ]
}
```

F18: mupdate request (WSF to UE)

```
{
  "msgType": "request",
  "method": "mupdate",
  "transactionId": 133,
  "mediaSessionId": "WSF1-UE1-001",
  "updatingKeys": [ "mediaSessionState", "mediaInfo" ]
  "mediaSessionState": "routed",
  "mediaInfo": {
    "type": "info",
    "mc": {
      "metadata": [
        {
          "index": 1,
          "actType": "mod",
          "state": {
            "connected": "true",
            "routed": "true"
          }
        }
      ]
    }
  }
}
```

F19: mupdate response (UE to WSF)

```
{
  "msgType": "response",
  "method": "mupdate",
  "success": "true",
  "transactionId": 133,
  "mediaSessionId": "WSF1-UE1-001",
  "updatedKeys": [ "mediaSessionState", "mediaInfo" ]
}
```

F20: mdisc request (UE to WSF)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 132,
```

```
  "mediaSessionId": "WSF1-UE1-001"
}
```

F21: mdisc response (WSF to UE)

```
{
  "msgType": "response",
  "method": "mdisc",
  "success": "true",
  "transactionId": 132,
  "mediaSessionId": "WSF1-UE1-001"
}
```

F22: mdisc request (WSF to IWF)

```
{
  "msgType": "request",
  "method": "mdisc",
  "transactionId": 2005,
  "mediaSessionId": "IWF1-WSF1-20000"
}
```

F23: mdisc response (IWF to WSF)

```
{
  "msgType": "response",
  "method": "mdisc",
  "success": "true",
  "transactionId": 2005,
  "mediaSessionId": "IWF1-WSF1-20000"
}
```

F24: BYE request (IWF to IBCF)

```
BYE sip:192.0.2.234 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.123: 5060;branch=z9hG4bK12345678abcdefgj
Max-Forwards: 70
From: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 3 BYE
Content-Length: 0
```

F25: 200 (OK) response to BYE request (IBCF to IWF)

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.123:5060;branch=z9hG4bK12345678abcdefgj
From: <sip:+815033334444@rtc.example.com;user=phone>;tag=9876zyxw
To: <sip:+818011112222@ims.3gpp.org;user=phone>;tag=1234abcd
Call-ID: qwertyuiop123456@192.0.2.234
CSeq: 3 BYE
Content-Length: 0
```

# Annex D (informative): JSON data format for RESPECT

This Annex provides the JSON data format for RESPECT.

## D.1 Information elements for each message

This clause defines the information elements included in the signalling message.

### D.1.1 Authentication method

This clause describes the JSON format for Authentication method.

#### D.1.1.1 auth request

This clause describes the JSON format for "auth" request in Table D.1.1.1-1.

**Table D.1.1.1-1: JSON format of auth request**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "auth" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| rtcUserId | string | M | 1 | The value is set to RTC user ID which is required to be authenticated, according to clause 6.4.5.5.4.3.9. | |
| authType | AuthType | M | 1 | The value is set to the type of authentication, according to clause 6.4.5.5.4.3.10. | |
| authorization | string | O | 0..1 | The value is set to the credential token for authentication, according to clause 6.4.5.5.4.3.11. | |
| disconnectTtl | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.12. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

#### D.1.1.2 auth response

This clause describes the JSON format for "auth" response Table D.1.1.2-1.

**Table D.1.1.2-1: JSON format of auth response**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "response" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "auth" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| success | boolean | M | 1 | The value is set according to clause 6.4.5.5.4.3.2. | |
| problemDetails | ProblemDetails | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.3. | |
| retryAfetr | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.7. | |
| unsupportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.5. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4 | |
| expires | number (int32) | M | 1 | The value is set according to clause 6.4.5.5.4.3.15. | |
| disconnectTtl | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.12. | |
| webrtcReauthCredential | string | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.13. | |
| wwwAuthenticate | WwwAuthenticate | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.14. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.2 Media session setup method

This clause describes the JSON format for media session setup method.

### D.1.2.1 msetup request

This clause describes the JSON format for "msetup" request in Table D.1.2.1-1.

**Table D.1.2.1-1: JSON format of msetup request**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| dId | object | M | 1 | The value is set to RTC user ID or RTC resource ID, according to clause 6.4.5.5.4.3.9. | |
| mediaSessionId | string | M | 1 | The value is set according to clause 6.4.5.5.4.3.17. | |
| mediaSessionState | MediaSessionState | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.18. | |
| mediaInfo | MediaInfo | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.19. | |
| oId | OrigId | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.20. | |
| cId | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.24. | |
| userData | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.25. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.2.2 msetup response

This clause describes the JSON format for "msetup" response in Table D.1.2.2-1.

**Table D.1.2.2-1: JSON format of msetup response**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| success | boolean | M | 1 | The value is set according to clause 6.4.5.5.4.3.2. | |
| problemDetails | ProblemDetails | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.3. | |
| retryAfetr | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.7. | |
| unsupportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.5. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4 | |
| mediaSessionId | string | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.17. | |
| mediaSessionState | MediaSessionState | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.18. | |
| mediaInfo | MediaInfo | O | 0..1 | The value is set to the credential token for authentication, according to clause 6.4.5.5.4.3.19. | |
| oId | OrigId | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.20. | |
| userData | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.25. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.3 Media session update method

This clause describes the JSON format for media session update method.

### D.1.3.1 mupdate request

This clause describes the JSON format for "mupdate" request in Table D.1.3.1-1.

**Table D.1.3.1-1: JSON format of mupdate request**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| dId | object | O | 0..1 | The value is set to RTC user ID or RTC resource ID, according to clause 6.4.5.5.4.3.9. | |
| mediaSessionId | string | M | 1 | The value is set according to clause 6.4.5.5.4.3.17. | |
| mediaSessionState | MediaSessionState | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.18. | |
| updatingKeys | array[string] | M | 0..1 | The value is set according to clause 6.4.5.5.4.3.22. | |
| mediaInfo | MediaInfo | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.19. | |
| oId | OrigId | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.20. | |
| userData | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.25. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.3.2 mupdate response

This clause describes the JSON format for "mupdate" response in Table D.1.3.2-1.

**Table D.1.3.2-1: JSON format of mupdate response**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| success | boolean | M | 1 | The value is set according to clause 6.4.5.5.4.3.2. | |
| problemDetails | ProblemDetails | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.3. | |
| retryAfetr | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.7. | |
| unsupportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.5. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| mediaSessionId | string | M | 1 | The value is set according to clause 6.4.5.5.4.3.17. | |
| mediaSessionState | MediaSessionState | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.18. | |
| updatedKeys | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.23. | |
| mediaInfo | MediaInfo | O | 0..1 | The value is set to the credential token for authentication, according to clause 6.4.5.5.4.3.19. | |
| oId | OrigId | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.20. | |
| userData | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.25. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.4 Media session disconnection method

This clause describes the JSON format for media session disconnection method.

### D.1.4.1 mdisc request

This clause describes the JSON format for "mdisc" request in Table D.1.4.1-1.

**Table D.1.4.1-1: JSON format of mdisc request**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| mediaSessionId | string | M | 1 | The value is set according to clause 6.4.5.5.4.3.17. | |
| problemDetails | ProblemDetails | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.3. | |
| userData | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.25. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.4.2 mdisc response

This clause describes the JSON format for "mdisc" response in Table D.1.4.2-1.

**Table D.1.4.2-1: JSON format of mdisc response**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| success | boolean | M | 1 | The value is set according to clause 6.4.5.5.4.3.2. | |
| problemDetails | ProblemDetails | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.3. | |
| retryAfetr | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.7. | |
| unsupportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.5. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| mediaSessionId | string | M | 1 | The value is set according to clause 6.4.5.5.4.3.17. | |
| userData | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.25. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

# D.1.5 Get information method

This clause describes the JSON format for get information method.

## D.1.5.1 getinfo request

This clause describes the JSON format for "getinfo" request in Table D.1.5.1-1.

**Table D.1.5.1-1: JSON format of getinfo request**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| resourceReq | array[string] | M | 1 | The value is set according to clause 6.4.5.5.4.3.21. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

## D.1.5.2 getinfo response

This clause describes the JSON format for "getinfo" response in Table D.1.5.2-1.

**Table D.1.5.2-1: JSON format of getinfo response**

| IE name | Data type | P | Cardinality | Description | Applicability (NOTE) |
|---|---|---|---|---|---|
| msgType | MsgType | M | 1 | The value is set to "request" according to clause 6.4.5.5.4.2.2. | |
| method | Method | M | 1 | The value is set to "msetup" according to clause 6.4.5.5.4.2.3. | |
| transactionId | number (int64) | M | 1 | The value is generated according to clause 6.4.5.5.4.2.4. | |
| success | boolean | M | 1 | The value is set according to clause 6.4.5.5.4.3.2. | |
| problemDetails | ProblemDetails | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.3. | |
| retryAfetr | number (int32) | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.7. | |
| unsupportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.5. | |
| supportedExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.6. | |
| requireExtension | array[string] | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.4. | |
| resourceRes | object | O | 0..1 | The value is set according to clause 6.4.5.5.4.3.21. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | | | | |

# D.2 Structured data types

This clause describes structured data types for RESPECT.

## D.2.1 Type: ProblemDetails

**Table D.2.1-1: Definition of ProblemDetails data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| type | string | 1 | This IE includes a URI reference as specified in IETF RFC 7807 [32A]. The value is set according to clause 6.4.5.5.5.1. |
| title | string | 0..1 | This IE includes a short, human-readable summary of the problem type as specified in IETF RFC 7807 [32A] |
| status | number | 0..1 | This IE includes the HTTP status code as specified in IETF RFC 7807 [32A]. |
| detail | string | 0..1 | This IE includes a human-readable explanation specific to this occurrence of the problem as specified in IETF RFC 7807 [32A]. |
| instance | string | 0..1 | This IE includes a URI reference that identifies the specific occurrence of the problem as specified in IETF RFC 7807 [32A]. |

## D.2.2 Type: WwwAuthenticate

**Table D.2.2-1: Definition of WwwAuthenticate data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| authScheme | string | 1 | See IETF RFC 9110 [54]. |
| realm | string | 0..1 | See IETF RFC 9110 [54]. |
| domain | string | 0..1 | See IETF RFC 9110 [54]. |
| nonce | string | 0..1 | See IETF RFC 9110 [54]. |
| uri | string | 0..1 | See IETF RFC 9110 [54]. |
| qop | string | 0..1 | See IETF RFC 9110 [54]. |
| nc | string | 0..1 | See IETF RFC 9110 [54]. |
| cnonce | string | 0..1 | See IETF RFC 9110 [54]. |
| response | string | 0..1 | See IETF RFC 9110 [54]. |
| opaque | string | 0..1 | See IETF RFC 9110 [54]. |
| stale | string | 0..1 | See IETF RFC 9110 [54]. |
| algorithm | string | 0..1 | See IETF RFC 9110 [54]. |

## D.2.3 Type: MediaInfo

This type is required to comply with the provisions defined in table D.2.3-1.

**Table D.2.3-1: Definition of MediaInfo data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| type | oaType | 1 | Refer to clause 6.4.5.5.4.3.19.<br>This information element indicates the type of mediaInfo element. |
| sdp | Sdp | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| mc | mediaChannel | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| dc | dataChannel | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| participantDesc | array [participantDescElem] | 0..1 | Refer to clause 6.4.5.5.4.3.19. |

### D.2.3.1 Type: Sdp

This type is required to comply with the provisions defined in table D.2.3.1-1.

**Table D.2.3.1-1: Definition of Sdp data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| part | array [partElem] | 1 | Refer to clause 6.4.5.5.4.3.19. |
| label | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |

### D.2.3.1.1 Type: partElem

This type is required to comply with the provisions defined in table D.2.3.1.1-1.

**Table D.2.3.1.1-1: Definition of partElem data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| index | number (uiny32) | 1 | Refer to clause 6.4.5.5.4.3.19. |
| lines | array [string] | 1 | Refer to clause 6.4.5.5.4.3.19. |

## D.2.3.2 Type: mediaChannel

This type is required to comply with the provisions defined in table D.2.3.2-1.

**Table D.2.3.2-1: Definition of mediaChannel data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| metadata | array(mcMetadata) | 1 | Refer to clause 6.4.5.5.4.3.19. |

### D.2.3.2.1 Type: mcMetadata

This type is required to comply with the provisions defined in table D.2.3.2.1-1.

**Table D.2.3.2.1-1: Definition of mcMetadatal data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| index | number (uint32) | 1 | Refer to clause 6.4.5.5.4.3.19. |
| actType | ActType | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| groupLabel | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| label | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| state | State | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| handlingPref | HandlingPref | 0..1 | Refer to clause 6.4.5.5.4.3.19. |

#### D.2.3.2.1.1 Type: State

This type is required to comply with the provisions defined in table D.2.3.2.1.1-1.

**Table D.2.3.2.1.1-1: Definition of mcMetadatal data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| connected | boolean | 1 | Refer to clause 6.4.5.5.4.3.19. |
| routed | boolean | 1 | Refer to clause 6.4.5.5.4.3.19. |

#### D.2.3.2.1.2 Type: HandlingPref

This type is required to comply with the provisions defined in table D.2.3.2.1.2-1.

**Table D.2.3.2.1.2-1: Definition of mcMetadatal data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| index | number (uint32) | 1 | Refer to clause 6.4.5.5.4.3.19. |
| connectToDevice | ConnectTo Device | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| preferredStyle | PreferredS tyle | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| participantId | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |

## D.2.3.3 Type: dataChannel

This type is required to comply with the provisions defined in table D.2.3.3-1.

**Table D.2.3.3-1: Definition of mediaChannel data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| sdpIndex | number (uint32) | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| metadata | array(dcMe tadata) | 0..1 | Refer to clause 6.4.5.5.4.3.19. |

### D.2.3.3.1 Type: dcMetadata

This type is required to comply with the provisions defined in table D.2.3.3.1-1.

**Table D.2.3.3.1-1: Definition of mcMetadatal data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| id | number (uint16) | 1 | Refer to clause 6.4.5.5.4.3.19. |
| actType | ActType | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| groupLabel | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| label | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| state | State | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| subprotocol | string | 0..1 | Refer to clause 6.4.5.5.4.3.19.<br>equivalent to: a=dcmap subprotocol-opt (IETF RFC 8864 [52]) |
| ordered | boolean | 0..1 | Refer to clause 6.4.5.5.4.3.19.<br>equivalent to: a=dcmap ordering-opt (IETF RFC 8864 [52]) |
| maxretr | number | 0..1 | Refer to clause 6.4.5.5.4.3.19.<br>equivalent to: a=dcmap maxretr-opt (IETF RFC 8864 [52]) |
| maxtime | number | 0..1 | Refer to clause 6.4.5.5.4.3.19.<br>equivalent to: a=dcmap maxtime-opt (IETF RFC 8864 [52]) |
| priority | number | 0..1 | Refer to clause 6.4.5.5.4.3.19.<br>equivalent to: a=dcmap priority-opt (IETF RFC 8864 [52]) |

## D.2.3.4 Type: participantDescElem

This type is required to comply with the provisions defined in table D.2.3.4-1.

**Table D.2.3.4-1: Definition of participantDescElem data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| actType | ActType | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| participantId | string | 1 | Refer to clause 6.4.5.5.4.3.19. |
| displayText | string | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| displayImage | url | 0..1 | Refer to clause 6.4.5.5.4.3.19. |
| oId | OId | 0..1 | This information element is applicable only when the "actType" is set to "add". |
| userState | UserState | 0..1 | Refer to clause 6.4.5.5.4.3.19. |

## D.2.4 Type: OrigId

**Table D.2.4-1: Definition of OrigId data type**

| IE name | Data type | Cardinality | Description |
|---|---|---|---|
| user | object | 0..1 | Refer to clause 6.4.5.5.4.3.20. |
| network | object | 0..1 | Refer to clause 6.4.5.5.4.3.20. |
| privacy | array [string] | 0..1 | Refer to clause 6.4.5.5.4.3.20. |
| passport | object | 0..1 | Refer to clause 6.4.5.5.4.3.20. |
| NOTE: Only one IE is allowed to be set in the object. | | | |

## D.2.5 Type: ResourceReq

**Table D.2.5-1: Definition of ResourceReq data type**

| Information | Data type | Description |
|---|---|---|
| /net/conf/iceServers | object | The array of the RTCIceServer object for WebRTC API which is available for RESPECT endpoint (UE). |

# D.3 Simple data types

This clause describes simple data types for RESPECT.

## D.3.1 Enumeration: MsgType

**Table D.3.1-1: Enumeration MsgType**

| Enumeration value | Description | Applicability (NOTE) |
|---|---|---|
| request | The message is request. | |
| response | The message is response. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | |

## D.3.2 Enumeration: Method

**Table D.3.2-1: Enumeration method**

| Enumeration value | Description | Applicability (NOTE) |
|---|---|---|
| auth | auth method | |
| msetup | msetup method | |
| mupdate | mupdate method | |
| mdisc | mdisc method | |
| Getinfo | getinfo method | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | |

## D.3.3 Enumeration: MsgType

**Table D.3.3-1: Enumeration MsgType**

| Enumeration value | Description | Applicability (NOTE) |
|---|---|---|
| bearer | The authentication request intends to use bearer authentication. | |
| basic | The authentication request intends to use basic authentication. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | |

## D.3.4 Enumeration: MediaSessionState

**Table D.3.4-1: Enumeration MediaSessionState**

| Enumeration value | Description | Applicability (NOTE) |
|---|---|---|
| accepted | Refer to clause 6.4.5.5.4.3.18. | |
| connecting | Refer to clause 6.4.5.5.4.3.18. | |
| connected | Refer to clause 6.4.5.5.4.3.18. | |
| routed | Refer to clause 6.4.5.5.4.3.18. | |
| updateRequesting | Refer to clause 6.4.5.5.4.3.18. | |
| updating | Refer to clause 6.4.5.5.4.3.18. | |
| NOTE: This field is to describe if the use of the information element depends on the feature. | | |

## D.3.5 Enumeration: oaType

**Table D.3.5-1: Enumeration oaType**

| Enumeration value | Description | Applicability |
|---|---|---|
| offer | Refer to clause 6.4.5.5.4.3.19. | |
| preoffer | Refer to clause 6.4.5.5.4.3.19. | |
| answer | Refer to clause 6.4.5.5.4.3.19. | |
| info | Refer to clause 6.4.5.5.4.3.19. | |

## D.3.6 Enumeration: ActType

**Table D.3.6-1: Enumeration ActType**

| Enumeration value | Description | Applicability |
|---|---|---|
| add | Refer to clause 6.4.5.5.4.3.19.<br>Only used when the "part" information element is set to "offer" or "preoffer". | |
| del | Refer to clause 6.4.5.5.4.3.19.<br>Only used when the "part" information element is set to "offer" or "preoffer". | |
| mod | Refer to clause 6.4.5.5.4.3.19.<br>Only used when the "part" information element is set to "offer" or "preoffer". | |
| aly | Refer to clause 6.4.5.5.4.3.19.<br>Only used when the "part" information element is set to "answer". | |
| dcl | Refer to clause 6.4.5.5.4.3.19.<br>Only used when the "part" information element is set to "answer". | |

## D.3.7 Enumeration: ConnectToDevice

**Table D.3.7-1: Enumeration ConnectToDevice**

| Enumeration value | Description | Applicability |
|---|---|---|
| audioin | Refer to clause 6.4.5.5.4.3.19. | |
| audioout | Refer to clause 6.4.5.5.4.3.19. | |
| videoin | Refer to clause 6.4.5.5.4.3.19. | |
| display | Refer to clause 6.4.5.5.4.3.19. | |

## D.3.8 Enumeration: PreferredStyle

**Table D.3.8-1: Enumeration PreferredStyle**

| Enumeration value | Description | Applicability |
|---|---|---|
| mainview | Refer to clause 6.4.5.5.4.3.19. | |
| thumbnail | Refer to clause 6.4.5.5.4.3.19. | |
| screenshare | Refer to clause 6.4.5.5.4.3.19. | |

## D.3.9 Enumeration: UserState

**Table D.3.9-1: Enumeration UserState**

| Enumeration value | Description | Applicability |
|---|---|---|
| joiningIn | Refer to clause 6.4.5.5.4.3.19. | |
| alerting | Refer to clause 6.4.5.5.4.3.19. | |
| joined | Refer to clause 6.4.5.5.4.3.19. | |
| leaving | Refer to clause 6.4.5.5.4.3.19. | |

# Annex E (informative): Change history

| Change history | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2023-11 | SA4#126 | S4-231727 | | | | initial version | 0.1.0 |
| 2023-11 | SA4#126 | S4-231997 | | | | Scope, References, Definitions of terms, symbols and abbreviations, Motivations for Native WebRTC Signalling and assumptions, Key issue #1, Key issue #2, Key issue #3, Key issue #4, Key issue #5, Key issue #6, Key issue #10, Solution #1, Solution #2, Solution #4, Solution #6 are incorporated. | 0.2.0 |
| 2024-01 | SA4#127 | S4-240295 | | | | Scope, Key issue #1, Key issue #5, Key issue #6, Key issue #10, Key issue #11, Solution #1, Solution #2, Solution #5, Solution #10, Solution #11, Conclusions and Recommendations, Use cases. | 0.3.0 |
| 2024-01 | SA4#127 | S4-240340 | | | | Key issue #1, Key issue #7, Key issue #8, Key issue #9, Solution #1, Solution #3, Solution #6, Solution #7, Solution #8, Solution #9, Key findings, Message Examples for RESPECT call flow, Call flow examples for RTC-IMS interworking, JSON data format for RESPECT. | 0.4.0 |
| 2024-03 | SA#103 | SP-240027 | | | | Version 1.0.0 created by MCC | 1.0.0 |
| 2024-03 | | | | | | Version 18.0.0 created by MCC | 18.0.0 |
| 2025-10 | - | - | - | - | - | Update to Rel-19 version (MCC) | **19.0.0** |

# History

| Document history | | |
|---|---|---|
| V19.0.0 | October 2025 | Publication |
| | | |
| | | |
| | | |
| | | |