Stream:	Internet Engineering Task Force (IETF)	
RFC:	9725	
Updates:	8840, 8842	
Category:	Standards Track	
Published:	March 2025	
ISSN:	2070-1721	
Authors:	S. Garcia Murillo	A. Gouaillard
	Millicast	*CoSMo Software*

RFC 9725
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 Content Delivery Networks (CDNs).

This document updates RFCs 8840 and 8842.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9725.

Table of Contents

1. Introduction

The IETF RTCWEB Working Group standardized the JavaScript Session Establishment Protocol (JSEP) [RFC9429], 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 the application level.

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

While WebRTC can be integrated with standard signaling protocols like SIP [RFC3261] or Extensible Messaging and Presence Protocol (XMPP) [RFC6120], they are not designed to be used in broadcasting and streaming services, and there is also no sign of adoption in that industry. The Real-Time Streaming Protocol (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 a media ingestion method that:

- 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 classical 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, and
- is usable in both web browsers and 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 Ingestion 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, which is 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. This means that no modifications to the "m=" sections can be made after the initial SDP offer/answer exchange via HTTP POST is completed and that only ICE-related information can be updated via HTTP PATCH requests as defined in Section 4.3.

The following diagram illustrates the core operation of WHIP for initiating and terminating an ingest session:

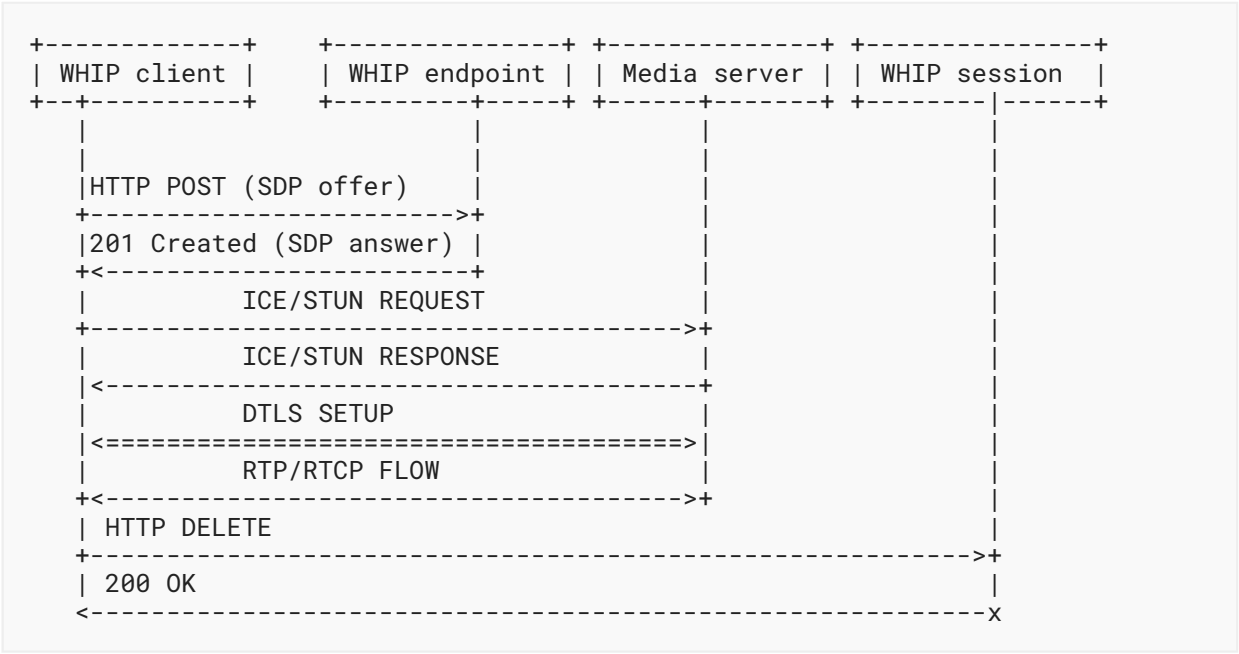

Figure 1: WHIP Session Setup and Teardown

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 WHIP 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: This 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: This indicates the server handling the allocated HTTP resource by the WHIP endpoint for an ongoing ingest session.

WHIP session URL: This refers to the URL of the WHIP resource allocated by the WHIP endpoint for a specific media session. To modify the session (e.g., ICE operations or session termination), the WHIP client can send requests to the WHIP session using this URL.

Figure 1 illustrates the communication flow between a WHIP client, WHIP endpoint, media server, and WHIP session. This flow outlines the process of setting up and tearing down an ingest session using WHIP, which involves negotiation, ICE for Network Address Translation (NAT) traversal, DTLS and the Secure Real-time Transport Protocol (SRTP) for security, and RTP/RTCP for media transport:

- The WHIP client initiates the communication by sending an HTTP POST with an SDP offer to the WHIP endpoint.
- The WHIP endpoint responds with a "201 Created" message containing an SDP answer.
- The WHIP client and media server establish ICE and DTLS sessions for NAT traversal and secure communication.
- RTP and RTCP flows are established for media transmission from the WHIP client to the media server, secured by the SRTP profile.
- The WHIP client sends an HTTP DELETE to terminate the WHIP session.
- The WHIP session responds with a "200 OK" to confirm the session termination.

4. Protocol Operation

4.1. HTTP Usage

Following the guidelines in [BCP56], WHIP clients **MUST NOT** match error codes returned by the WHIP endpoints and resources to a specific error cause indicated in this specification. WHIP clients **MUST** be able to handle all applicable status codes by gracefully falling back to the generic n00 semantics of a given status code on unknown error codes. WHIP endpoints and resources could convey finer-grained error information by a problem details json object in the response message body of the failed request as per [RFC9457].

The WHIP endpoints and sessions are origin servers as defined in Section 3.6 of [RFC9110]; they handle the requests and provide responses for the underlying HTTP resources. Those HTTP resources do not have any representation defined in this specification, so the WHIP endpoints and sessions **MUST** return a 2xx successful response with no content when a GET request is received.

4.2. Ingest Session Setup

In order to set up an ingest session, the WHIP client **MUST** generate an SDP offer according to the JSEP rules for an initial offer as per Section 5.2.1 of [RFC9429] and send an HTTP POST request as per Section 9.3.3 of [RFC9110] 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 **MUST** generate an SDP answer according to the JSEP rules for an initial answer as per Section 5.3.1 of [RFC9429] and return the following: 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. If the HTTP POST to the WHIP endpoint has a content type different than "application/sdp" or the SDP is malformed, the WHIP endpoint **MUST** reject the HTTP POST request with an appropriate 4xx error response.

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

Figure 2 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: 1101

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=ice-options:trickle ice2
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=fingerprint:sha-256 DA:7B:57:DC:28:CE:04:4F:31:79:85:C4:31:67:EB:
   27:58:29:ED:77:2A:0D:24:AE:ED:AD:30:BC:BD:F1:9C:02
a=setup:actpass
a=mid:0
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=sendonly
a=msid:d46fb922-d52a-4e9c-aa87-444eadc1521b ce326ecf-a081-453a-8f9f-
   0605d5ef4128
a=rtcp-mux
a=rtcp-mux-only
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10;useinbandfec=1
m=video 0 UDP/TLS/RTP/SAVPF 96 97
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 3956b460-40f4-4d05-acef-
   03abcdd8c6fd
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: 1053
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=ice-options:trickle ice2
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:70:86:4F:46:D9:2D:CC:D0:BC:81:9F:67:EF:34:2E:BD
a=candidate:1 1 UDP 2130706431 198.51.100.1 39132 typ host
a=setup:passive
a=mid:0
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=recvonly
a=rtcp-mux
a=rtcp-mux-only
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10;useinbandfec=1
m=video 0 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 0.0.0.0
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=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 the SDP Offer/Answer Exchange Done via an HTTP POST

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

To explicitly terminate a WHIP session, the WHIP client **MUST** send an HTTP DELETE request to the WHIP session 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 Section 5.2 of [RFC7675] for immediate revocation of consent.

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].

4.3. ICE Support

ICE [RFC8445] is a protocol that addresses the complexities of 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 [RFC8838] 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.

4.3.1. 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. This HTTP PATCH request **MUST** contain a body with an SDP fragment with media type "application/trickle-ice-sdpfrag" as specified in [RFC8840], which carries the relevant ICE information. If the HTTP PATCH request sent to the WHIP session URL has a content type different than "application/trickle-ice-sdpfrag" or the SDP fragment is malformed, the WHIP session **MUST** reject the HTTP PATCH with an appropriate 4xx error response.

If the WHIP session supports either Trickle ICE or ICE restarts, but not both, it **MUST** return a "422 Unprocessable Content" error response for the HTTP PATCH requests that are not supported as per Section 15.5.21 of [RFC9110].

The WHIP client **MAY** send overlapping HTTP PATCH requests to one WHIP session. Consequently, those HTTP PATCH requests may be received out of order by the WHIP session. Thus, if the WHIP session supports ICE restarts, it **MUST** generate a unique strong entity-tag identifying the ICE session as per Section 8.8.3 of [RFC9110]. 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, 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.

Missing or outdated ETags in the PATCH requests from WHIP clients will be answered by WHIP sessions as per Section 13.1.1 of [RFC9110] and Section 3 of [RFC6585], with a "428 Precondition Required" response for a missing entity-tag and a "412 Precondition Failed" response for a non-matching entity-tag.

4.3.2. 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 [RFC8445]. For the purpose of reducing setup times, when using Trickle ICE, the WHIP client **SHOULD** send the SDP offer (containing either locally gathered ICE candidates or an empty list of candidates) as soon as possible.

In order to simplify the protocol, the WHIP session cannot signal additional ICE candidates to the WHIP client after the SDP answer has been sent. 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.

As the WHIP client needs to know the WHIP session URL 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 the "201 Created" HTTP response to the initial POST request is received. In order to reduce 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 the response is received. Additionally, if ICE restarts are supported by the WHIP session, 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; thus, it **MUST** also wait and buffer any gathered candidates until it receives the HTTP response with the new entity-tag value to the last PATCH request performing an ICE restart.

WHIP clients generating the HTTP PATCH body with the SDP fragment and its subsequent processing by WHIP sessions **MUST** follow the guidelines defined in Section 4.4 of [RFC8840] with the following considerations:

- As per [RFC9429], only "m=" sections not marked as bundle-only can gather ICE candidates, so given that the "max-bundle" policy is being used, the SDP fragment will contain only the offerer-tagged "m=" line of the bundle group.
- The WHIP client **MAY** exclude ICE candidates from the HTTP PATCH body if they have already been confirmed by the WHIP session with a successful HTTP response to a previous HTTP PATCH request.

WHIP sessions and clients that support Trickle ICE **MUST** make use of entity-tags and conditional requests as explained in Section 4.3.1.

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.

Figure 3 shows an example of the Trickle ICE procedure where the WHIP client sends a PATCH request with updated ICE candidate information and receives a successful response from the WHIP session.

```
PATCH /session/id HTTP/1.1
Host: whip.example.com
If-Match: "xyzzy"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 576

a=group:BUNDLE 0 1
m=audio 9 UDP/TLS/RTP/SAVPF 111
a=mid:0
a=ice-ufrag:EsAw
a=ice-pwd:P2uYro0UCOQ4zxjKXaWCBui1
a=candidate:1387637174 1 udp 2122260223 192.0.2.1 61764 typ host
   generation 0 ufrag EsAw network-id 1
a=candidate:3471623853 1 udp 2122194687 198.51.100.2 61765 typ host
   generation 0 ufrag EsAw network-id 2
a=candidate:473322822 1 tcp 1518280447 192.0.2.1 9 typ host tcptype
   active generation 0 ufrag EsAw network-id 1
a=candidate:2154773085 1 tcp 1518214911 198.51.100.2 9 typ host
   tcptype active generation 0 ufrag EsAw network-id 2
a=end-of-candidates

HTTP/1.1 204 No Content
```

Figure 3: Example of a Trickle ICE Request and Response

4.3.3. ICE Restarts

As defined in [RFC8839], when an ICE restart occurs, a new SDP offer/answer exchange is triggered. However, as WHIP does not support renegotiation of non-ICE-related SDP information, a WHIP client will not send a new offer when an ICE restart occurs. Instead, the WHIP client and WHIP session will only exchange the relevant ICE information via an HTTP PATCH request as defined in Section 4.3.1 and **MUST** assume that the previously negotiated non-ICE-related SDP information still applies after the ICE restart.

When performing an ICE restart, the WHIP client **MUST** include the updated "ice-pwd" and "ice-ufrag" in the SDP fragment of the HTTP PATCH request body as well as the new set of gathered ICE candidates as defined in [RFC8840]. Similar to what is defined in Section 4.3.2, as per [RFC9429], only "m=" sections not marked as bundle-only can gather ICE candidates, so given that the "max-bundle" policy is being used, the SDP fragment will contain only the offerer-tagged "m=" line of the bundle group. A WHIP client sending a PATCH request for performing ICE restart **MUST** contain an If-Match header field with a field-value of "*" as per Section 13.1.1 of [RFC9110].

[RFC8840] states that an agent **MUST** discard any received requests containing "ice-pwd" and "ice-ufrag" attributes that do not match those of the current ICE Negotiation Session. However, any WHIP session receiving updated "ice-pwd" and "ice-ufrag" attributes **MUST** consider the request as performing an ICE restart instead and, if supported, **SHALL** return a "200 OK" with an "application/trickle-ice-sdpfrag" body containing the new ICE username fragment and password

and a new set of ICE candidates for the WHIP session. 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.

As defined in Section 4.4.1.1.1 of [RFC8839], the set of candidates after an ICE restart may include some, none, or all of the previous candidates for that data stream and may include a totally new set of candidates. Therefore, after performing a successful ICE restart, both the WHIP client and the WHIP session **MUST** replace the previous set of remote candidates with the new set exchanged in the HTTP PATCH request and response, discarding any remote ICE candidate not present on the new set. Both the WHIP client and the WHIP session **MUST** ensure that the HTTP PATCH request and response bodies include the same "ice-options," "ice-pacing," and "ice-lite" attributes as those used in the SDP offer or answer.

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 Section 5.1 of [RFC7675].

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 fragment and password 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 Session Traversal Utilities for NAT (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.

Figure 4 demonstrates a Trickle ICE restart procedure example. The WHIP client sends a PATCH request containing updated ICE information, including a new username fragment and password, along with newly gathered ICE candidates. In response, the WHIP session provides ICE information for the session after the ICE restart, including the updated username fragment and password, as well as the previous ICE candidate.

```
PATCH /session/id HTTP/1.1
Host: whip.example.com
If-Match: "*"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 82

a=ice-options:trickle ice2
a=group:BUNDLE 0 1
m=audio 9 UDP/TLS/RTP/SAVPF 111
a=mid:0
a=ice-ufrag:ysXw
a=ice-pwd:vw5LmwG4y/e6dPP/zAP9Gp5k
a=candidate:1387637174 1 udp 2122260223 192.0.2.1 61764 typ host
   generation 0 ufrag EsAw network-id 1
a=candidate:3471623853 1 udp 2122194687 198.51.100.2 61765 typ host
   generation 0 ufrag EsAw network-id 2
a=candidate:473322822 1 tcp 1518280447 192.0.2.1 9 typ host tcptype
   active generation 0 ufrag EsAw network-id 1
a=candidate:2154773085 1 tcp 1518214911 198.51.100.2 9 typ host
   tcptype active generation 0 ufrag EsAw network-id 2

HTTP/1.1 200 OK
ETag: "abccd"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 252

a=ice-lite
a=ice-options:trickle ice2
a=group:BUNDLE 0 1
m=audio 9 UDP/TLS/RTP/SAVPF 111
a=mid:0
a=ice-ufrag:289b31b754eaa438
a=ice-pwd:0b66f472495ef0ccac7bda653ab6be49ea13114472a5d10a
a=candidate:1 1 udp 2130706431 198.51.100.1 39132 typ host
a=end-of-candidates
```

Figure 4: Example of an ICE Restart Request and Response

4.4. WebRTC Constraints

To simplify the implementation of WHIP in both clients and media servers, WHIP introduces specific restrictions on WebRTC usage. The following subsections will explain these restrictions in detail.

4.4.1. SDP Bundle

Both the WHIP client and the WHIP endpoint **SHALL** support [RFC9143] and use the "max-bundle" policy as defined in [RFC9429]. The WHIP client and the media server **MUST** support multiplexed media associated with the BUNDLE group as per Section 9 of [RFC9143]. 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=" section as per Section 3 of [RFC8858].

4.4.2. Single MediaStream

WHIP only supports a single MediaStream as defined in [RFC8830]; therefore, all "m=" sections **MUST** contain a "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 MediaStreamTracks for the same media (audio or video). However, it would be possible for future revisions of this specification to allow more than a single MediaStream or MediaStreamTrack of each media kind. Therefore, in order to ensure forward compatibility, if the number of audio and/or video MediaStreamTracks or the number of MediaStreams are not supported by the WHIP endpoint, it **MUST** reject the HTTP POST request with a "422 Unprocessable Content" or "400 Bad Request" error response. The WHIP endpoint **MAY** also return a problem statement that provides further error details about the failed request, as recommended in Section 4.1.

4.4.3. No Partially Successful Answers

The WHIP endpoint **SHOULD NOT** reject individual "m=" sections, as specified in Section 5.3.1 of [RFC9429], if an error occurs when processing the "m=" section; instead, it **SHOULD** reject the HTTP POST request with a "422 Unprocessable Content" or "400 Bad Request" error response to prevent having partially successful ingest sessions, which can be misleading to end users. The WHIP endpoint **MAY** also return a problem statement as recommended in Section 4.1 proving further error details about the failed request.

4.4.4. 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 Content" or "400 Bad Request" error 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.

4.4.5. Trickle ICE and ICE Restarts

The media server **SHOULD** support full ICE, unless it is connected to the Internet with an IP address that is accessible by each WHIP client that is authorized to use it, in which case it **MAY** support only ICE lite. The WHIP client **MUST** implement and use full ICE.

Trickle ICE and ICE restart support is **OPTIONAL** for both the WHIP clients and media servers as explained in Section 4.3.

4.5. 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 Section 15.4 of [RFC9110]. In order to avoid POST requests being redirected as GET requests, status codes "301 Moved Permanently" and "302 Found" **MUST NOT** be used; the preferred method for performing load balancing is via the "307 Temporary Redirect" response status code as described in Section 15.4.8 of [RFC9110]. 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 Section 15.6.4 of [RFC9110], 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 Section 10.2.3 of [RFC9110].

4.6. 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 Traversal Using Relays around NAT (TURN) server, then this attribute specifies the username to use with that TURN server.
- credential: This attribute represents a long-term authentication password, as described in Section 9.2 of [RFC8489].

Figure 5 illustrates the Link headers included in a "201 Created" response, providing the ICE server URLs and associated credentials.

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

Figure 5: Example of a STUN/TURN Server's Configuration

NOTE: The naming of both the "rel" attribute value of "ice-server" and the target attributes follows that used in the RTCConfiguration dictionary in Section 4.2.1 of the W3C WebRTC recommendation (see [W3C.REC-webrtc-20250313]). The "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 Section 4.1.18 of [RFC9429]. In order to support these clients, the WHIP endpoint **MAY** also include the STUN/TURN server configuration in the responses to OPTIONS requests sent to the WHIP endpoint URL before the POST request is sent. However, this method is **NOT RECOMMENDED** to be used by the WHIP clients, and if it is supported by the underlying WHIP client's WebRTC implementation, the WHIP client **SHOULD** wait for the information to be returned by the WHIP endpoint in the response of the HTTP POST request instead.

The generation of the TURN server credentials may require sending 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 a POST value and 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.

4.6.1. Congestion Control

[RFC8836] defines the congestion control requirements for interactive real-time media to be used in WebRTC. These requirements are based on the assumption that the data needs to be provided continuously within a very limited time window (a delay of no more than hundreds of milliseconds end-to-end). If the latency target is higher, some of the requirements present in [RFC8836] could be relaxed to allow more flexible implementations.

4.7. Authentication and Authorization

All WHIP endpoints, sessions, and clients **MUST** support HTTP authentication as per Section 11 of [RFC9110]. Additionally, 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 Section 11.6 of [RFC9110].

4.7.1. 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 Section 2.1 of [RFC6750]. 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, are outside the scope of this document. Examples of tokens that could be used include, but are not limited to, JSON Web Tokens (JWTs) as per [RFC8725] and 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 Real Time Messaging Protocol (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.

4.8. 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 ingesting, it **MUST** negotiate the support in the SDP offer according to the procedures in Section 5.3 of [RFC8853]. A server accepting a simulcast offer **MUST** create an answer according to the procedures in Section 5.3.2 of [RFC8853].

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.

4.9. Protocol Extensions

In order to support future extensions to be defined for WHIP, 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 target attribute with an unknown "rel" attribute value, and WHIP sessions **MUST NOT** require the usage of any extension.

Each protocol extension **MUST** register a unique "rel" attribute value that starts with the prefix "urn:ietf:params:whip:ext" in the "WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs" registry (Section 6.4).

For example, consider a potential extension of server-to-client communication using server-sent events as specified in Section 9.2 of [HTML]. The URL for connecting to the server-sent event resource for the ingested 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:

Figure 6 shows the "201 Created" response to the HTTP POST request in this theoretical case (i.e., the WHIP extension supported by the WHIP session, as indicated in the Link header of the "201 Created" response).

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

Figure 6: Example of a WHIP 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: See [RFC8826]. HTTPS **SHALL** be used in order to preserve the WebRTC security model.
- Transport Layer Security (TLS): See [RFC8446] and [RFC9147].
- HTTP security: See Section 11 of [RFC9112] and Section 17 of [RFC9110].
- URI security: See Section 7 of [RFC3986].

On top of that, WHIP exposes a thin new attack surface specific to 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 for ingesting a WHIP stream to make multiple HTTP POST requests to the WHIP endpoint. This will force the WHIP endpoint to process the incoming SDP and allocate resources for being able to set up 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 they will not be able to process legitimate incoming ingests. 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 (IDORs) for WHIP session URLs: If the URLs returned by the WHIP endpoint for the location of WHIP sessions 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 IDentifiers (UUIDs) in Section 8 of [RFC9562] are applicable for generating the WHIP session URLs.
- HTTP PATCH flooding: Similar to the HTTP POST flooding, a malicious client could also create resource exhaustion by sending multiple HTTP PATCH requests 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

Per this specification, IANA has added a new link relation type and a new URN sub-namespace for WHIP. IANA has also created registries to manage entries within the "urn:ietf:params:whip" and "urn:ietf:params:whip:ext" namespaces.

6.1. Link Relation Type: ice-server

The link relation type below has been registered by IANA in the "Link Relation Types" registry 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: RFC 9725

6.2. URN Sub-namespace for WHIP (urn:ietf:params:whip)

IANA has added a new entry in the "IETF URN Sub-namespace for Registered Protocol Parameter Identifiers" registry, following the template in [RFC3553]:

Registry name: whip

Specification: RFC 9725

Repository: <https://www.iana.org/assignments/whip>

Index value: An IANA-assigned positive integer that identifies the registration. The first entry added to this registry uses the value 1, and this value is incremented for each subsequent entry added to the registry.

To manage this sub-namespace, IANA has created two registries within a new registry group called "WebRTC-HTTP Ingestion Protocol (WHIP)":

- "WebRTC-HTTP Ingestion Protocol (WHIP) URNs" registry (Section 6.3)
- "WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs" registry (Section 6.4)

6.3. WebRTC-HTTP Ingestion Protocol (WHIP) URNs Registry

The "WebRTC-HTTP Ingestion Protocol (WHIP) URNs" registry is used to manage entries within the "urn:ietf:params:whip" namespace. The registration procedure is "Specification Required" [RFC8126]. The registry contains the following fields: URN, Name, Reference, IANA Registry Reference, and Change Controller. This document is listed as the reference.

The registry contains a single initial entry:

URN: urn:ietf:params:whip:ext

Name: WebRTC-HTTP Ingestion Protocol (WHIP) extension URNs

Reference: Section 6.4 of RFC 9725

IANA Registry Reference: See "WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs" on <https://www.iana.org/assignments/whip>

Change Controller: IETF

6.4. WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs Registry

The "WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs" registry is used to manage entries within the "urn:ietf:params:whip:ext" namespace. The registration procedure is "Specification Required" [RFC8126]. The registry contains the following fields: URN, Name, Reference, IANA Registry Reference, and Change Controller. This document is listed as the reference.

A WHIP extension URN is used as a value in the "rel" attribute of the Link header as defined in Section 4.9 for the purpose of signaling the WHIP extensions supported by the WHIP endpoint. WHIP extension URNs have an "ext" type.

6.5. Registering WHIP URNs and WHIP Extension URNs

This section defines the process for registering new URNs in the "WebRTC-HTTP Ingestion Protocol (WHIP) URNs" registry (Section 6.3) and the "WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs" registry (Section 6.4).

6.5.1. Registration Procedure

The IETF has created a mailing list, <wish@ietf.org>, which can be used for public discussion of proposals prior to registration. Use of the mailing list is strongly encouraged. A designated expert (DE) [RFC8126], appointed by the IESG, will monitor the <wish@ietf.org> mailing list and review registrations.

Registration of new entries in the WHIP registries defined in this document **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 DE as per Section 4.6 of [RFC8126]. A Standards Track RFC is **REQUIRED** for the registration of new value data types that modify existing properties. A Standards Track RFC is also **REQUIRED** for registration of WHIP extension URNs that modify WHIP extensions previously documented in an existing RFC.

The registration procedure begins when a completed registration template, defined in Section 6.5.3, is sent to <iana@iana.org>. Decisions made by the DE can be appealed to an Applications and Real-Time (ART) Area Director, then to the IESG. The normal appeals procedure described in RFC 2026 [BCP9] is to be followed.

Once the registration procedure concludes successfully, IANA will create or modify the corresponding record in the "WebRTC-HTTP Ingestion Protocol (WHIP) URNs Registry" or "WebRTC-HTTP Ingestion Protocol (WHIP) Extension URNs" registry.

An RFC specifying one or more new WHIP extension URNs **MUST** include the completed registration template(s), which **MAY** be expanded with additional information. These completed template(s) 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.

6.5.2. Guidance for the Designated Expert

The DE is expected to do the following:

- Ascertain the existence of suitable documentation (a specification) as described in [RFC8126] and verify that the document is permanently and publicly available. Specifications should be documented in an Internet-Draft.
- Check the clarity of purpose and use of the requested registration.
- Verify that any request for one of these registrations has been made available for review and comments by posting the request to the <wish@ietf.org> mailing list.

- Ensure that any other request for a code point does not conflict with work that is active or already published by the IETF.

6.5.3. Registration Template

A WHIP extension URN is defined by completing the following template:

URN: A unique URN (e.g., "urn:ietf:params:whip:ext:example:server-sent-events")

Name: A descriptive name (e.g., "Sender Side events")

Reference: A formal reference to the publicly available specification

IANA Registry Reference: The registry related to the new URN

Change Controller: For Standards Track documents, this is "IETF". Otherwise, this is the name of the person or body that has change control over the specification.

7. References

7.1. Normative References

[FETCH] WHATWG, "Fetch", WHATWG Living Standard, <https://fetch.spec.whatwg.org>. Commit snapshot: <https://fetch.spec.whatwg.org/commit-snapshots/edfa8d100cf1ecfde385f65c172e0e8d018fcd98/>.

[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/info/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/info/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/info/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/info/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/info/rfc4086>.

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

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/info/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/info/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/info/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/info/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/info/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/info/rfc8174>.

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

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[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/info/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/info/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/info/rfc8725>.

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

[RFC8830] Alvestrand, H., "WebRTC MediaStream Identification in the Session Description Protocol", RFC 8830, DOI 10.17487/RFC8830, January 2021, <https://www.rfc-editor.org/info/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/info/rfc8838>.

[RFC8839] Petit-Huguenin, M., Nandakumar, S., Holmberg, C., Keränen, A., and R. Shpount, "Session Description Protocol (SDP) Offer/Answer Procedures for Interactive Connectivity Establishment (ICE)", RFC 8839, DOI 10.17487/RFC8839, January 2021, <https://www.rfc-editor.org/info/rfc8839>.

[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/info/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/info/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/info/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/info/rfc8858>.

[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/info/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/info/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/info/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/info/rfc9147>.

[RFC9429] Uberti, J., Jennings, C., and E. Rescorla, Ed., "JavaScript Session Establishment Protocol (JSEP)", RFC 9429, DOI 10.17487/RFC9429, April 2024, <https://www.rfc-editor.org/info/rfc9429>.

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, DOI 10.17487/RFC9562, May 2024, <https://www.rfc-editor.org/info/rfc9562>.

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

7.2. Informative References

[BCP56] Best Current Practice 56, <https://www.rfc-editor.org/info/bcp56>.
At the time of writing, this BCP comprises the following:

Nottingham, M., "Building Protocols with HTTP", BCP 56, RFC 9205, DOI 10.17487/RFC9205, June 2022, <https://www.rfc-editor.org/info/rfc9205>.

[BCP9] Best Current Practice 9, <https://www.rfc-editor.org/info/bcp9>.
At the time of writing, this BCP comprises the following:

Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

Dusseault, L. and R. Sparks, "Guidance on Interoperation and Implementation Reports for Advancement to Draft Standard", BCP 9, RFC 5657, DOI 10.17487/RFC5657, September 2009, <https://www.rfc-editor.org/info/rfc5657>.

Housley, R., Crocker, D., and E. Burger, "Reducing the Standards Track to Two Maturity Levels", BCP 9, RFC 6410, DOI 10.17487/RFC6410, October 2011, <https://www.rfc-editor.org/info/rfc6410>.

Resnick, P., "Retirement of the "Internet Official Protocol Standards" Summary Document", BCP 9, RFC 7100, DOI 10.17487/RFC7100, December 2013, <https://www.rfc-editor.org/info/rfc7100>.

Kolkman, O., Bradner, S., and S. Turner, "Characterization of Proposed Standards", BCP 9, RFC 7127, DOI 10.17487/RFC7127, January 2014, <https://www.rfc-editor.org/info/rfc7127>.

Dawkins, S., "Increasing the Number of Area Directors in an IETF Area", BCP 9, RFC 7475, DOI 10.17487/RFC7475, March 2015, <https://www.rfc-editor.org/info/rfc7475>.

Halpern, J., Ed. and E. Rescorla, Ed., "IETF Stream Documents Require IETF Rough Consensus", BCP 9, RFC 8789, DOI 10.17487/RFC8789, June 2020, <https://www.rfc-editor.org/info/rfc8789>.

Rosen, B., "Responsibility Change for the RFC Series", BCP 9, RFC 9282, DOI 10.17487/RFC9282, June 2022, <https://www.rfc-editor.org/info/rfc9282>.

[HTML] WHATWG, "HTML", WHATWG Living Standard, <https://html.spec.whatwg.org/>. Commit snapshot: <https://html.spec.whatwg.org/commit-snapshots/09db56ba9343c597340b2c7715f43ff9b10826f6/>.

[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/info/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/info/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/info/rfc7826>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8836] Jesup, R. and Z. Sarker, Ed., "Congestion Control Requirements for Interactive Real-Time Media", RFC 8836, DOI 10.17487/RFC8836, January 2021, <https://www.rfc-editor.org/info/rfc8836>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/info/rfc9457>.

[W3C.REC-webrtc-20250313] Jennings, C., Ed., Castelli, F., Ed., Boström, H., Ed., and J. Bruaroey, Ed., "WebRTC: Real-Time Communication in Browsers", W3C Recommendation, 13 March 2025, <https://www.w3.org/TR/2025/REC-webrtc-20250313/>. Latest version available at: <https://www.w3.org/TR/webrtc/>.

Acknowledgements

The authors wish to thank Lorenzo Miniero, Juliusz Chroboczek, Adam Roach, Nils Ohlmeier, Christer Holmberg, Cameron Elliott, Gustavo Garcia, Jonas Birme, Sandro Gauci, Christer Holmberg, 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.

Authors' Addresses

Sergio Garcia Murillo
Millicast
Email: sergio.garcia.murillo@cosmosoftware.io

Alexandre Gouaillard
CoSMo Software
Email: alex.gouaillard@cosmosoftware.io