Abstract

This document describes a simple HTTP-based protocol that will allow WebRTC-based ingestion 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 26 January 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
Table of Contents

1. Introduction
2. Terminology
3. Overview
4. Protocol Operation
   4.1. ICE and NAT support
   4.2. WebRTC constraints
   4.3. Load balancing and redirections
   4.4. STUN/TURN server configuration
   4.5. Authentication and authorization
   4.6. Simulcast and scalable video coding
   4.7. Protocol extensions
5. Security Considerations
6. IANA Considerations
   6.1. Link Relation Type: ice-server
   6.2. Registration of WHIP URN Sub-namespace and WHIP Registry
   6.3. URN Sub-namespace for WHIP
      6.3.1. Specification Template
7. Acknowledgements
8. References
   8.1. Normative References
   8.2. Informative References
Authors' Addresses

1. Introduction

While WebRTC has been very successful in a wide range of scenarios, its adoption in the broadcasting/streaming industry is lagging behind.

The IETF RTCWEB working group standardized JSEP ([RFC8829]), a mechanism used to control the setup, management, and teardown of a multimedia session. It also describes how to negotiate media flows using the Offer/Answer Model with the Session Description Protocol (SDP) [RFC3264] as well as the formats for data sent over the wire (e.g., media types, codec parameters, and encryption). WebRTC intentionally does not specify a signaling transport protocol at application level. 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 use of hardware encoders that make it very simple to plug in cables carrying raw media, encode it in-place, and push it to any streaming service or CDN ingest is already ubiquitous. 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 [RFC3261] or XMPP [RFC6120], they are not designed to be used in broadcasting/streaming services, and there also is no sign of adoption in that industry. RTSP [RFC7826], which is based on RTP and may be the closest in terms of features to WebRTC, is not compatible with the SDP offer/answer model [RFC3264].

So, currently, there is no standard protocol designed for ingesting media into a streaming service using WebRTC and so content providers still rely heavily on protocols like RTMP for doing so. Most of those protocols are not RTP based, requiring media protocol translation when doing egress via WebRTC. Avoiding this media protocol translation is desirable as there is no functional parity between those protocols and WebRTC and it increases the implementation complexity at the media server side.

Also, the media codecs used in those protocols tend to be limited and not negotiated, not always matching the mediack codes supported in WebRTC. This requires transcoding on the ingest node, which introduces delay, degrades media quality and increases the processing workload required on the server side. Server side transcoding that has traditionally been done to present multiple renditions in Adaptive Bit Rate Streaming (ABR) implementations can be replaced with Simulcast [RFC8853] 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.

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

* Is easy to implement,

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

* Is fully compliant with WebRTC and RTCWEB specs

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

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

* Is usable both in web browsers and in native 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.

*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.
4. Protocol Operation

In order to set up an ingestion session, the WHIP client will generate an SDP offer according to the JSEP rules and perform an HTTP POST request to the configured WHIP endpoint 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", the SDP answer as the body, and a Location header field pointing to the newly created resource.

The SDP offer **SHOULD** use the "sendonly" attribute and the SDP answer **MUST** use the "recvonly" attribute.
POST /whip/endpoint HTTP/1.1
Host: whip.example.com
Content-Type: application/sdp
Content-Length: 1326

v=0
o=- 5228595038118931041 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group: BUNDLE 0 1
a=extmap-allow-mixed
a=msid-semantic: WMS
m=audio 9 UDP/TLS/RTP/SAVPF 111
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag: zjkk
a=ice-pwd: bP+XJMM09aR8AiX1jdukzR6Y
a=ice-options: trickle
a=setup: actpass
a=mid: 0
a=bundle-only
a=extmap: 4 urn:ietf:params:rtp-hdrext:sdes:mid
a=sendonly
a=msid:- d46fb922-d52a-4e9c-aa87-444eadc1521b
a=rtcp-mux
a=rtcpmap:111 opus/48000/2
a=fmtp:111 minptime=10; useinbandfec=1
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag: zjkk
a=ice-pwd: bP+XJMM09aR8AiX1jdukzR6Y
a=ice-options: trickle
a=setup: actpass
a=mid: 1
a=bundle-only
a=extmap: 4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap: 10 urn:ietf:params:rtp-hdrext:sdes:rtp-stream-id
a=sendonly
a=msid:- d46fb922-d52a-4e9c-aa87-444eadc1521b
a=rtcp-mux
a=rtcp-rsize
a=rtcpmap:96 VP8/90000
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
HTTP/1.1 201 Created
ETag: "38sdf4fdsf54:EsAw"
Content-Type: application/sdp
Content-Length: 1400
Location: https://whip.example.com/resource/id

v=0
o= 1657793490019 1 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE 0 1
a=extmap-allow-mixed
a=ice-lite
a=msid-semantic: WMS *
m=audio 9 UDP/TLS/RTP/SAVPF 111
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:526be20a538ee422
a=ice-pwd:2e13dde17c1cb009202f627fab90cbe358d766d049c9697
a=candidate:1 1 UDP 2130706431 198.51.100.1 39132 typ host
a=setup:passive
a=mid:0
a=bundle-only
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=recvonly
a=rtcp-mux
a=rtcp-rsize
a=rtpmap:111 opus/48000/2
a=fmtp:111 minptime=10;useinbandfec=1
m=video 9 UDP/TLS/RTP/SAVPF 96 97
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:526be20a538ee422
a=ice-pwd:2e13dde17c1cb009202f627fab90cbe358d766d049c9697
a=candidate:1 1 UDP 2130706431 198.51.100.1 39132 typ host
a=setup:passive
a=mid:1
a=bundle-only
a=extmap:4 urn:ietf:params:rtp-hdrext:sdes:mid
a=extmap:10 urn:ietf:params:rtp-hdrext:sdes:rt-payload-id
a=extmap:11 urn:ietf:params:rtp-hdrext:sdes:repaired-rt-payload-id
a=recvonly
a=rtcp-mux
a=rtcp-rsize
a=rtpmap:96 VP8/90000
a=rtcp-fb:96 ccm fir
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
a=rtpmap:97 rtx/90000
a=fmtp:97 apt=96
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 a session, the WHIP client **MUST** perform an HTTP DELETE request to the resource URL returned in the Location header field of the initial HTTP POST. Upon receiving the HTTP DELETE request, the WHIP 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 it **MAY** 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 ICE restarts [RFC8863] by sending an 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]. When used for trickle ICE, the body of this PATCH message will contain the new ICE candidate; when used for ICE restarts, it will contain a new ICE ufrag/pwd pair.

Trickle ICE and ICE restart support is **OPTIONAL** for a WHIP resource. If both Trickle ICE or ICE restarts are not supported by the WHIP resource, it **MUST** return a 405 Method Not Allowed response for any HTTP PATCH request. If the WHIP resource supports either Trickle ICE
or ICE restarts, but not both, it **MUST** return a 501 Not Implemented for the HTTP PATCH requests that are not supported.

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 [RFC9110] section 2.3. The initial value of the entity-tag identifying the initial ICE session **MUST** be returned in an ETag header field in the 201 response to the initial POST request to the WHIP endpoint. It **MUST** also be returned in the 200 OK of any PATCH request that triggers an ICE restart.

A WHIP client sending a PATCH request for performing trickle ICE **MUST** include an "If-Match" header field with the latest known entity-tag as per [RFC9110] section 3.1. When the PATCH request is received by the WHIP resource, it **MUST** compare the indicated entity-tag value with the current entity-tag of the resource as per [RFC9110] section 3.1 and return a "412 Precondition Failed" response if they do not match.

WHIP clients **SHOULD NOT** use entity-tag validation when matching a specific ICE session is not required, such as for example when initiating a DELETE request to terminate a session.

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:P2uYro0UCQQ4zxjjKXawCBui1
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 activ
a=candidate:2154773985 1 tcp 1518214911 198.51.100.2 9 typ host tcptype
a=end-of-candidates

HTTP/1.1 204 No Content
```
A WHIP client sending a PATCH request for performing ICE restart MUST contain an "If-Match" header field with a field-value "*" as per [RFC9110] section 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. The response may optionally contain 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 field.

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

 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: "289b31b754eea438:ysXw"
 Content-Type: application/trickle-ice-sdpfrag
 Content-Length: 102

 a=ice-lite
 a=ice-ufrag:289b31b754eea438
 a=ice-pwd:0b66f472495ef0ccac7bda653ab6be49ea13114472a5d10a

Because the WHIP client needs to know the entity-tag associated with the ICE session in order to send new ICE candidates, it MUST buffer any gathered candidates before it receives the HTTP response to the initial POST request or the PATCH request with the new entity-tag value. Once it knows the entity-tag value, the WHIP client SHOULD send a single aggregated HTTP PATCH request with all the ICE candidates it has buffered so far.
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(pwd) frags and ignore any further HTTP PATCH response received from a pending HTTP PATCH request. 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 client and at the server (because of an out-of-order request), the STUN requests will contain invalid ICE information and will be rejected by the server. When this situation is detected by the WHIP Client, it **SHOULD** send a new ICE restart request to the server.

### 4.2. WebRTC constraints

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-ietf-rtcweb-gateways].

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

Both the WHIP client and the WHIP endpoint **SHALL** use SDP bundle [RFC9143]. Each "m=" section **MUST** be part of a single BUNDLE group. Hence, when a WHIP client sends an SDP offer, it **MUST** include a "bundle-only" attribute in each bundled "m=" section. The WHIP client and the Media Server **MUST** support multiplexed media associated with the BUNDLE group as per [RFC9143] section 9. In addition, per [RFC9143] the WHIP client and Media Server will use RTP/RTCP multiplexing for all bundled media. The WHIP client and Media Server **SHOULD** include the "rtcp-mux-only" attribute in each bundled "m=" sections.

While this version of the specification only supports a single audio and video track, in order to ensure forward compatibility, if the number of audio and or video tracks or number streams is not supported by the WHIP Endpoint, it **MUST** reject the HTTP POST request with a 406 Not Acceptable error code.

Furthermore, the WHIP Endpoint **SHOULD NOT** reject individual "m=" sections as per [RFC8829] section 5.3.1 in case there is any error processing the "m=" section, but reject the HTTP POST request with a 406 Not Acceptable error code to prevent having partially successful WHIP sessions.

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

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

Tricke ICE and ICE restarts support is OPTIONAL for both the WHIP clients and Media Servers as explained in section 4.1.

4.3. Load balancing and redirections

WHIP endpoints and Media Servers might not be colocated on the same server, so it is possible to load balance incoming requests to different Media Servers. WHIP clients SHALL support HTTP redirection via the "307 Temporary Redirect" response code as described in [RFC9110] section 6.4.7. The WHIP resource URL MUST be a final one, and redirections are not required to be supported for the PATCH and DELETE requests 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 might send a Retry-After header field indicating the minimum time that the user agent ought to wait before making a follow-up request.

4.4. STUN/TURN server configuration

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

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 URL as defined in [RFC7064] and [RFC7065]. The credentials are encoded in the Link target attributes as follows:

*username: If the Link header field represents a TURN server, and credential-type is "password", then this attribute specifies the username to use with that TURN server.
credential: If the "credential-type" attribute is missing or has a "password" value, the credential attribute represents a long-term authentication password, as described in [RFC8489], Section 10.2.

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

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

Figure 5: Example ICE server configuration

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

There are some WebRTC implementations that do not support updating the STUN/TURN server configuration after the local offer has been created as specified in [RFC8829] section 4.1.18. In order to support these clients, the WHIP endpoint MAY also include the STUN/TURN server configuration on the responses to OPTIONS request sent to the WHIP endpoint URL before the POST request is sent.

The generation of the TURN server credentials may require performing a request to an external provider, which can both add latency to the OPTION 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 OPTION request is a preflight request for CORS, that is, if the OPTION request does not contain an Access-Control-Request-Method with "POST" value and the the Access-Control-Request-Headers HTTP header does not contain the "Link" value.

It might 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 field with a Bearer token as specified in [RFC6750] section 2.1. WHIP clients must implement this authentication and authorization mechanism and send the HTTP Authorization header field in all HTTP requests sent to either the WHIP endpoint or resource except the preflight OPTIONS requests for CORS.

WHIP endpoints and resources may require the HTTP request to be authenticated using an HTTP Authorization header field with a Bearer token as specified in [RFC6750] section 2.1. WHIP clients must implement this authentication and authorization mechanism and send the HTTP Authorization header field in all HTTP requests sent to either the WHIP endpoint or 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 (similar to the way RTMP URLs and Stream Keys are distributed).

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 field must not be sent in any request.

4.6. Simulcast and scalable video coding

Both Simulcast [RFC8853] and Scalable Video Coding (SVC), including K-SVC (also known as "S modes", in which multiple encodings are sent on the same SSRC), 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 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, 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 value at IANA starting with the prefix: "urn:ietf:params:whip:ext" as defined in {urn-whip-subspace}.

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

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

```text
HTTP/1.1 201 Created
Content-Type: application/sdp
Location: https://whip.example.com/resource/id
```

5. Security Considerations

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

6. IANA Considerations

This specification adds a new link relation type and a registry for URN sub-namespaces for WHIP protocol extensions.
6.1. Link Relation Type: ice-server

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

Relation Name: ice-server

Description: For the WHIP protocol, conveys the STUN and TURN servers that can be used by an ICE Agent to establish a connection with a peer.

Reference: TBD

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

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

To manage this sub-namespace, IANA has created the "System for Cross-domain Identity Management (WHIP) Schema URIs" registry, which is used to manage entries within the "urn:ietf:params:whip" namespace. The registry description is as follows:

*Registry name: WHIP

*Specification: this document (RFC TBD)

*Repository: See Section Section 6.3

*Index value: See Section Section 6.3

6.3. URN Sub-namespace for WHIP

WHIP Endpoint utilize URIs to identify the supported WHIP protocol extensions on the "rel" attribute of the Link header as defined in Section 4.7. This section creates and registers an IETF URN Sub-namespace for use in the WHIP specifications and future extensions.

6.3.1. Specification Template

Namespace ID:

The Namespace ID "whip" has been assigned.

Registration Information:
Version: 1
Date: TBD

Declared registrant of the namespace:
The Internet Engineering Task Force.

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

Declaration of Syntactic Structure:
The Namespace Specific String (NSS) of all URNs that use the "whip" Na
The keywords have the following meaning:
- type: The entity type. This specification only defines the "ext" typ
- name: A required US-ASCII string that conforms to the URN syntax req
- other: Any US-ASCII string that conforms to the URN syntax requirem

Relevant Ancillary Documentation:
None

Identifier Uniqueness Considerations:
The designated contact shall be responsible for reviewing and enforcin

Identifier Persistence Considerations:
Once a name has been allocated, it MUST NOT be reallocated for a diffe
The rules provided for assignments of values within a sub-namespace MU
This registration mechanism is not appropriate for naming values whose

Process of Identifier Assignment:
Namespace with type "ext" (e.g., "urn:ietf:params:whip:ext") is reserv

Process of Identifier Resolution:
None specified.

Rules for Lexical Equivalence:
No special considerations; the rules for lexical equivalence specified

Conformance with URN Syntax:
No special considerations.

Validation Mechanism:

None specified.

Scope:

Global.

7. Acknowledgements

8. References

8.1. Normative References


8.2. Informative References


Authors' Addresses

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

Alexandre Gouaillard
CoSMo Software

Email: alex.gouaillard@cosmosoftware.io