**Agenda item:** 10.5

**Source:** Qualcomm Inc.

**Title: [iRTCW] Simple WebRTC Application Protocol (SWAP)**

**Document for** Discussion andAgreement

# Introduction

In this contribution, we propose the specification text for the definition of that protocol to address the needs of collaboration scenario 3.

# General

The Simple WebRTC Application Protocol (SWAP) allows the exchange of information to control the setup and management of a WebRTC session between two endpoints. The protocol is used to negotiate and setup media and data channel streams and their associated transport connections. The protocol defines a minimum set of signaling messages to support offer/answer exchange as well as the exchange of IC candidates.

In addition to the message formats, the protocol also defines the transport channel for the protocol messages. The protocol aligns with the RTCPeerConnection API as defined by W3C WebRTC 1.0, which facilitates the integration in web-based WebRTC applications. The JSON format is used to encode the SWAP messages to leverage the native Javascript support for JSON parsing. Efficient JSON parsing is however not limited to web environments and is widely available in all platforms and programming languages.

SWAP is designed to fulfil the agreed requirements, which are listed here for convenience:

* + It shall support any WebRTC application, i.e. it should not overfit for a specific use case.
	+ It shall enable communicating parties to match based on
	+ Flexible matching with a wide range of matching criteria that suit the needs of different WebRTC applications.
	+ Secure matching to avoid security issues such as DDoS attacks
	+ It may be extended in the future to support:
	+ Global matching possibilities for applications that work across multiple WebRTC signaling servers, potentially hosted by different MNOs.
	+ It shall enable communicating parties to verify each other’s identity, if required by the application.
	+ It shall support the secure exchange of messages supporting integrity-protection and/or encryption.
	+ It shall protect user privacy and mitigate the linkability and tracking attack caused by unnecessary user information disclosure.
	Note: detailed security requirements and mechanisms need further co-work with SA3.
	+ It shall support basic session setup messages allowing extensions for application-specific information.
	+ It should be web-friendly to support 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, etc…) defined in IETF and W3C, with an exception for codecs
	+ It shall be simple to implement and deploy (e.g. simpler in complexity compared to SIP).

# Transport

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



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

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

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



SWAP shall be identified by the “3gpp.SWAP.v1” subprotocol identifier in the Sec-WebSocket-Protocol header field, i.e. as part of the HTTP upgrade request.

The SWAP protocol is designed to adhere to the JSON Session Establishment Protocol (JSEP) state machine as defined in RFC8829. The JSEP state machine is reproduced in the following figure.



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

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

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

# Message Syntax and Semantics

# Common Message Fields

# Source Id

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

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

# Message Id

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

The message identifier shall be a positive monotonically increasing number.

Message Type

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

* Register
* Response
* Connect
* Accept
* Reject
* Update
* Close
* Application

# register message

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

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

# Parameters

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

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

* ipv4: The IPv4 address of the target endpoint
* ipv6: The IPv6 address of the target endpoint
* fqdn: The FQDN of the target endpoint
* service: An identifier of a service or an application
* user: An identifier of the user such as a SIP address, a GPSI, or an MSISDN
* eas: An EAS identifier
* app: application-specific matching criteria that is compared using binary or string comparison.
* location: one or more identifiers of a geographic location or area.
* qos: a description of the QoS that is supported by the connection to the endpoint.
* processing: a profile description of the processing capabilities of the endpoint.

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

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

# response message

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

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

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

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

# Parameters

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

**source:** the source identifier of the message source.

**request**: the message identifier of the request.

**description**: a description of the error message.

# connect message

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

# Parameters

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

**matching\_criteria**: an array that contains the matching criteria for the target endpoint. Each object shall be comply with the definition of a matching criteria as described in clause 4.2.1.

# accept message

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

# Parameters

**answer**: This parameter shall contain the answer SDP.

# update message

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

# Parameters

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

# reject message

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

# Parameters

**source:** the source identifier of the message source.

**request**: the message identifier of the request.

**error\_id**: an identifier of the error message.

**description**: a description of the error message.

# close message

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

# application message

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

# Parameters

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

**value**: an object that contains the application message content.

# Integrity and Security

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

* a key derivation mechanism is configured by the application provider to the session participants, e.g. using a shared secret algorithm
* For integrity protection, the derived key is used to provide integrity protection, e.g. using a Message Authentication Code (MAC) for message payload.
* For encryption, the derived key is used to encrypt the message payload. The encrypted data may then be encoded using base64 to enable embedding it in JSON.

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

# JSON Schema

Th e JSON schema of the SWAP messages is defined in the following table:

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

# Proposal

We propose to agree the proposed definition of the SWAP protocol into the TS.