Workgroup: wish
Internet-Draft: draft-ietf-wish-whip-02
Published: 7 March 2022
Intended Status: Standards Track
Expires: 8 September 2022
Authors: S. Murillo
CoSMo Software
A. Gouaillard
CoSMo Software

WebRTC-HTTP ingestion protocol (WHIP)

Abstract

While WebRTC has been very successful in a wide range of scenarios, its adoption in the broadcasting/streaming industry is lagging behind. Currently there is no standard protocol (like SIP or RTSP) designed for ingesting media into a streaming service using WebRTC and so content providers still rely heavily on protocols like RTMP for it.

These protocols are much older than WebRTC and by default lack some important security and resilience features provided by WebRTC with minimal overhead and additional latency.

The media codecs used for ingestion in older protocols tend to be limited and not negotiated. WebRTC includes support for negotiation of codecs, potentially alleviating transcoding on the ingest node (which can introduce delay and degrade media quality). Server side transcoding that has traditionally been done to present multiple renditions in Adaptive Bit Rate Streaming (ABR) implementations can be replaced with simulcasting and SVC codecs that are well supported by WebRTC clients. In addition, WebRTC clients can adjust client-side encoding parameters based on RTCP feedback to maximize encoding quality.

Encryption is mandatory in WebRTC, therefore secure transport of media is implicit.

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 September 2022.

Table of Contents

1. Introduction

RTCWEB standardized JSEP ([RFC8829]), a mechanism used to control the setup, management, and teardown of a multimedia session, how to apply it using the SDP Offer/Answer model and all the formats for the data sent over the wire (media, codec, encryption, ...). Also, WebRTC intentionally does not specify a signaling transport protocol at application level. This flexibility has allowed the

implementation of a wide range of services. However, those services are typically standalone silos which don't require interoperability with other services or leverage the existence of tools that can communicate with them.

In the broadcasting/streaming world, the usage of hardware encoders that make it very simple to plug in (SDI) cables carrying raw media, encode it in place, and push it to any streaming service or CDN ingest is already ubiquitous. It is the adoption of a custom signaling transport protocol for each WebRTC service has hindered broader adoption as an ingestion protocol.

While some standard signaling protocols are available that can be integrated with WebRTC, like SIP or XMPP, they are not designed to be used in broadcasting/streaming services, and there also is no sign of adoption in that industry. RTSP, which is based on RTP and may be the closest in terms of features to WebRTC, is not compatible with the WebRTC SDP offer/answer model.

In the specific case of media ingestion into a streaming service, some assumptions can be made about the server-side which simplifies the WebRTC compliance burden, as detailed in webrtc-gateway document [I-D.draft-alvestrand-rtcweb-gateways].

This document proposes a simple protocol for supporting WebRTC as media ingestion method which is:

* Easy to implement,

* As easy to use as current RTMP URIs.

* Fully compliant with WebRTC and RTCWEB specs.

* Allows for both ingest in traditional media platforms and ingest in WebRTC end-to-end platforms with the lowest possible latency.

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

* Usable both in web browsers and in native encoders.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

* WHIP client: WebRTC media encoder or producer that acts as a client of the WHIP protocol by encoding and delivering the media to a remote media server.

*WHIP endpoint: Ingest server receiving the initial WHIP request.

*WHIP endpoint URL: URL of the WHIP endpoint that will create the WHIP resource.

*Media Server: WebRTC media server or consumer that establishes the media session with the WHIP client and receives the media produced by it.

*WHIP resource: Allocated resource by the WHIP endpoint for an ongoing ingest session that the WHIP client can send requests for altering the session (ICE operations or termination, for example).

*WHIP resource URL: URL allocated to a specific media session by the WHIP endpoint which can be used to perform operations such as terminating the session or ICE restarts.

3. Overview

The WebRTC-HTTP ingest protocol (WHIP) uses an HTTP POST request to perform a single shot SDP offer/answer so an ICE/DTLS session can be established between the encoder/media producer (WHIP client) and the broadcasting ingestion endpoint (media server).

Once the ICE/DTLS session is set up, the media will flow unidirectionally from the encoder/media producer (WHIP client) to the broadcasting ingestion endpoint (media server). In order to reduce complexity, no SDP renegotiation is supported, so no tracks or streams can be added or removed once the initial SDP offer/answer over HTTP is completed.

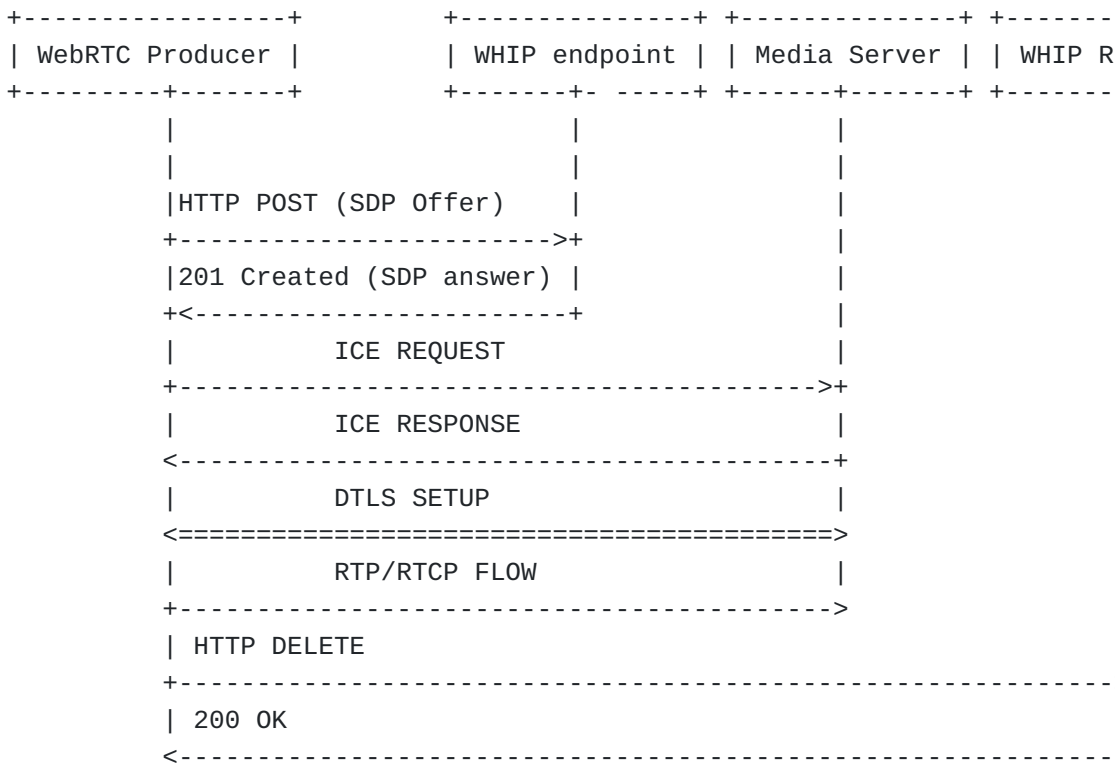

Figure 1: WHIP session setup and teardown

4. Protocol Operation

In order to setup an ingestion session, the WHIP client will generate an SDP offer according to the JSEP rules and do an HTTP POST request to the WHIP endpoint configured URL.

The HTTP POST request will 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 and the SDP answer as the body and a Location header pointing to the newly created resource.

The SDP offer SHOULD use the sendonly attribute and the SDP answer MUST use the recvonly attribute.

Once a session is setup, ICE consent freshness [RFC7675] will be used to detect abrupt disconnection and DTLS teardown for session termination by either side.

To explicitly terminate the session, the WHIP client MUST perform an HTTP DELETE request to the resource URL returned in the Location header of the initial HTTP POST. Upon receiving the HTTP DELETE request, the WHIP resource 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 HTTP 405 response for any HTTP GET, HEAD or PUT requests on the resource URL in order to reserve its usage for future versions of this protocol specification.

The WHIP resources MUST return an HTTP 405 response for any HTTP GET, HEAD, POST or PUT requests on the resource URL in order to reserve its usage for future versions of this protocol specification.

4.1. ICE and NAT support

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 only contain local candidates or even an empty list of candidates.

In order to simplify the protocol, there is no support for exchanging gathered trickle candidates from media server ICE candidates once the SDP answer is 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. The media server MAY use ICE lite, while the WHIP client MUST implement full ICE.

The WHIP client MAY perform trickle ICE or an ICE restarts [RFC8863] by sending a HTTP PATCH request to the WHIP resource URL with a body containing a SDP fragment with MIME type "application/trickle-ice-sdpfrag" as specified in [RFC8840] with the new ICE candidate or ICE ufrag/pwd for ICE restarts. A WHIP resource MAY not support trickle ICE (i.e. ICE lite media servers) or ICE restart, in that case, it MUST return a 405 Method Not Allowed response for any HTTP PATCH request.

As the HTTP PATCH request sent by a WHIP client may be received out of order by the WHIP resource, the WHIP resource MUST generate a unique strong entity-tag identifying the ICE session as per [RFC7232] section 2.3. The initial value of the entity-tag identifying the initial ICE session MUST be returned in an ETag header in the 201 response to the initial POST request to the WHIP endpoint and in the 200 OK of a PATCH request that triggers an ICE restart.

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

<SDP Offer>

HTTP/1.1 201 Created
ETag: "38sdf4fdsf54:EsAw"
Content-Type: application/sdp
Location: https://whip.example.org/resource/id

<SDP answer>
```

A WHIP client sending a PATCH request for performing trickle ICE MUST contain an If-Match header with the latest known entity-tag as per [RFC7232] section 3.1. When the PATCH request is received by the WHIP resource, it MUST compare the entity-tag value requested with the current entinty-tag of the resource as per [RFC7232] section 3.1 and return a 412 Precondition Failed response if they do not match. Entity-tag validation MUST only be used for HTTP requests requiring to match a known ICE session and SHOULD NOT be used otherwise, for example in the HTTP DELETE request to terminate the session.

A WHIP resource receiving a PATCH request with new ICE candidates, but which does not perform an ICE restart, MUST return a 204 No content response without body. If the media server does not support a candidate transport or is not able to resolve the connection address it MUST accept the HTTP request with the 204 response and silently discard the candidate.

```
PATCH /resource/id HTTP/1.1
Host: whip.example.com
If-Match: "38sdf4fdsf54:EsAw"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 548

a=ice-ufrag:EsAw
a=ice-pwd:P2uYro0UCOQ4zxjKXaWCBui1
m=audio 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 2: Trickle ICE request

A WHIP client sending a PATCH request for performing ICE restart MUST contain an If-Match header with a field-value "*" as per [RFC7232] section 3.1.

If the HTTP PATCH request results in an ICE restart, the WHIP resource SHALL return a 200 OK with an "application/trickle-ice-sdpfrag" body containing the new ICE username fragment and password and, optionally, the new set of ICE candidates for the media server and the new entity-tag correspond to the new ICE session in an ETag response header.

If the ICE request can not be performed by the WHIP resource it MUST return an appropriate HTTP error code but MUST NOT terminate the session immediately. The WHIP client COULD try again to perform a new ICE restart or terminate the session issuing a 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.

```
PATCH /resource/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: "289b31b754eaa438:ysXw"
Content-Type: application/trickle-ice-sdpfrag
Content-Length: 102

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

Figure 3: ICE restart request

Given that in order to send new ICE candidates to the WHIP resource, the WHIP client needs to know the entity-tag associated to the ICE session, it MUST buffer any gathered candidates before the HTTP response to the initial PUT request or the PATCH request with the new entity-tag value is received. Once the entity-tag value is known the WHIP client SHOULD send a single aggregated HTTP PATCH request with all the ICE candidates it has buffered so far.

4.2. WebRTC constraints

In order to reduce the complexity of implementing WHIP in both clients and media servers, some restrictions regarding WebRTC usage are made.

SDP bundle SHALL be used by both the WHIP client and the media server. The SDP offer created by the WHIP client MUST include the bundle-only attribute in all m-lines as per [RFC8843]. Also, RTCP muxing SHALL be supported by both the WHIP client and the media server.

Unlike [RFC5763] a WHIP client MAY use a setup attribute value of setup:active in the SDP offer, in which case the WHIP endpoint MUST use a setup attribute value of setup:passive in the SDP answer.

4.3. Load balancing and redirections

WHIP endpoints and media servers MAY 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 redirection via the 307 Temporary Redirect response code in the initial HTTP response to the WHIP endpoint URL. The WHIP resource URL MUST be a final one, and redirections are not required to be supported for the PATCH and DELETE request sent to it.

In case of high load, the WHIP endpoints MAY return a 503 (Service Unavailable) status code indicating that the server is currently unable to handle the request due to a temporary overload or scheduled maintenance, which will likely be alleviated after some delay.

The WHIP endpoint MAY send a Retry-After header field indicating the minimum time that the user agent is asked to wait before issuing the redirected request.

4.4. STUN/TURN server configuration

The WHIP endpoint MAY return ICE server configuration urls and credentials usable by the client in the 201 Created response to the HTTP POST request to the WHIP endpoint url.

Each ICE server will be returned on a Link header with a "rel" attribute value of "ice-server" where the Link target URI is the ICE server URL and the credentials are encoded in the Link target attributes as follows:

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

*credential: If 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 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"; userna
Link: turn:turn.example.net?transport=tcp; rel="ice-server"; userna
Link: turns:turn.example.net?transport=tcp; rel="ice-server"; usern
```

Figure 4: Example ICE server configuration

There are some webrtc implementations that do not support updating the ICE server configuration after the local offer has been created. In order to support these clients, the WHIP endpoint MAY also include the ICE server configuration on the responses to an authenticated OPTIONS request sent to the WHIP endpoint URL sent before the POST requests.

It COULD be also possible to configure the STUN/TURN server URLs with long term credentials provided by either the broadcasting service or an external TURN provider on the WHIP client overriding the values provided by the WHIP endpoint.

4.5. Authentication and authorization

WHIP endpoints and resources MAY require the HTTP request to be authenticated using an HTTP Authorization header 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 in all HTTP requests sent to either the WHIP endpoint or resource.

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.

WHIP endpoints and resources COULD perform the authentication and authorization by encoding an authentication token within the urls for the WHIP endpoints or resources instead. In case the WHIP client

is not configured to use a bearer token the HTTP Authorization header must not be sent in any request.

4.6. Simulcast and scalable video coding

Both simulcast and scalable video coding (including K-SVC modes) 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.

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 server MUST be advertised to the WHIP client on the 201 Created response to the initial HTTP POST request sent to the WHIP endpoint. The WHIP endpoint MUST return one Link header for each extension with the extension "rel" type attribute and the URI 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 servers MUST NOT require the usage of any of the extensions.

Each protocol extension MUST register a unique "rel" attribute values at IANA starting with the prefix: "urn:ietf:params:whip:".

For example, taking 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 side event resource for the published stream will be returned in the initial HTTP "201 Created" response with a "Link" header and a "rel" attribute of "urn:ietf:params:whip:server-sent-events".

The HTTP 201 response to the HTTP POST request would look like:

```
HTTP/1.1 201 Created
Content-Type: application/sdp
Location: https://whip.example.org/resource/id
Link: <https://whip.ietf.org/publications/213786HF/sse>;rel="urn:ietf:pa
```

5. Security Considerations

HTTPS SHALL be used in order to preserve the WebRTC security model.

6. IANA Considerations

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

6.1. Link Relation Type: ice-server

Relation Name: ice-server

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

Reference: TBD

7. Acknowledgements

8. Normative References

[I-D.draft-alvestrand-rtcweb-gateways] Alvestrand, H. and U. Rauschenbach, "WebRTC Gateways", Work in Progress, Internet-Draft, draft-alvestrand-rtcweb-gateways-02, 9 March 2015, <https://www.ietf.org/archive/id/draft-alvestrand-rtcweb-gateways-02.txt>.

[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>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/info/rfc5763>.

[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>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC

7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

[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>.

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

[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>.

[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/info/rfc8829>.

[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>.

[RFC8843] Holmberg, C., Alvestrand, H., and C. Jennings, "Negotiating Media Multiplexing Using the Session Description Protocol (SDP)", RFC 8843, DOI 10.17487/RFC8843, January 2021, <https://www.rfc-editor.org/info/rfc8843>.

[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>.

[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/info/rfc8863>.

Authors' Addresses

Sergio Garcia Murillo
CoSMo Software

Email: sergio.garcia.murillo@cosmosoftware.io

Alexandre Gouaillard
CoSMo Software

Email: alex.gouaillard@cosmosoftware.io