saag
Internet-Draft
Intended status: Informational
Expires: 19 October 2025

A. Tulshibagwale
SGNL
A. Deshpande
A. Parecki
Okta

17 April 2025

Push And Pull Based Security Event Token (SET) Delivery
draft-tulshibagwale-saag-pushpull-delivery-03

Abstract

This specification defines how multiple Security Event Tokens (SETs) can be delivered and received to and by an intended recipeint using HTTP POST over TLS or WebSocket binding.

This specification enabled following two use cases -

* In situations where a transmitter of Security Event Tokens (SETs) to a network peer is also a receiver of SETs from the same peer, it is helpful to have an efficient way of sending and receiving SETs in one HTTP transaction. In many cases, such as when using the OpenID Shared Signals Framework (SSF), the situation where each entity is both a transmitter and receiver is getting increasingly common.

* In situations where a transmitter of Security Event Tokens (SETs) wants to transmit multiple SETs to the reciever in a single HTTP call.

Using current mechanisms such as "Push-Based Delivery of Security Event Tokens (SETs) Using HTTP" or "Poll-Based Delivery of Security Event Tokens (SETs) Using HTTP" both require two or more HTTP connections to exchange SETs between peers. This is inefficient due to the latency of setting up each communication. This specification enables mutiple events to be transmitted in bi-directional transmission and reception of multiple SETs in one HTTP connection, and enables them to do so over a single HTTP or WebSocket binding.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://sgnl-ai.github.io/pushpull/. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tulshibagwale-saag-pushpull-delivery/.

Source for this draft and an issue tracker can be found at https://github.com/SGNL-ai/pushpull.

Status of This Memo

This Internet-Draft will expire on 19 October 2025.

Copyright Notice

Table of Contents

1. Introduction

Entities that exchange SETs [RFC8417] with each other ("Transceivers") can do so efficiently using the protocol defined in this specification. This specification extends the mechanisms described in the DeliveryPush [RFC8935] and DeliveryPoll [RFC8936] with the following additional mechanisms:

* A Transceiver initiating a communication can send multiple SETs in one HTTP connection to a Peer

* The Transceiver initiating communication can acknowledge previously received SETs in the same HTTP connection to the Peer

* The Peer responding to the communication can send multiple SETs in its response to a connection from the Transceiver

* The Peer responding to the communication can acknowledge previously received SETs in its response to the Transceiver

This specification also defines a mechanism by which a transmitter of a Security Event Token (SET) [RFC8417] can deliver multiple SETs to an intended SET Recipient via HTTP POST [RFC7231] over TLS in a

single POST call. [RFC8935] focuses on the delivery of the single SET to the receiver. This specification builds onto [RFC8935] to transmit multiple SETs to the receiver in a single POST call.

Multi-push SET delivery is intended to help in following scenarios:

* The transmitter of the SET has multiple outstanding SETs to be communicated to the receiver

* The transmitter wants to reduce the number of outbound calls to the same receiver to optimize performance, avoid being ratelimited when number of SETs to be communicated is high

* The receiver wants to optimize processing multiple SETs

2. Notational Conventions

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. Terminology

Transmitter A networked entity that can act as a transmitter of SETs. It communicates with other trusted Receivers to transmit using the protocol defined herein.

Receiver A networked entity that can act as a receiver of SETs. It communicates with other trusted Transmitter to receive SETs using the protocol defined herein.

Transceiver A networked entity that can act both as a transmitter of SETs and a receiver of SETs. It communicates with other trusted Transceivers to transmit and receive SETs using the protocol defined herein.

Peer Another name for a Transceiver, used to signify the other end of the communication from a Transceiver.

Initiator A Transceiver initiating communication with a Peer.

Responder A Transceiver responding to communication from a Peer.

DeliveryPush The IETF RFC titled "Push-Based Delivery of Security Event Tokens (SETs) Using HTTP" [RFC8935].

DeliveryPoll The IETF RFC titled "Poll-Based Delivery of Security Event Tokens (SETs) Using HTTP" [RFC8936].

4. Pushpull Endpoint

Each Transceiver that supports this specification MUST support a "Pushpull" endpoint. This endpoint MUST be capable of serving HTTP [RFC9110] requests. This endpoint MUST be TLS [RFC8446] enabled and MUST reject any communication not using TLS. The Pushpull endpoint MUST support the HTTP method POST and reject all other HTTP methods.

5. Communication Object

A Communication Object is a JSON object [RFC8259], and is a unit of communication defined in this specification for use in both requests and responses. When used in a request, the Initiator MAY include additional fields defined in the later sections below. The common fields of this object are:

sets OPTIONAL. A JSON object containing key-value pairs in which the key of a field is a string that contains the jti value of the SET that is specified in the value of the field. This field MAY be omitted to indicate that no SETs are being delivered by the initiator in this communication.

ack OPTIONAL. An array of strings, in which each string is the jti value of a previously received SET that is acknowledged in this object. This array MAY be empty or this field MAY be omitted to indicate that no previously received SETs are being acknowledged in this communication.

setErrs OPTIONAL. A JSON object containing key-value pairs in which the key of a field is a string that contains the jti value of a previously received SET that the sender of the communication object was unable to process. The value of the field is a JSON object that has the following fields:

err REQUIRED. The short reason why the specified SET failed to be processed.

description REQUIRED. An explanation of why the SET failed to be processed.

5.1. Example

The following is a non-normative example of a Communication Object

```
{
  "sets": {
    "dfc38da2-939e-4536-bec9-b8a16ed45c4e":
    "eyJhbGciOiJIUzI1NiJ9.
    eyJqdGkiOiJkZmMzOGRhMi05MzllLTQ1MzYtYmVjOS1iOGExNmVkNDVjNGUiLC
    JpYXQiOjE0NTg0OTY0MDQsImlzcyI6Imh0dHBzOi8vc2NpbS5leGFtcGxlLmNv
    bSIsImF1ZCI6WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vRmVlZHMvOThkNT
    I0NjFmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vc2NpbS5leGFtcGxlLmNv
    bS9GZWVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sImV2ZW50cyI6ey
    J1cm46aWV0ZjpwYXJhbXM6c2NpbTpldmVudDpjcmVhdGUiOnsicmVmIjoiaHR0
    cHM6Ly9zY2ltLmV4YW1wbGUuY29tL1VzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MW
    U3NTIxZDkiLCJhdHRyaWJ1dGVzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwi
    cGFzc3dvcmQiLCJlbWFpbHMiXX19fQ.XuVUJWrU6l80dcJ8bTRf-erMzFtQFYo
    kZLN--Kzd98o",
    "d93341ad-7329-4d1b-ba4a-9ff6f9f34003":
    "eyJhbGciOiJub25lIn0.
    eyJqdGkiOiIzZDBjM2NmNzk3NTg0YmQxOTNiZDBmYjFiZDRlN2QzMCIsImlhdC
    I6MTQ1ODQ5NjAyNSwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwi
    YXVkIjpbImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MW
    ZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9qaHViLmV4YW1wbGUuY29tL0Zl
    ZWRzLzVkNzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwic3ViIjoiaHR0cHM6Ly
    9zY2ltLmV4YW1wbGUuY29tL1VzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIx
    ZDkiLCJldmVudHMiOnsidXJuOmlldGY6cGFyYW1zOnNjaW06ZXZlbnQ6cGFzc3
    dvcmRSZXNldCI6eyJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkifSwi
    aHR0cHM6Ly9leGFtcGxlLmNvbS9zY2ltL2V2ZW50L3Bhc3N3b3JkUmVzZXRFeH
    QiOnsicmVzZXRBdHRlbXB0cyI6NX19fQ."
  },
  "ack": [
    "f52901c4-3996-11ef-9454-0242ac120002",
    "50924f49-55a9-47ca-820d-b1161cb0a602",
    "d563c724-79a0-4ff0-ba41-657fa5e2cb11"
  ],
  "setErrs": {
    "5c436b19-0958-4367-b408-2dd542606d3b" : {
      "err": "invalid subject",
      "description": "subject format not supported"
    }
  }
}
```


Figure 1: Example of a Communication Object

6. HTTP Request Response Binding

This section describes how Transceivers can use HTTP Requests and Responses to exchange Communication Objects described in Section 5.

6.1. Initiating Communication

A Transceiver can initiate communication with a Peer in order to:

* Send one or more SETs to the Peer.

* Positively or negatively acknowledge previously received SETs from the Peer.

* Both acknowledge previously received SETs from the Peer and send SETs to the Peer.

To initiate communication, the Initiator makes a HTTP POST request to the Responder’s Pushpull Endpoint Section 4. The body of this request is of the content type application/json. It contains a Communication Object Section 5, and the following additional field MAY be present:

maxResponseEvents OPTIONAL. A number which specifies the maximum number of events the Responder can include in its response to the Initiator. If this field is absent in the request, the Responder MAY include any number of events in the response. If this field is present, then the Responder MUST NOT include more events than the value of "maxResponseEvents" in its response to the specific request.

6.2. Response Communication

A Responder MUST respond to a communication from an Initiator by sending an HTTP Response.

6.2.1. Success Response

If the Responder is successful in receiving the request, it MUST return the HTTP status code 200 (OK). This status only indicates that the communication received was well formatted and was successfully parsed by the Responder. It does not indicate anything about whether any SETs in the communication were accepted or not.

The response MUST have the content-type application/json and the response MUST include a Communication Object Section 5.

6.2.2. Error Response

The Responder MUST respond with an error response if it is unable to process the request. This error response means that the responder was unable to parse the communication or the responder encountered a system error while attempting to process the communication. It does not indicate a positive or negative acknowledgement of any SETs in the communication.

The error response MUST include the appropriate error code as described in Section 2.4 of DeliveryPush [RFC8935].

6.2.3. Out Of Order Responses

A Communication Object in a Response may contain jti values in its ack or setErrs that do not correspond to the SETs received in the same Request to which the Response is being sent. They MAY consist of values received in previous Requests.

6.3. Example Request and Response

The following is a non-normative example of a request and its corresponding response

6.3.1. Request

```
POST /pushpull-endpoint HTTP/1.1
Host: sharedsignals-transceiver.myorg.example
Content-type: application/json
Authorization: Bearer eyJraWQiOiIyMDIwXzEiLCJJhbGciOiJSUzI1NiJ9...

{
  "ack": [],
  "sets": {
    "9deb50b0-d2f8-4793-a420-5e5678cf25a8":
    "eyJhbGciOiJIUzI1NiJ9.
    eyJqdGkiOiI5ZGViNTBiMC1kMmY4LTQ3OTMtYTQyMC01ZTU2NzhjZjI1YTgiLC
    JpYXQiOjE0NTg0OTY0MDQsImlzcyI6Imh0dHBzOi8vc2NpbS5leGFtcGxlLmNv
    bSIsImF1ZCI6WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vRmVlZHMvOThkNT
    I0NjFmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vc2NpbS5leGFtcGxlLmNv
    bS9GZWVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sImV2ZW50cyI6ey
    J1cm46aWV0ZjpwYXJhbXM6c2NpbTpldmVudDpjcmVhdGUiOnsicmVmIjoiaHR0
    cHM6Ly9zY2ltLmV4YW1wbGUuY29tL1VzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MW
    U3NTIxZDkiLCJhdHRyaWJ1dGVzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwi
    cGFzc3dvcmQiLCJlbWFpbHMiXX19fQ.KAaZj082ge8I1AiXfnmYw49ILFc5hEA
    tTZC9LkGg7IA",
    "d93341ad-7329-4d1b-ba4a-9ff6f9f34003":
    "eyJhbGciOiJIUzI1NiJ9.
    eyJqdGkiOiJkOTMzNDFhZC03MzI5LTRkMWItYmE0YS05ZmY2ZjlmMzQwMDMiLC
    JpYXQiOjE0NTg0OTYwMjUsImlzcyI6Imh0dHBzOi8vc2NpbS5leGFtcGxlLmNv
    bSIsImF1ZCI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNT
    I0NjFmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNv
    bS9GZWVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInN1YiI6Imh0dH
    BzOi8vc2NpbS5leGFtcGxlLmNvbS9Vc2Vycy80NGY2MTQyZGY5NmJkNmFiNjFl
    NzUyMWQ5IiwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50On
    Bhc3N3b3JkUmVzZXQiOnsiaWQiOiI0NGY2MTQyZGY5NmJkNmFiNjFlNzUyMWQ5
    In0sImh0dHBzOi8vZXhhbXBsZS5jb20vc2NpbS9ldmVudC9wYXNzd29yZFJlc2
    V0RXh0Ijp7InJlc2V0QXR0ZW1wdHMiOjV9fX0.IGbGOmSBtyS8wOGyMhWHe83v
    YgbGjUoezk-cIpYzVeY"
  },
  "setErrs": {
    "5c436b19-0958-4367-b408-2dd542606d3b" : {
      "err": "invalid subject",
      "description": "subject format not supported"
    }
  },
  "maxResponseEvents": 10
}
```

Figure 2: Example Pushpull request

In the above example request, the Initiator does not acknowledge any previous events, delivers two SETs, and reports an error on a previously received SET.

6.3.2. Response

The following is a non-normative example of a response:

```
HTTP/1.1 200 OK
Content-type: application/json

{
  "ack": [
    "d8d439e6-b103-47c7-86d9-d5951ce774d1"
  ],
  "sets": {
    "3f1c5fc7-99c5-4c2b-a9a3-68ea90be9ca9":
    "eyJ0eXAiOiJzZWNldmVudCtqd3QiLCJhbGciOiJIUzI1NiJ9.
    eyJpc3MiOiJodHRwczovL2lkcC5leGFtcGxlLmNvbS8iLCJqdGkiOiIzZjFjNW
    ZjNy05OWM1LTRjMmItYTlhMy02OGVhOTBiZTljYTkiLCJpYXQiOjE1MDgxODQ4
    NDUsImF1ZCI6IjYzNkM2OTY1NkU3NDVGNjk2NCIsImV2ZW50cyI6eyJodHRwcz
    ovL3NjaGVtYXMub3BlbmlkLm5ldC9zZWNldmVudC9yaXNjL2V2ZW50LXR5cGUv
    YWNjb3VudC1kaXNhYmxlZCI6eyJzdWJqZWN0Ijp7InN1YmplY3RfdHlwZSI6Im
    lzcy1zdWIiLCJpc3MiOiJodHRwczovL2lkcC5leGFtcGxlLmNvbS8iLCJzdWIi
    OiI3Mzc1NjI2QTY1NjM3NCJ9LCJyZWFzb24iOiJoaWphY2tpbmcifX19._Jwjs
    2M2AbxvPRRJJi5Kjl_Xepveugdd9Wb_Bh2Jj8s"
  }
}
```

Figure 3: Example Pushpull response

In the above example, the Responder acknowledges one of the SETs it previously received and provides a SET to deliver to the initiator. There are no errors reported by the Responder.

7. WebSocket Binding

Transceivers MAY use WebSockets [RFC6455] to send and receive Communication Objects described in Section 5. Since WebSockets are a symmetric protocol, a Transceiver MAY send a Communication Object at any time to its Peer. In such communication, a Transceiver sends a Communication Object as Payload data over the WebSocket protocol to a Peer. Similarly, a Transceiver MAY receive a Communication Object from a Peer over a WebSocket connection, wherein the Communication Object is the Payload data. In all such WebSocket communication, the Payload data does not have any Extension data in it.

7.1. Using WebSockets

During any communication initiated by a Transceiver, the Transceiver MAY request the Peer to use WebSockets [RFC6455] by requesting that the connection be upgraded to a WebSocket connection. If the Transceiver and its Peer can successfully perform the WebSocket handshake for the Pushpull Subprotocol described in Section 7.1.1, then the Transceiver and Peer MUST use WebSockets until the connection is closed. If the handshake fails, the Transceiver and Peer MAY use the HTTP Request Response Binding as described in Section 6

7.1.1. Pushpull Subprotocol Handshake

The Pushpull subprotocol is used to transport Communication Objects Section 5 over a WebSocket connection. The Transceiver and its Peer agree to this subprotocol during the WebSocket handshake (see Section 1.3 of [RFC6455]).

During the Websocket handshake, the Initiator MUST include the value pushpull in the list of protocols for the Sec-WebSocket-Protocol header. The reply from the Responder MUST also include the value pushpull in the list of values in its own Sec-WebSocket-Protocol header, in order for the Initiator and Responder to use WebSockets.

8. Authentication and Authorization

8.1. Verifying the Responder

The Initiator MUST verify the identity of the Responder by validating the TLS certification presented by the Responder, and verifying that it is the intended recipient of the request, before sending the Communication Object Section 5.

8.2. Verifying the Initiator

The Responder MUST verify the identity and authorization of the Initiator. The mechanism by which the Responder verifies the Initiator is out of scope of this specification, but may include mechanisms such as Mutual TLS (MTLS) client authentication, or OAuth access tokens.

The Initiator MUST attempt to obtain the OAuth Protected Resource Metadata [OPRM] for the Responder endpoint. If such metadata is found, the Initiator MUST obtain an access token using an OAuth authorization server discovered in the metadata. If no such metadata is found, then the mechanism of authenticating to the Responder is out of scope of this specification.

9. Delivery Reliability

A Transceiver MUST attempt to deliver any SETs it has previously attempted to deliver to a Peer until:

* It receives an acknowledgement through the ack value for that SET in a subsequent communication with the Peer

* It receives a setErrs object for that SET in a subsequent communication with the Peer

* It has attempted to deliver the SET a maximum number of times and has failed to communicate either due to communication errors or lack of inclusion in ack or setErrs in subsequent communications that were conducted for the maximum number of times. The maximum number of attempts MAY be set by the Transceiver for itself and SHOULD be communicated offline to the Peers.

If a Transceiver previously attempted to deliver a SET in a response to a Peer’s request, the Transceiver MAY Initiate a request to the Peer in order to retry delivery of the SET. A Peer MUST be able to either provide acks or setErrs for the same SETs either through requests or responses.

9.1. All SETs Accounted For

A Transceiver MUST ensure that it includes the jti value of each SET it receives, either in an ack or a setErrs value, to the Transceiver from which it received the SETs. A Transceiver SHOULD retry sending the same SET again if it was never responded to either in an ack value or in a setErrs value by a receiving Transceiver in a reasonable time period. A Transceiver MAY limit the number of times it retries sending a SET. A Transceiver MAY publish the retry time period and maximum number of retries to its peers, but such publication is outside the scope of this specification.

10. Conformance

This section describes conformance criteria for entities conforming to this specification.

10.1. Multi-Push

The "Multi-Push" conformance class enables a Transmitter to deliver multiple SETs to a Receiver. To conform to the "Multi-Push" class, a Transceiver acts as only one role: a Transmitter or Receiver.

* A Transmitter MUST support sets in the request Communication Object.

* A Receiver MUST support acks and setErrs in the response Communication Object.

10.2. Push-Pull

The "Push-Pull" conformance class enables bidirectional communication between Transceivers to both send and receive SETs in the same request and response. Transceivers MUST support sets, acks and setErrs in all Communication Objects.

11. Security Considerations

11.1. Authentication and Authorization

Transceivers MUST follow the procedures described in Section 8 in order to securely authenticate and authorize Peers.

11.2. HTTP and TLS

Transceivers MUST use TLS [RFC8446] to communicate with Peers and is subject to the security considerations of HTTP [RFC9110] Section 17.

11.3. Denial of Service

A Responder may be vulnerable to denial of service attacks wherein a large number of spurious requests need to be processed. Having efficient authorization mechanisms such as OAuth 2.0 [RFC6749] can mitigate such attacks by leveraging standard infrastructure that is designed to handle such attacks.

11.4. Temporary Disconnection

Transceivers must make sure they respond to each SET received in a timely manner as described in the "All SETs Accounted For" (Section 9.1). This ensures that if there was a temporary disconnection between two Transceivers, for example when a Responding Transceiver sent a Communication Object in the HTTP Response, that such disconnection is detected and the missing SETs can be retried.

12. Privacy Considerations

SETs may contain confidential information, and Transceivers receiving SETs must be careful not to log such content or ensure that sensitive information from the SET is redacted before logging.

13. IANA Considerations

The following WebSocket subprotocol will be added to the "WebSocket Subprotocol Name Registry" [IANA.WebSocket.Subprotocol]

* Subprotocol Identifier: pushpull
* Subprotocol Common Name: WebSocket transport for Pushpull delivery of SETs
* Subprotocol Definition: Section Section 7.1.1 of this document.

14. Normative References

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

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/rfc/rfc6455>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

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

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8417] Hunt, P., Ed., Jones, M., Denniss, W., and M. Ansari, "Security Event Token (SET)", RFC 8417, DOI 10.17487/RFC8417, July 2018, <https://www.rfc-editor.org/rfc/rfc8417>.

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

[RFC8935] Backman, A., Ed., Jones, M., Ed., Scurtescu, M., Ansari, M., and A. Nadalin, "Push-Based Security Event Token (SET) Delivery Using HTTP", RFC 8935, DOI 10.17487/RFC8935, November 2020, <https://www.rfc-editor.org/rfc/rfc8935>.

[RFC8936] Backman, A., Ed., Jones, M., Ed., Scurtescu, M., Ansari, M., and A. Nadalin, "Poll-Based Security Event Token (SET) Delivery Using HTTP", RFC 8936, DOI 10.17487/RFC8936, November 2020, <https://www.rfc-editor.org/rfc/rfc8936>.

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

[OPRM] Jones, M. B., Hunt, P., and A. Parecki, "OAuth 2.0 Protected Resource Metadata", May 2024, <https://datatracker.ietf.org/doc/draft-ietf-oauth-resource-metadata/>.

[IANA.WebSocket.Subprotocol] IANA, "WebSocket Subprotocol Name Registry", n.d., <https://www.iana.org/assignments/websocket/websocket.xml#subprotocol-name>.

Contributors

Erik Gustavson
SGNL
Email: erik@sgnl.ai

Authors' Addresses

Atul Tulshibagwale
SGNL
Email: atul@sgnl.ai

Apoorva Deshpande
Okta
Email: apoorva.deshpande@okta.com

Aaron Parecki
Okta
Email: aaron@parecki.com