

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: December 11, 2021

S. Murillo  
A. Gouaillard  
CoSMo Software  
June 09, 2021

WebRTC-HTTP ingestion protocol (WHIP)  
draft-murillo-whip-02

## 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 in a streaming service, and content providers still rely heavily on protocols like RTMP for it.

These protocols are much older than webrtc and lack by default some important security and resilience features provided by webrtc with minimal delay.

The media codecs used in older protocols do not always match those being used in WebRTC, mandating transcoding on the ingest node, introducing delay and degrading media quality. This transcoding step is always present in traditional streaming to support e.g. ABR, and comes at no cost. However webrtc implements client-side ABR, also called Network-Aware Encoding by e.g. Huavision, by means of simulcast and SVC codecs, which otherwise alleviate the need for server-side transcoding. Content protection and Privacy Enhancement can be achieved with End-to-End Encryption, which preclude any server-side media processing.

This document proposes a simple HTTP based protocol that will allow WebRTC endpoints to ingest content into streaming services and/or CDNs to fill this gap and facilitate deployment.

## 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

Internet-Draft

whip

June 2021

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 December 11, 2021.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">2</a>
<a href="#">2.</a>	Terminology . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Overview . . . . .	<a href="#">4</a>
<a href="#">4.</a>	Protocol Operation . . . . .	<a href="#">5</a>
<a href="#">4.1.</a>	ICE and NAT support . . . . .	<a href="#">6</a>
<a href="#">4.2.</a>	WebRTC constraints . . . . .	<a href="#">7</a>
<a href="#">4.3.</a>	Load balancing and redirections . . . . .	<a href="#">7</a>
<a href="#">4.4.</a>	STUN/TURN server configuration . . . . .	<a href="#">7</a>
<a href="#">4.5.</a>	Authentication and authorization . . . . .	<a href="#">8</a>
<a href="#">4.6.</a>	Simulcast and scalable video coding . . . . .	<a href="#">8</a>
<a href="#">4.7.</a>	Protocol extensions . . . . .	<a href="#">8</a>
<a href="#">5.</a>	Security Considerations . . . . .	<a href="#">9</a>
<a href="#">6.</a>	IANA Considerations . . . . .	<a href="#">9</a>
<a href="#">7.</a>	Acknowledgements . . . . .	<a href="#">9</a>
<a href="#">8.</a>	Normative References . . . . .	<a href="#">9</a>
	Authors' Addresses . . . . .	<a href="#">10</a>

## [1.](#) Introduction

WebRTC intentionally does not specify a signaling transport protocol at application level, while RTCWEB standardized the signalling

protocol itself (JSEP, SDP O/A) and everything that was going over the wire (media, codec, encryption, ...). This flexibility has allowed for implementing a wide range of services. However, those services are typically standalone silos which don't require

Internet-Draft

whip

June 2021

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 would make it very simple to plug in (SDI) cables carrying raw media, encoding it in place, and pushing it to any streaming service or CDN ingest is ubiquitous. Having to implement a custom signalling transport protocol for each different webrtc services has hindered adoption.

While some standard signalling 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 maybe the closest in terms of features to webrtc, is not compatible with WebRTC SDP offer/answer model.

In the specific case of ingest into a platform, some assumption 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 ingest method which is:

- o Easy to implement,
- o As easy to use as current RTMP URIs.
- o Fully compliant with Webrtc and RTCWEB specs.
- o Allow for both ingest in traditional media platforms for extension and ingest in webrtc end-to-end platform for lowest possible latency.
- o Lowers the requirements on both hardware encoders and broadcasting

services to support webrtc.

- o 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](#)].

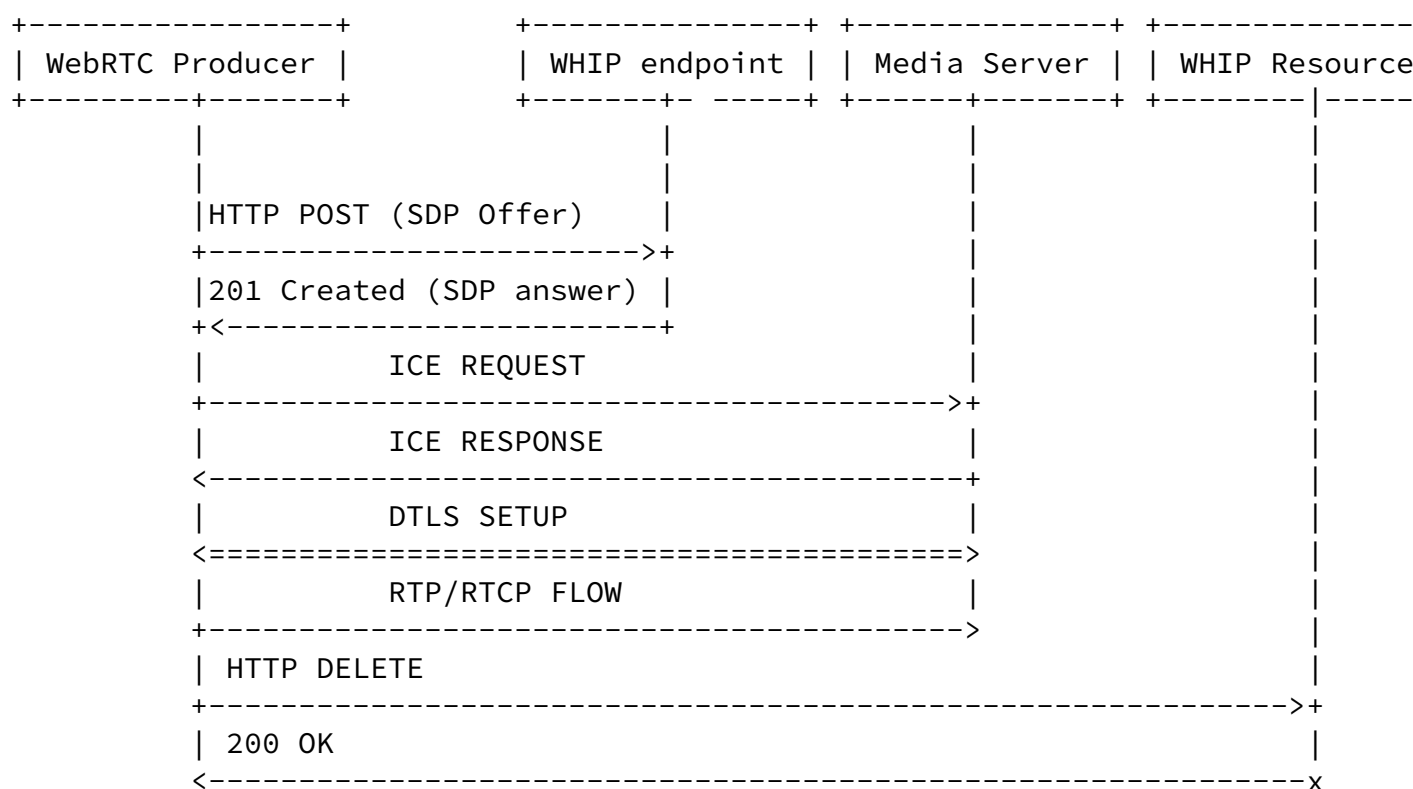
- o WHIP client: WebRTC Media encoder or producer that acts as client on the WHIP protocol and encodes and delivers the media to a remote media server.
- o WHIP endpoint: Ingest server receiving the initial WHIP request.
- o WHIP endpoint URL: URL of the WHIP endpoint that will create the WHIP resource
- o Media Server: WebRTC media server that establishes the media session with the WHIP client and receives the media produced by it.
- o WHIP resource: Allocated resource by the WHIP endpoint for an ongoing ingest session that the WHIP client can send request for altering the session (ICE operations or termination, for example).
- o WHIP resource URL: URL allocated to a specific media session by the WHIP endpoint which can be used to perform operations such 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 and the broadcasting ingestion endpoint.

Once the ICE/DTLS session is set up, the media will flow

unidirectionally from the encoder/media producer to the broadcasting ingestion endpoint. In order to reduce complexity, no SDP renegotiation is supported, so no tracks or streams can be added or removed once the initial SDP O/A over HTTP is completed.



#### [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 body. The WHIP endpoint will generate an SDP answer and return it on a 201 Accepted response with content type of application/sdp and the SDP answer as body and a Location header pointing to the newly created resource.

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 on 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

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. So in order to support the WHIP client behind NAT, the WHIP media server SHOULD be publicly accessible.

The initial offer by the WHIP client MAY be sent after the full ICE gathering is complete containing the full list of ICE candidates, or only contain local candidates or even an empty list of candidates.

The WHIP endpoint SDP answer SHALL contain the full list of ICE candidates publicly accessible 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 either trickle ICE (i.e. ICE lite media servers) or ICE restart, and it MUST return a 405 Method Not Allowed for any HTTP PATCH request.

A WHIP client receiving a 405 response for an HTTP PATCH request SHALL not send further request for ICE trickle or restart. If the WHIP client gathers additional candidates (via STUN/TURN) after the SDP offer is sent, it MUST send STUN request to the ICE candidates received from the media server as per [[RFC8838](#)] regardless if the HTTP PATCH is supported by either the WHIP client or the WHIP resource.

#### [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 307 Temporary Redirect response code.

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

Configuration of the TURN or STUN servers used by the WHIP client is out of the scope of this document.

It is RECOMMENDED that broadcasting server provides an HTTP interface for provisioning the TURN/STUN servers url and short term credentials as in [\[I-D.draft-uberti-behave-turn-rest-00\]](#). Note that the authentication information or the url of this API are not related to the WHIP endpoint URLs or authentication.

It could also be possible to configure the STUN/TURN server URLs and long term credentials provided by the either broadcasting service or an external TURN provider.

#### [4.5.](#) Authentication and authorization



Authentication and authorization is supported by the Authorization HTTP header with a bearer token as per [\[RFC6750\]](#).

#### [4.6.](#) Simulcast and scalable video coding

Both simulcast and scalable video coding (including K-SVC modes) MAY be supported by both media servers and WHIP clients and negotiated in the SDP O/A.

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 initial HTTP POST request to the WHIP endpoint by inserting 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 request 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 an 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.ietf.org/publications/213786HF> Link:  
<<https://whip.ietf.org/publications/213786HF/sse>>;rel="urn:ietf:params:whip:server-side-events " "

## 5. Security Considerations

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

## 6. IANA Considerations

## 7. Acknowledgements

## 8. Normative References

[I-D.[draft-alvestrand-rtcweb-gateways](#)]

Alvestrand, H. and U. Rauschenbach, "WebRTC Gateways",  
[draft-alvestrand-rtcweb-gateways-02](#) (work in progress),  
March 2015.

[I-D.[draft-uberti-behave-turn-rest-00](#)]

Uberti, J., "A REST API For Access To TURN Services",  
[draft-uberti-behave-turn-rest-00](#) (work in progress), July  
2013.

[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>>.

[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>>.

Internet-Draft

whip

June 2021

- [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>>.
- [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](mailto:sergio.garcia.murillo@cosmosoftware.io)

Alexandre Gouaillard  
CoSMo Software

Email: alex.gouaillard@cosmosoftware.io

Murillo & Gouaillard Expires December 11, 2021

[Page 10]