Workgroup: wish
Internet-Draft: draft-ietf-wish-whip-10
Updates: RFC8842 (if approved)
Published: 14 December 2023
Intended Status: Standards Track
Expires: 16 June 2024
Authors: S. Murillo A. Gouaillard
Millicast CoSMo Software

WebRTC-HTTP ingestion protocol (WHIP)

Abstract

This document describes a simple HTTP-based protocol that will allow WebRTC-based ingestion of content into streaming services and/or CDNs.

This document updates RFC 8842.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The IETF RTCWEB working group standardized JSEP ([RFC8829]), a mechanism used to control the setup, management, and teardown of a multimedia session. It also describes how to negotiate media flows using the Offer/Answer Model with the Session Description Protocol (SDP) [RFC3264] including the formats for data sent over the wire (e.g., media types, codec parameters, and encryption). WebRTC intentionally does not specify a signaling transport protocol at application level.

Unfortunately, the lack of a standardized signaling mechanism in WebRTC has been an obstacle to adoption as an ingestion protocol within the broadcast/streaming industry, where a streamlined production pipeline is taken for granted: plug in cables carrying raw media to hardware encoders, then push the encoded media to any streaming service or Content Delivery Network (CDN) ingest using an ingestion protocol.

While WebRTC can be integrated with standard signaling protocols like SIP [RFC3261] or XMPP [RFC6120], they are not designed to be used in broadcasting/streaming services, and there is also no sign of adoption in that industry. RTSP [RFC7826], which is based on RTP, does not support the SDP offer/answer model [RFC3264] for negotiating the characteristics of the media session.

This document proposes a simple protocol based on HTTP for supporting WebRTC as media ingestion method which:

* Is easy to implement,

* Is as easy to use as popular IP-based broadcast protocols

* Is fully compliant with WebRTC and RTCWEB specs

* Enables ingestion on both traditional media platforms and WebRTC end-to-end platforms, achieving the lowest possible latency.

* Lowers the requirements on both hardware encoders and broadcasting services to support WebRTC.

* Is usable both in web browsers and in standalone encoders.

2. Terminology

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Overview

The WebRTC-HTTP Ingest Protocol (WHIP) is designed to facilitate a one-time exchange of Session Description Protocol (SDP) offers and answers using HTTP POST requests. This exchange is a fundamental step in establishing an Interactive Connectivity Establishment (ICE) and Datagram Transport Layer Security (DTLS) session between the WHIP client, which represents the encoder or media producer, and the media server, the broadcasting ingestion endpoint.

Upon successful establishment of the ICE/DTLS session, unidirectional media data transmission commences from the WHIP client to the media server. It is important to note that SDP renegotiations are not supported in WHIP, meaning that no modifications to the "m=" sections can be made after the initial SDP offer/answer exchange via HTTP POST is completed.

The following diagram illustrates the core operation of the WHIP protocol for initiating and terminating a WHIP session:

```
 +-------------+    +---------------+ +--------------+ +---------------+
 | WHIP client |    | WHIP endpoint | | media server | | WHIP session  |
 +--+----------+    +---------+-----+ +------+-------+ +--------|------+
    |                         |              |                  |
    |                         |              |                  |
    |HTTP POST (SDP Offer)    |              |                  |
    +------------------------>+              |                  |
    |201 Created (SDP answer) |              |                  |
    +<------------------------+              |                  |
    |          ICE REQUEST                   |                  |
    +--------------------------------------->+                  |
    |          ICE RESPONSE                  |                  |
    |<---------------------------------------+                  |
    |          DTLS SETUP                    |                  |
    |<======================================>|                  |
    |          RTP/RTCP FLOW                 |                  |
    +<-------------------------------------->+                  |
    | HTTP DELETE                                               |
    +---------------------------------------------------------->+
    | 200 OK                                                    |
    <-----------------------------------------------------------x
```

Figure 1: WHIP session setup and teardown

The elements in Figure 1 are described as follows:

* WHIP client: This represents the WebRTC media encoder or producer, which functions as a client of the WHIP protocol by encoding and delivering media to a remote media server.

* WHIP endpoint: This denotes the ingest server that receives the initial WHIP request.

* WHIP endpoint URL: Refers to the URL of the WHIP endpoint responsible for creating the WHIP session.

* media server: This is the WebRTC media server or consumer responsible for establishing the media session with the WHIP client and receiving the media content it produces.

* WHIP session: Indicates the allocated WHIP session by the WHIP endpoint for an ongoing ingest session.

* WHIP session URL: Refers to the URL of the WHIP resource allocated by the WHIP endpoint for a specific media session. The WHIP client can send requests to the WHIP session using this URL to modify the session, such as ICE operations or termination.

4. Protocol Operation

In order to set up an ingestion session, the WHIP client will generate an SDP offer according to the JSEP rules and perform an HTTP POST request as per [RFC9110] Section 9.3.3 to the configured WHIP endpoint URL.

The HTTP POST request **MUST** have a content type of "application/sdp" and contain the SDP offer as the body. The WHIP endpoint will generate an SDP answer and return a "201 Created" response with a content type of "application/sdp", the SDP answer as the body, and a Location header field pointing to the newly created WHIP session.

As the WHIP protocol only supports the ingestion use case with unidirectional media, the WHIP client **SHOULD** use "sendonly" attribute in the SDP offer but **MAY** use the "sendrecv" attribute instead, "inactive" and "recvonly" attributes **MUST NOT** be used. The WHIP endpoint **MUST** use "recvonly" attribute in the SDP answer.

If the HTTP POST to the WHIP endpoint has a content type different than "application/sdp", the WHIP endpoint **MUST** reject the HTTP POST request with a "415 Unsupported Media Type" error response.

If the SDP body is malformed, the WHIP session **MUST** reject the HTTP POST with a "400 Bad Request" error response.

Following is an example of an HTTP POST sent from a WHIP client to a WHIP endpoint and the "201 Created" response from the WHIP endpoint containing the Location header pointing to the newly created WHIP session:

```
POST /whip/endpoint HTTP/1.1
Host: whip.example.com
Content-Type: application/sdp
Content-Length: 1326

v=0
o=- 5228595038118931041 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE 0 1
a=extmap-allow-mixed
a=msid-semantic: WMS
m=audio 9 UDP/TLS/RTP/SAVPF 111
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:EsAw
a=ice-pwd:bP+XJMM09aR8AiX1jdukzR6Y
a=ice-options:trickle
a=fingerprint:sha-256 DA:7B:57:DC:28:CE:04:4F:31:79:85:C4:31:67:EB:27:58
a=setup:actpass
a=mid:0
a=bundle-only
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=sendonly
a=msid:- d46fb922-d52a-4e9c-aa87-444eadc1521b
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10;useinbandfec=1
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:EsAw
a=ice-pwd:bP+XJMM09aR8AiX1jdukzR6Y
a=ice-options:trickle
a=fingerprint:sha-256 DA:7B:57:DC:28:CE:04:4F:31:79:85:C4:31:67:EB:27:58
a=setup:actpass
a=mid:1
a=bundle-only
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:10 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:11 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
a=sendonly
a=msid:- d46fb922-d52a-4e9c-aa87-444eadc1521b
a=rtcp-mux
a=rtcp-rsize
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
```

```
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96

HTTP/1.1 201 Created
ETag: "xyzzy"
Content-Type: application/sdp
Content-Length: 1400
Location: https://whip.example.com/session/id

v=0
o=- 1657793490019 1 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE 0 1
a=extmap-allow-mixed
a=ice-lite
a=msid-semantic: WMS *
m=audio 9 UDP/TLS/RTP/SAVPF 111
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:38sdf4fdsf54
a=ice-pwd:2e13dde17c1cb009202f627fab90cbec358d766d049c9697
a=fingerprint:sha-256 F7:EB:F3:3E:AC:D2:EA:A7:C1:EC:79:D9:B3:8A:35:DA:70
a=candidate:1 1 UDP 2130706431 198.51.100.1 39132 typ host
a=setup:passive
a=mid:0
a=bundle-only
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=recvonly
a=rtcp-mux
a=rtcp-rsize
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10;useinbandfec=1
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:38sdf4fdsf54
a=ice-pwd:2e13dde17c1cb009202f627fab90cbec358d766d049c9697
a=fingerprint:sha-256 F7:EB:F3:3E:AC:D2:EA:A7:C1:EC:79:D9:B3:8A:35:DA:70
a=candidate:1 1 UDP 2130706431 198.51.100.1 39132 typ host
a=setup:passive
a=mid:1
a=bundle-only
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:10 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=extmap:11 urn:ietf:params:rtp-hdrext:sdes:repaired-rtp-stream-id
a=recvonly
a=rtcp-mux
a=rtcp-rsize
```

```
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
```

Figure 2: Example of SDP offer/answer exchange done via an HTTP POST

Once a session is setup, consent freshness as per [RFC7675] **SHALL** be used to detect non-graceful disconnection and DTLS teardown for session termination by either side.

To explicitly terminate a session, the WHIP client **MUST** perform an HTTP DELETE request to the resource URL returned in the Location header field of the initial HTTP POST. Upon receiving the HTTP DELETE request, the WHIP session will be removed and the resources freed on the media server, terminating the ICE and DTLS sessions.

A media server terminating a session **MUST** follow the procedures in [RFC7675] Section 5.2 for immediate revocation of consent.

The WHIP endpoints **MUST** return an "405 Method Not Allowed" response for any HTTP request method different than OPTIONS and POST on the endpoint URL in order to reserve their usage for future versions of this protocol specification.

The WHIP endpoints **MUST** support OPTIONS requests for Cross-Origin Resource Sharing (CORS) as defined in [FETCH]. The "200 OK" response to any OPTIONS request **SHOULD** include an "Accept-Post" header with a media type value of "application/sdp" as per [W3C.REC-ldp-20150226].

The WHIP sessions **MUST** return an "405 Method Not Allowed" response for any HTTP request method different than PATCH and DELETE on the session URLs in order to reserve their usage for future versions of this protocol specification.

4.1. ICE support

ICE [RFC8838] is a protocol addressing the complexities of Network Address Translation (NAT) traversal, commonly encountered in Internet communication. NATs hinder direct communication between devices on different local networks, posing challenges for real-time applications. ICE facilitates seamless connectivity by employing techniques to discover and negotiate efficient communication paths.

Trickle ICE optimizes the connectivity process by incrementally sharing potential communication paths, reducing latency, and facilitating quicker establishment.

ICE Restarts are crucial for maintaining connectivity in dynamic network conditions or disruptions, allowing devices to re-establish communication paths without complete renegotiation. This ensures minimal latency and reliable real-time communication.

Trickle ICE and ICE restart support are **RECOMMENDED** for both WHIP sessions and clients.

4.1.1. HTTP PATCH request usage

The WHIP client **MAY** perform trickle ICE or ICE restarts by sending an HTTP PATCH request as per [RFC5789] to the WHIP session URL, with a body containing a SDP fragment with media type "application/trickle-ice-sdpfrag" as specified in [RFC8840]. When used for trickle ICE, the body of this PATCH message will contain the new ICE candidate; when used for ICE restarts, it will contain a new ICE ufrag/pwd pair as defined in [RFC8838] Section 5.4.

If the HTTP POST to the WHIP session has a content type different than "application/trickle-ice-sdpfrag", the WHIP session **MUST** reject the HTTP POST request with a "415 Unsupported Media Type" error response. If the SDP fragment is malformed, the WHIP session **MUST** reject the HTTP POST with a "400 Bad Request" error response.

If the WHIP session supports either Trickle ICE or ICE restarts, but not both, it **MUST** return a "422 Unprocessable Content" response for the HTTP PATCH requests that are not supported as per [RFC9110] Section 15.5.21. If neither feature is supported, the WHIP session **MUST** return a "501 Not Implemented" response for such HTTP PATCH requests, as described in [RFC9110] Section 15.6.2.

The WHIP client **MAY** send overlapping HTTP PATCH requests to one WHIP session. Consequently, as those HTTP PATCH requests may be received out-of-order by the WHIP session, if WHIP session supports ICE restarts, it **MUST** generate a unique strong entity-tag identifying the ICE session as per [RFC9110] Section 8.8.3, being **OPTIONAL** otherwise. The initial value of the entity-tag identifying the initial ICE session **MUST** be returned in an ETag header field in the "201 Created" response to the initial POST request to the WHIP endpoint.

WHIP clients **SHOULD NOT** use entity-tag validation when matching a specific ICE session is not required, such as for example when initiating a DELETE request to terminate a session. WHIP sessions **MUST** ignore any entity-tag value sent by the WHIP client when ICE session matching is not required, as in the HTTP DELETE request.

4.1.2. Trickle ICE

Depending on the Trickle ICE support on the WHIP client, the initial offer by the WHIP client **MAY** be sent after the full ICE gathering is complete with the full list of ICE candidates, or it **MAY** only contain local candidates (or even an empty list of candidates) as per [RFC8863]. In order to reduce the setup times, Trickle ICE support is **RECOMMENDED** for WHIP clients and the WHIP client **SHOULD** send the SDP offer as soon as possible containing either local gathered ICE candidates or an empty list of candidates.

Because the WHIP client needs to know the entity-tag associated with the ICE session in order to send a PATCH request containing new ICE candidates, it **MUST** wait and buffer any gathered candidates until it receives the HTTP response with the new entity-tag value to either the initial POST request or the last PATCH request performing an ICE restart. In order to lower the HTTP traffic and processing time required, the WHIP client **SHOULD** send a single aggregated HTTP PATCH request with all the buffered ICE candidates once it receives the new entity-tag value.

In order to simplify the protocol, exchanging media server's gathered ICE candidates after sending the SDP answer is not supported. The WHIP Endpoint **SHALL** gather all the ICE candidates for the media server before responding to the client request and the SDP answer **SHALL** contain the full list of ICE candidates of the media server. The media server **MAY** use ICE lite, while the WHIP client **MUST** implement full ICE.

A WHIP client sending a PATCH request for performing trickle ICE **MUST** include an "If-Match" header field with the latest known entity-tag as per [RFC9110] Section 13.1.1. When the PATCH request is received by the WHIP session, it **MUST** compare the indicated entity-tag value with the current entity-tag of the resource as per [RFC9110] Section 13.1.1 and return a "412 Precondition Failed" response if they do not match. If the HTTP PATCH request does not contain an "If-Match" header the WHIP session **MUST** return an "428 Precondition Required" response as per [RFC6585] Section 3.

When a WHIP session receives a PATCH request that adds new ICE candidates without performing an ICE restart, it **MUST** return a "204 No Content" response without a body and **MUST NOT** include an ETag header in the response. If the WHIP session does not support a candidate transport or is not able to resolve the connection address, it **MUST** silently discard the candidate and continue processing the rest of the request normally.

```
PATCH /session/id HTTP/1.1
Host: whip.example.com
If-Match: "xyzzy"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 548

a=ice-ufrag:EsAw
a=ice-pwd:P2uYro0UCOQ4zxjKXaWCBui1
m=audio 9 RTP/AVP 0
a=mid:0
a=candidate:1387637174 1 udp 2122260223 192.0.2.1 61764 typ host generat
a=candidate:3471623853 1 udp 2122194687 198.51.100.1 61765 typ host gene
a=candidate:473322822 1 tcp 1518280447 192.0.2.1 9 typ host tcptype acti
a=candidate:2154773085 1 tcp 1518214911 198.51.100.2 9 typ host tcptype
a=end-of-candidates

HTTP/1.1 204 No Content
```

Figure 3: Example of a Trickle ICE request and response

4.1.3. ICE Restarts

A WHIP client sending a PATCH request for performing ICE restart **MUST** contain an "If-Match" header field with a field-value "*" as per [RFC9110] Section 13.1.1.

If the HTTP PATCH request results in an ICE restart, the WHIP session **SHALL** return a "200 OK" with an "application/trickle-ice-sdpfrag" body containing the new ICE username fragment and password and OPTIONALLY a new set of ICE candidates for the WHIP client. Also, the "200 OK" response for a successful ICE restart **MUST** contain the new entity-tag corresponding to the new ICE session in an ETag response header field and **MAY** contain a new set of ICE candidates for the media server.

If the ICE restart request cannot be satisfied by the WHIP session, the resource **MUST** return an appropriate HTTP error code and **MUST NOT** terminate the session immediately and keep the existing ICE session. The WHIP client **MAY** retry performing a new ICE restart or terminate the session by issuing an HTTP DELETE request instead. In any case, the session **MUST** be terminated if the ICE consent expires as a consequence of the failed ICE restart as per [RFC7675] Section 5.1.

In case of unstable network conditions, the ICE restart HTTP PATCH requests and responses might be received out of order. In order to mitigate this scenario, when the client performs an ICE restart, it **MUST** discard any previous ICE username and passwords fragments and ignore any further HTTP PATCH response received from a pending HTTP PATCH request. WHIP clients **MUST** apply only the ICE information received in the response to the last sent request. If there is a mismatch between the ICE information at the WHIP client and at the WHIP session (because of an out-of-order request), the STUN requests will contain invalid ICE information and will be dropped by the receiving side. If this situation is detected by the WHIP Client, it **MUST** send a new ICE restart request to the server.

```
PATCH /session/id HTTP/1.1
Host: whip.example.com
If-Match: "*"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 54

a=ice-ufrag:ysXw
a=ice-pwd:vw5LmwG4y/e6dPP/zAP9Gp5k

HTTP/1.1 200 OK
ETag: "abccd"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 102

a=ice-lite
a=ice-ufrag:289b31b754eaa438
a=ice-pwd:0b66f472495ef0ccac7bda653ab6be49ea13114472a5d10a
```

Figure 4: Example of an ICE restart request and response

4.2. WebRTC constraints

In order to reduce the complexity of implementing WHIP in both clients and media servers, WHIP imposes the following restrictions regarding WebRTC usage:

4.2.1. SDP Bundle

Both the WHIP client and the WHIP endpoint **SHALL** support and use SDP bundle [RFC9143]. Each "m=" section **MUST** be part of a single BUNDLE group. Hence, when a WHIP client sends an SDP offer, it **MUST** include a "bundle-only" attribute in each bundled "m=" section. The WHIP client and the media server **MUST** support multiplexed media associated with the BUNDLE group as per [RFC9143] Section 9. In addition, per [RFC9143] the WHIP client and media server **SHALL** use RTP/RTCP multiplexing for all bundled media. In order to reduce the network resources required at the media server, both The WHIP client and WHIP endpoints **MUST** include the "rtcp-mux-only" attribute in each bundled "m=" sections as per [RFC8858] Section 3.

4.2.2. Single MediaStream

This version of the specification only supports a single MediaStream as defined in [RFC8830] and therefore all "m=" sections **MUST** contain an "msid" attribute with the same value. The MediaStream **MUST** contain at least one MediaStreamTrack of any media kind and it **MUST NOT** have two or more than MediaStreamTracks for the same media (audio or video). However, it would be possible for future revisions of this spec to allow more than a single MediaStream or MediaStreamTrack of each media kind, so in order to ensure forward compatibility, if the number of audio and or video MediaStreamTracks or number of MediaStreams are not supported by the WHIP Endpoint, it **MUST** reject the HTTP POST request with a "406 Not Acceptable" error response.

4.2.3. No partially successful answers

The WHIP Endpoint **SHOULD NOT** reject individual "m=" sections as per [RFC8829] Section 5.3.1 in case there is any error processing the "m=" section, but reject the HTTP POST request with a "406 Not Acceptable" error response to prevent having partially successful WHIP sessions which can be misleading to end users.

4.2.4. DTLS setup role and SDP "setup" attribute

When a WHIP client sends an SDP offer, it **SHOULD** insert an SDP "setup" attribute with an "actpass" attribute value, as defined in [RFC8842]. However, if the WHIP client only implements the DTLS client role, it **MAY** use an SDP "setup" attribute with an "active" attribute value. If the WHIP endpoint does not support an SDP offer with an SDP "setup" attribute with an "active" attribute value, it **SHOULD** reject the request with a "422 Unprocessable Entity" response.

NOTE: [RFC8842] defines that the offerer must insert an SDP "setup" attribute with an "actpass" attribute value. However, the WHIP client will always communicate with a media server that is expected to support the DTLS server role, in which case the client might choose to only implement support for the DTLS client role.

4.2.5. Trickle ICE and ICE restarts

Trickle ICE and ICE restarts support is **OPTIONAL** for both the WHIP clients and media servers as explained in section 4.1.

4.3. Load balancing and redirections

WHIP endpoints and media servers might not be colocated on the same server, so it is possible to load balance incoming requests to different media servers.

WHIP clients **SHALL** support HTTP redirections as per [RFC9110] Section 15.4. In order to avoid POST requests to be redirected as GET requests, status codes 301 and 302 **MUST NOT** be used and the preferred method for performing load balancing is via the "307 Temporary Redirect" response status code as described in [RFC9110] Section 15.4.8. Redirections are not required to be supported for the PATCH and DELETE requests.

In case of high load, the WHIP endpoints **MAY** return a "503 Service Unavailable" response indicating that the server is currently unable to handle the request due to a temporary overload or scheduled maintenance as described in [RFC9110] Section 15.6.4, which will likely be alleviated after some delay. The WHIP endpoint might send a Retry-After header field indicating the minimum time that the user agent ought to wait before making a follow-up request as described in [RFC9110] Section 10.2.3.

4.4. STUN/TURN server configuration

The WHIP endpoint **MAY** return STUN/TURN server configuration URLs and credentials usable by the client in the "201 Created" response to the HTTP POST request to the WHIP endpoint URL.

A reference to each STUN/TURN server will be returned using the "Link" header field [RFC8288] with a "rel" attribute value of "ice-server". The Link target URI is the server URI as defined in [RFC7064] and [RFC7065]. The credentials are encoded in the Link target attributes as follows:

* username: If the Link header field represents a TURN server, and credential-type is "password", then this attribute specifies the username to use with that TURN server.

* credential: If the "credential-type" attribute is missing or has a "password" value, the credential attribute represents a long-term authentication password, as described in [RFC8489], Section 10.2.

* credential-type: If the Link header field represents a TURN server, then this attribute specifies how the credential attribute value should be used when that TURN server requests authorization. The default value if the attribute is not present is "password".

```
  Link: <stun:stun.example.net>; rel="ice-server"
  Link: <turn:turn.example.net?transport=udp>; rel="ice-server";
        username="user"; credential="myPassword"; credential-type="pa
  Link: <turn:turn.example.net?transport=tcp>; rel="ice-server";
        username="user"; credential="myPassword"; credential-type="pa
  Link: <turns:turn.example.net?transport=tcp>; rel="ice-server";
        username="user"; credential="myPassword"; credential-type="pa
```

Figure 5: Example of a STUN/TURN servers configuration

NOTE: The naming of both the "rel" attribute value of "ice-server" and the target attributes follows the one used on the W3C WebRTC recommendation [W3C.REC-webrtc-20210126] RTCConfiguration dictionary in section 4.2.1. "rel" attribute value of "ice-server" is not prepended with the "urn:ietf:params:whip:" so it can be reused by other specifications which may use this mechanism to configure the usage of STUN/TURN servers.

NOTE: Depending on the ICE Agent implementation, the WHIP client may need to call the setConfiguration method before calling the setLocalDescription method with the local SDP offer in order to avoid having to perform an ICE restart for applying the updated STUN/TURN server configuration on the next ICE gathering phase.

There are some WebRTC implementations that do not support updating the STUN/TURN server configuration after the local offer has been created as specified in [RFC8829] Section 4.1.18. In order to support these clients, the WHIP endpoint **MAY** also include the STUN/TURN server configuration on the responses to OPTIONS request sent to the WHIP endpoint URL before the POST request is sent. However, this method is not **NOT RECOMMENDED** to be used by the WHIP clients and, if supported by the underlying WHIP Client's webrtc implementation, the WHIP Client **SHOULD** wait for the information to be returned by the WHIP Endpoint on the response of the HTTP POST request instead.

The generation of the TURN server credentials may require performing a request to an external provider, which can both add latency to the OPTIONS request processing and increase the processing required to handle that request. In order to prevent this, the WHIP Endpoint **SHOULD NOT** return the STUN/TURN server configuration if the OPTIONS request is a preflight request for CORS as defined in [FETCH], that is, if The OPTIONS request does not contain an Access-Control-Request-Method with "POST" value and the the Access-Control-Request-Headers HTTP header does not contain the "Link" value.

The WHIP clients **MAY** also support configuring the STUN/TURN server URIs with long term credentials provided by either the broadcasting service or an external TURN provider, overriding the values provided by the WHIP endpoint.

4.5. Authentication and authorization

All WHIP endpoints, sessions and clients **MUST** support HTTP Authentication as per [RFC9110] Section 11 and in order to ensure interoperability, bearer token authentication as defined in the next section **MUST** be supported by all WHIP entities. However this does not preclude the support of additional HTTP authentication schemes as defined in [RFC8819] Section 11.6.

4.5.1. Bearer token authentication

WHIP endpoints and sessions **MAY** require the HTTP request to be authenticated using an HTTP Authorization header field with a Bearer token as specified in [RFC6750] Section 2.1. WHIP clients **MUST** implement this authentication and authorization mechanism and send the HTTP Authorization header field in all HTTP requests sent to either the WHIP endpoint or session except the preflight OPTIONS requests for CORS.

The nature, syntax, and semantics of the bearer token, as well as how to distribute it to the client, is outside the scope of this document. Some examples of the kind of tokens that could be used are, but are not limited to, JWT tokens as per [RFC6750] and [RFC8725] or a shared secret stored on a database. The tokens are typically made available to the end user alongside the WHIP endpoint URL and configured on the WHIP clients (similar to the way RTMP URLs and Stream Keys are distributed).

WHIP endpoints and sessions could perform the authentication and authorization by encoding an authentication token within the URLs for the WHIP endpoints or sessions instead. In case the WHIP client is not configured to use a bearer token, the HTTP Authorization header field must not be sent in any request.

4.6. Simulcast and scalable video coding

Simulcast as per [RFC8853] **MAY** be supported by both the media servers and WHIP clients through negotiation in the SDP offer/answer.

If the client supports simulcast and wants to enable it for publishing, it **MUST** negotiate the support in the SDP offer according to the procedures in [RFC8853] Section 5.3. A server accepting a simulcast offer **MUST** create an answer according to the procedures [RFC8853] Section 5.3.2.

It is possible for both media servers and WHIP clients to support Scalable Video Coding (SVC). However, as there is no universal negotiation mechanism in SDP for SVC, the encoder must consider the negotiated codec(s), intended usage, and SVC support in available decoders when configuring SVC.

4.7. Protocol extensions

In order to support future extensions to be defined for the WHIP protocol, a common procedure for registering and announcing the new extensions is defined.

Protocol extensions supported by the WHIP sessions **MUST** be advertised to the WHIP client in the "201 Created" response to the initial HTTP POST request sent to the WHIP endpoint. The WHIP endpoint **MUST** return one "Link" header field for each extension that it supports, with the extension "rel" attribute value containing the extension URN and the URL for the HTTP resource that will be available for receiving requests related to that extension.

Protocol extensions are optional for both WHIP clients and servers. WHIP clients **MUST** ignore any Link attribute with an unknown "rel" attribute value and WHIP session **MUST NOT** require the usage of any of the extensions.

Each protocol extension **MUST** register a unique "rel" attribute value at IANA starting with the prefix: "urn:ietf:params:whip:ext" as defined in Section 6.3.

For example, considering a potential extension of server-to-client communication using server-sent events as specified in https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events, the URL for connecting to the server-sent event resource for the published stream could be returned in the initial HTTP "201 Created" response with a "Link" header field and a "rel" attribute of "urn:ietf:params:whip:ext:example:server-sent-events" (this document does not specify such an extension, and uses it only as an example).

In this theoretical case, the "201 Created" response to the HTTP POST request would look like:

```
HTTP/1.1 201 Created
Content-Type: application/sdp
Location: https://whip.example.com/session/id
Link: <https://whip.ietf.org/publications/213786HF/sse>;
      rel="urn:ietf:params:whip:ext:example:server-sent-events"
```

Figure 6: Example of a WHIP protocol extension

5. Security Considerations

This document specifies a new protocol on top of HTTP and WebRTC, thus, security protocols and considerations from related specifications apply to the WHIP specification. These include:

* WebRTC security considerations: [RFC8826]. HTTPS **SHALL** be used in order to preserve the WebRTC security model.

* Transport Layer Security (TLS): [RFC8446] and [RFC9147].

* HTTP security: Section 11 of [RFC9112] and Section 17 of [RFC9110].

* URI security: Section 7 of [RFC3986].

On top of that, the WHIP protocol exposes a thin new attack surface specific of the REST API methods used within it:

* HTTP POST flooding and resource exhaustion: It would be possible for an attacker in possession of authentication credentials valid to publish a WHIP stream to make multiple HTTP POST to the WHIP endpoint. This will force the WHIP endpoint to process the incoming SDP and allocate resources for being able to setup the DTLS/ICE connection. While the malicious client does not need to initiate the DTLS/ICE connection at all, the WHIP session will have to wait for the DTLS/ICE connection timeout in order to release the associated resources. If the connection rate is high enough, this could lead to resource exhaustion on the servers handling the requests and it will not be able to process legitimate incoming publications. In order to prevent this scenario, WHIP endpoints **SHOULD** implement a rate limit and avalanche control mechanism for incoming initial HTTP POST requests.

* Insecure direct object references (IDOR) on the WHIP session locations: If the URLs returned by the WHIP endpoint for the WHIP sessions location are easy to guess, it would be possible for an attacker to send multiple HTTP DELETE requests and terminate all the WHIP sessions currently running. In order to prevent this scenario, WHIP endpoints **SHOULD** generate URLs with enough randomness, using a cryptographically secure pseudorandom number generator following the best practices in Randomness Requirements for Security [RFC4086], and implement a rate limit and avalanche control mechanism for HTTP DELETE requests. The security considerations for Universally Unique IDentifier (UUID) [RFC4122] Section 6 are applicable for generating the WHIP sessions location URL.

* HTTP PATCH flooding: Similar to the HTTP POST flooding, a malicious client could also create a resource exhaustion by sending multiple HTTP PATCH request to the WHIP session, although the WHIP sessions can limit the impact by not allocating new ICE candidates and reusing the existing ICE candidates when doing ICE restarts. In order to prevent this scenario, WHIP endpoints **SHOULD** implement a rate limit and avalanche control mechanism for incoming HTTP PATCH requests.

6. IANA Considerations

This specification adds a new link relation type and a registry for URN sub-namespaces for WHIP protocol extensions.

6.1. Link Relation Type: ice-server

The link relation type below has been registered by IANA per Section 4.2 of [RFC8288].

Relation Name: ice-server

Description: Conveys the STUN and TURN servers that can be used by an ICE Agent to establish a connection with a peer.

Reference: TBD

6.2. Registration of WHIP URN Sub-namespace and WHIP Registry

IANA is asked to add an entry to the "IETF URN Sub-namespace for Registered Protocol Parameter Identifiers" registry and create a sub-namespace for the Registered Parameter Identifier as per [RFC3553]: "urn:ietf:params:whip".

To manage this sub-namespace, IANA is asked to create the "WebRTC-HTTP ingestion protocol (WHIP) URNs" registry, which is used to manage entries within the "urn:ietf:params:whip" namespace. The registry description is as follows:

* Registry name: WebRTC-HTTP ingestion protocol (WHIP) URNs

* Specification: this document (RFC TBD)

* Registration policy: Specification Required

* Repository: See Section Section 6.3

* Index value: See Section Section 6.3

6.3. URN Sub-namespace for WHIP

WHIP Endpoint utilizes URNs to identify the supported WHIP protocol extensions on the "rel" attribute of the Link header as defined in Section 4.7.

This section creates and registers an IETF URN Sub-namespace for use in the WHIP specifications and future extensions.

6.3.1. Specification Template

Namespace ID:

* The Namespace ID "whip" has been assigned.

Registration Information:

* Version: 1

* Date: TBD

Declared registrant of the namespace:

* Registering organization: The Internet Engineering Task Force.

* Designated contact: A designated expert will monitor the WHIP public mailing list, "wish@ietf.org".

Declaration of Syntactic Structure:

* The Namespace Specific String (NSS) of all URNs that use the "whip" Namespace ID shall have the following structure: urn:ietf:params:whip:{type}:{name}:{other}.

* The keywords have the following meaning:

  - type: The entity type. This specification only defines the "ext" type.

  - name: A required US-ASCII string that conforms to the URN syntax requirements (see [RFC8141]) and defines a major namespace of a WHIP protocol extension. The value **MAY** also be an industry name or organization name.

  - other: Any US-ASCII string that conforms to the URN syntax requirements (see [RFC8141]) and defines the sub-namespace (which **MAY** be further broken down in namespaces delimited by colons) as needed to uniquely identify an WHIP protocol extension.

Relevant Ancillary Documentation:

* None

Identifier Uniqueness Considerations:

* The designated contact shall be responsible for reviewing and enforcing uniqueness.

Identifier Persistence Considerations:

* Once a name has been allocated, it **MUST NOT** be reallocated for a different purpose.

* The rules provided for assignments of values within a sub-namespace **MUST** be constructed so that the meanings of values cannot change.

* This registration mechanism is not appropriate for naming values whose meanings may change over time.

Process of Identifier Assignment:

* Namespace with type "ext" (e.g., "urn:ietf:params:whip:ext") is reserved for IETF-approved WHIP specifications.

Process of Identifier Resolution:

* None specified.

Rules for Lexical Equivalence:

* No special considerations; the rules for lexical equivalence specified in [RFC8141] apply.

Conformance with URN Syntax:

* No special considerations.

Validation Mechanism:

* None specified.

Scope:

* Global.

6.4. Registering WHIP Protocol Extensions URNs

This section defines the process for registering new WHIP protocol extensions URNs with IANA in the "WebRTC-HTTP ingestion protocol (WHIP) URNs" registry (see Section 6.3).

A WHIP Protocol Extension URNs is used as a value in the "rel" attribute of the Link header as defined in Section 4.7 for the purpose of signaling the WHIP protocol extensions supported by the WHIP Endpoints.

WHIP Protocol Extensions URNs have a "ext" type as defined in Section 6.3.

6.4.1. Registration Procedure

The IETF has created a mailing list, "wish@ietf.org", which can be used for public discussion of WHIP protocol extensions proposals prior to registration. Use of the mailing list is strongly encouraged. The IESG has appointed a designated expert RFC8126 who will monitor the wish@ietf.org mailing list and review registrations.

Registration of new "ext" type URNs (in the namespace "urn:ietf:params:whip:ext") belonging to a WHIP Protocol Extension **MUST** be documented in a permanent and readily available public specification, in sufficient detail so that interoperability between independent implementations is possible and reviewed by the designated expert as per [BCP26] Section 4.6. An RFC is **REQUIRED** for the registration of new value data types that modify existing properties. An RFC is also **REQUIRED** for registration of WHIP Protocol Extensions URNs that modify WHIP Protocol Extensions previously documented in an existing RFC.

The registration procedure begins when a completed registration template, defined in the sections below, is sent to iana@iana.org. Decisions made by the designated expert can be appealed to an Applications and Real Time (ART) Area Director, then to the IESG. The normal appeals procedure described in [BCP9] is to be followed.

Once the registration procedure concludes successfully, IANA creates or modifies the corresponding record in the WHIP Protocol Extension registry.

An RFC specifying one or more new WHIP Protocol Extension URNs **MUST** include the completed registration templates, which **MAY** be expanded with additional information. These completed templates are intended to go in the body of the document, not in the IANA Considerations section. The RFC **MUST** include the syntax and semantics of any extension-specific attributes that may be provided in a Link header field advertising the extension.

6.4.2. Guidance for Designated Experts

The Designated Expert (DE) is expected to ascertain the existence of suitable documentation (a specification) as described in RFC8126 and to verify that the document is permanently and publicly available.

The DE is also expected to check the clarity of purpose and use of the requested registration.

Additionally, the DE must verify that any request for one of these registrations has been made available for review and comment within the IETF: the DE will post the request to the WebRTC Ingest Signaling over HTTPS (wish) Working Group mailing list (or a successor mailing list designated by the IESG).

If the request comes from within the IETF, it should be documented in an Internet-Draft. Lastly, the DE must ensure that any other request for a code point does not conflict with work that is active or already published within the IETF.

6.4.3. WHIP Protocol Extension Registration Template

A WHIP Protocol Extension URNs is defined by completing the following template:

- *URN: A unique URN for the WHIP Protocol Extension (e.g., "urn:ietf:params:whip:ext:example:server-sent-events").

- *Reference: A formal reference to the publicly available specification

- *Name: A descriptive name of the WHIP Protocol Extension extension (e.g., "Sender Side events").

- *Description: A brief description of the function of the extension, in a short paragraph or two

- *Contact information: Contact information for the organization or person making the registration

7. Acknowledgements

The authors wish to thank Lorenzo Miniero, Juliusz Chroboczek, Adam Roach, Nils Ohlmeier, Christer Holmberg, Cameron Elliott, Gustavo Garcia, Jonas Birme, Sandro Gauci and everyone else in the WebRTC community that have provided comments, feedback, text and

improvement proposals on the document and contributed early implementations of the spec.

8. References

8.1. Normative References

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, June 2017.

[BCP9] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.
Dusseault, L. and R. Sparks, "Guidance on Interoperation and Implementation Reports for Advancement to Draft Standard", BCP 9, RFC 5657, September 2009.
Housley, R., Crocker, D., and E. Burger, "Reducing the Standards Track to Two Maturity Levels", BCP 9, RFC 6410, October 2011.
Resnick, P., "Retirement of the "Internet Official Protocol Standards" Summary Document", BCP 9, RFC 7100, December 2013.
Kolkman, O., Bradner, S., and S. Turner, "Characterization of Proposed Standards", BCP 9, RFC 7127, January 2014.
Dawkins, S., "Increasing the Number of Area Directors in an IETF Area", BCP 9, RFC 7475, March 2015.
Halpern, J., Ed. and E. Rescorla, Ed., "IETF Stream Documents Require IETF Rough Consensus", BCP 9, RFC 8789, June 2020.
Rosen, B., "Responsibility Change for the RFC Series", BCP 9, RFC 9282, June 2022.

[FETCH] WHATWG, "Fetch - Living Standard", n.d., <https://fetch.spec.whatwg.org>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/rfc/rfc3264>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <https://www.rfc-editor.org/rfc/rfc3553>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/rfc/rfc4122>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/rfc/rfc5789>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/rfc/rfc6585>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/rfc/rfc6750>.

[RFC7064] Nandakumar, S., Salgueiro, G., Jones, P., and M. Petit-Huguenin, "URI Scheme for the Session Traversal Utilities for NAT (STUN) Protocol", RFC 7064, DOI 10.17487/RFC7064, November 2013, <https://www.rfc-editor.org/rfc/rfc7064>.

[RFC7065] Petit-Huguenin, M., Nandakumar, S., Salgueiro, G., and P. Jones, "Traversal Using Relays around NAT (TURN) Uniform Resource Identifiers", RFC 7065, DOI 10.17487/RFC7065, November 2013, <https://www.rfc-editor.org/rfc/rfc7065>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/

RFC7675, October 2015, <https://www.rfc-editor.org/rfc/rfc7675>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/rfc/rfc8288>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8489] Petit-Huguenin, M., Salgueiro, G., Rosenberg, J., Wing, D., Mahy, R., and P. Matthews, "Session Traversal Utilities for NAT (STUN)", RFC 8489, DOI 10.17487/RFC8489, February 2020, <https://www.rfc-editor.org/rfc/rfc8489>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

[RFC8819] Hopps, C., Berger, L., and D. Bogdanovic, "YANG Module Tags", RFC 8819, DOI 10.17487/RFC8819, January 2021, <https://www.rfc-editor.org/rfc/rfc8819>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/RFC8826, January 2021, <https://www.rfc-editor.org/rfc/rfc8826>.

[RFC8829] Uberti, J., Jennings, C., and E. Rescorla, Ed., "JavaScript Session Establishment Protocol (JSEP)", RFC 8829, DOI 10.17487/RFC8829, January 2021, <https://www.rfc-editor.org/rfc/rfc8829>.

[RFC8830] Alvestrand, H., "WebRTC MediaStream Identification in the Session Description Protocol", RFC 8830, DOI 10.17487/RFC8830, January 2021, <https://www.rfc-editor.org/rfc/rfc8830>.

[RFC8838] Ivov, E., Uberti, J., and P. Saint-Andre, "Trickle ICE: Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (ICE) Protocol", RFC 8838, DOI 10.17487/RFC8838, January 2021, <https://www.rfc-editor.org/rfc/rfc8838>.

[RFC8840] Ivov, E., Stach, T., Marocco, E., and C. Holmberg, "A Session Initiation Protocol (SIP) Usage for Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (Trickle ICE)", RFC 8840, DOI 10.17487/RFC8840, January 2021, <https://www.rfc-editor.org/rfc/rfc8840>.

[RFC8842] Holmberg, C. and R. Shpount, "Session Description Protocol (SDP) Offer/Answer Considerations for Datagram Transport Layer Security (DTLS) and Transport Layer Security (TLS)", RFC 8842, DOI 10.17487/RFC8842, January 2021, <https://www.rfc-editor.org/rfc/rfc8842>.

[RFC8853] Burman, B., Westerlund, M., Nandakumar, S., and M. Zanaty, "Using Simulcast in Session Description Protocol (SDP) and RTP Sessions", RFC 8853, DOI 10.17487/RFC8853, January 2021, <https://www.rfc-editor.org/rfc/rfc8853>.

[RFC8858] Holmberg, C., "Indicating Exclusive Support of RTP and RTP Control Protocol (RTCP) Multiplexing Using the Session Description Protocol (SDP)", RFC 8858, DOI 10.17487/RFC8858, January 2021, <https://www.rfc-editor.org/rfc/rfc8858>.

[RFC8863] Holmberg, C. and J. Uberti, "Interactive Connectivity Establishment Patiently Awaiting Connectivity (ICE PAC)", RFC 8863, DOI 10.17487/RFC8863, January 2021, <https://www.rfc-editor.org/rfc/rfc8863>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/rfc/rfc9112>.

[RFC9143] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 9143, DOI 10.17487/RFC9143, February 2022, <https://www.rfc-editor.org/rfc/rfc9143>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

[W3C.REC-ldp-20150226]
Malhotra, A., Ed., Arwe, J., Ed., and S. Speicher, Ed., "Linked Data Platform 1.0", W3C REC REC-ldp-20150226, W3C REC-ldp-20150226, 26 February 2015, <https://www.w3.org/TR/2015/REC-ldp-20150226/>.

8.2. Informative References

[RFC3261]
Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/rfc/rfc3261>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/rfc/rfc6120>.

[RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <https://www.rfc-editor.org/rfc/rfc7826>.

[RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017, <https://www.rfc-editor.org/rfc/rfc8141>.

[W3C.REC-webrtc-20210126] Jennings, C., Ed., Boström, H., Ed., and J. Bruaroey, Ed., "WebRTC 1.0: Real-Time Communication Between Browsers", W3C REC REC-webrtc-20210126, W3C REC-webrtc-20210126, 26 January 2021, <https://www.w3.org/TR/2021/REC-webrtc-20210126/>.

Authors' Addresses

Sergio Garcia Murillo
Millicast

Email: sergio.garcia.murillo@cosmosoftware.io

Alexandre Gouaillard
CoSMo Software

Email: alex.gouaillard@cosmosoftware.io